Minerule

Generated by Doxygen 1.9.3

1 Namespace Index 1

1.1 Namespace List . . . 1

2 Hierarchical Index 3

2.1 Class Hierarchy . . . 3

3 Data Structure Index 7

3.1 Data Structures . . . 7

4 File Index 11

4.1 File List . . . 11

5 Namespace Documentation 15

5.1 minerule Namespace Reference . . . 15

5.1.1 Detailed Description . . . 19

5.1.2 Typedef Documentation . . . 20

5.1.2.1 Bits . . . 20

5.1.2.2 boolean . . . 21

5.1.2.3 EncodingBaseType . . . 21

5.1.2.4 GidList . . . 21

5.1.2.5 GroupMap . . . 21

5.1.2.6 GroupPair . . . 21

5.1.2.7 ItemSet . . . 21

5.1.3 Enumeration Type Documentation . . . 21

5.1.3.1 AlgorithmTypes . . . 21

5.1.3.2 MineruleErrors . . . 22

5.1.3.3 MiningTasks . . . 23

5.1.4 Function Documentation . . . 23

5.1.4.1 abort_handler() . . . 23

5.1.4.2 algorithmTypeToString() . . . 23

5.1.4.3 angledSpeenWheelUpdateHandler() . . . 24

5.1.4.4 barSpeenWheelUpdateHandler() . . . 24

5.1.4.5 buildStringWithListOfAlgorithmTypes() . . . 24

5.1.4.6 buildStringWithListOfMiningTasks() . . . 25

5.1.4.7 clockHandsUpdateHandler() . . . 25

5.1.4.8 floatFromText() . . . 26

5.1.4.9 getParserOutputObj() . . . 26

5.1.4.10 init_mrparser() . . . 26

5.1.4.11 initializeOptionsFromFile() . . . 26

5.1.4.12 initializeOptionsFromString() . . . 27

5.1.4.13 install_abort_handler() . . . 27

5.1.4.14 logRespectingMaxLogLength() . . . 27

5.1.4.15 me_error_begin() . . . 28

5.1.4.16 me_error_end() . . . 28

5.1.4.17 me_error_name() . . . 28
5.1.4.18 miningTaskToString() . . . 28
5.1.4.19 MRDebug() [1/2] . . . 28
5.1.4.20 MRDebug() [2/2] . . . 29
5.1.4.21 MRDebugPop() . . . 29
5.1.4.22 MRDebugPush() . . . 29
5.1.4.23 MRErr() [1/2] . . . 29
5.1.4.24 MRErr() [2/2] . . . 29
5.1.4.25 MRErrPop() . . . 30
5.1.4.26 MRErrPush() . . . 30
5.1.4.27 MRLog() [1/2] . . . 30
5.1.4.28 MRLog() [2/2] . . . 30
5.1.4.29 MRLogGetNewID() . . . 30
5.1.4.30 MRLogPop() . . . 31
5.1.4.31 MRLogPush() . . . 31
5.1.4.32 MRLogShowMeasurements() . . . 31
5.1.4.33 MRLogStartMeasuring() . . . 31
5.1.4.34 MRLogStopMeasuring() . . . 31
5.1.4.35 MRWarn() [1/2] . . . 32
5.1.4.36 MRWarn() [2/2] . . . 32
5.1.4.37 MRWarnPop() . . . 32
5.1.4.38 MRWarnPush() . . . 32
5.1.4.39 operator<<() [1/14] . . . 32
5.1.4.40 operator<<() [2/14] . . . 33
5.1.4.41 operator<<() [3/14] . . . 33
5.1.4.42 operator<<() [4/14] . . . 33
5.1.4.43 operator<<() [5/14] . . . 33
5.1.4.44 operator<<() [6/14] . . . 34
5.1.4.45 operator<<() [7/14] . . . 34
5.1.4.46 operator<<() [8/14] . . . 35
5.1.4.47 operator<<() [9/14] . . . 35
5.1.4.48 operator<<() [10/14] . . . 35
5.1.4.49 operator<<() [11/14] . . . 35
5.1.4.50 operator<<() [12/14] . . . 36
5.1.4.51 operator<<() [13/14] . . . 36
5.1.4.52 operator<<() [14/14] . . . 36
5.1.4.53 operator>>() . . . 36
5.1.4.54 parseMinerule() . . . 37
5.1.4.55 popOptionClassFromContext() . . . 37
5.1.4.56 pushOptionClassIntoContext() . . . 37
5.1.4.57 removeAllNodes() . . . 38
5.1.4.58 removeNode() . . . 38

5.1.4.59 setOption() . . . 38
5.1.4.60 stringToAlgorithmType() . . . 38
5.1.4.61 stringToMiningTask() . . . 39
5.1.4.62 stringWithListOfAlgorithmTypes() . . . 39
5.1.4.63 stringWithListOfMiningTasks() . . . 39
5.1.4.64 tickNumberUpdateHandler() . . . 39
5.1.4.65 trim() . . . 40
5.1.5 Variable Documentation . . . 40
5.1.5.1 context . . . 40
5.1.5.2 DB_LEVEL_ALL . . . 40
5.1.5.3 DEBUG_LEVEL . . . 40
5.1.5.4 DISJ_LOSS . . . 40
5.1.5.5 HAM_LOSS . . . 41
5.1.5.6 MineruleErrorsNames . . . 41
5.1.5.7 outputObj . . . 41
5.1.5.8 QUERY_LOSS . . . 41
5.2 mrc Namespace Reference . . . 41
5.2.1 Enumeration Type Documentation . . . 42
5.2.1.1 Results . . . 42
5.2.2 Function Documentation . . . 43
5.2.2.1 addDerivedQuery() . . . 43
5.2.2.2 checkCatalogue() . . . 43
5.2.2.3 checkQueryName() . . . 43
5.2.2.4 deleteQuery() . . . 44
5.2.2.5 doLoadOptions() . . . 44
5.2.2.6 installCatalogue() . . . 44
5.2.2.7 parseOptions() . . . 45
5.2.2.8 performCommands() . . . 46
5.2.2.9 printHelp() . . . 46
5.2.2.10 printMRQueryList() . . . 47
5.2.2.11 printVersion() . . . 47
5.2.2.12 uninstallCatalogue() . . . 48
5.3 mrdb Namespace Reference . . . 48
5.4 mrmatch Namespace Reference . . . 48
5.4.1 Typedef Documentation . . . 49
5.4.1.1 MatcherSpecs . . . 49
5.4.2 Enumeration Type Documentation . . . 49
5.4.2.1 ExitCodes . . . 49
5.4.2.2 MatchKind . . . 49
5.4.2.3 MatchOutput . . . 50
5.4.3 Function Documentation . . . 50
5.4.3.1 execute() . . . 50

5.4.3.2 sourceTableRequirements() . . . 51
5.5 mrprint Namespace Reference . . . 51
5.5.1 Enumeration Type Documentation . . . 51
5.5.1.1 MRPRINT_ERRORS . . . 51
5.5.2 Function Documentation . . . 52
5.5.2.1 getQueryName() . . . 52
5.5.2.2 newFormatter() . . . 52
5.5.2.3 printError() . . . 53
5.5.2.4 printRules() . . . 53

6 Data Structure Documentation **55**
6.1 __taglist_AND_node Struct Reference . . . 55
6.1.1 Detailed Description . . . 55
6.1.2 Field Documentation . . . 55
6.1.2.1 next . . . 55
6.1.2.2 sp . . . 55
6.2 __taglist_OR_node Struct Reference . . . 56
6.2.1 Detailed Description . . . 56
6.2.2 Field Documentation . . . 56
6.2.2.1 l_and . . . 56
6.2.2.2 next . . . 56
6.3 __tagminerule Struct Reference . . . 56
6.3.1 Detailed Description . . . 57
6.3.2 Field Documentation . . . 57
6.3.2.1 ca . . . 57
6.3.2.2 cc . . . 57
6.3.2.3 ga . . . 57
6.3.2.4 gc . . . 57
6.3.2.5 mc . . . 57
6.3.2.6 ra . . . 58
6.4 __tagsimple_pred Struct Reference . . . 58
6.4.1 Detailed Description . . . 58
6.4.2 Field Documentation . . . 58
6.4.2.1 op . . . 58
6.4.2.2 val1 . . . 58
6.4.2.3 val2 . . . 59
6.5 minerule::AggregateAntiMonoConstraint Class Reference . . . 59
6.5.1 Detailed Description . . . 59
6.5.2 Member Function Documentation . . . 59
6.5.2.1 check() [1/2] . . . 60
6.5.2.2 check() [2/2] . . . 60
6.5.3 Field Documentation . . . 60

6.5.3.1 attributeIndex . . . 60
6.6 minerule::AggregateConstraint Class Reference . . . 60
6.6.1 Detailed Description . . . 61
6.6.2 Constructor & Destructor Documentation . . . 61
6.6.2.1 AggregateConstraint() [1/2] . . . 61
6.6.2.2 AggregateConstraint() [2/2] . . . 61
6.6.3 Member Function Documentation . . . 61
6.6.3.1 check() [1/2] . . . 62
6.6.3.2 check() [2/2] . . . 62
6.6.4 Friends And Related Function Documentation . . . 62
6.6.4.1 operator<< . . . 62
6.6.4.2 operator>> . . . 62
6.6.5 Field Documentation . . . 62
6.6.5.1 attributeIndex . . . 63
6.7 minerule::AggregateMonoConstraint Class Reference . . . 63
6.7.1 Detailed Description . . . 63
6.7.2 Member Function Documentation . . . 63
6.7.2.1 check() [1/2] . . . 64
6.7.2.2 check() [2/2] . . . 64
6.7.3 Field Documentation . . . 64
6.7.3.1 attributeIndex . . . 64
6.8 minerule::Algorithms Class Reference . . . 64
6.8.1 Detailed Description . . . 65
6.8.2 Member Function Documentation . . . 65
6.8.2.1 executeExtractionAlgorithm() . . . 65
6.8.2.2 executeIncrementalAlgorithm() . . . 65
6.8.2.3 executeMinerule() . . . 66
6.8.2.4 getBestItemsetsMiningAlgorithm() . . . 67
6.8.2.5 getBestRulesMiningAlgorithm() . . . 67
6.8.2.6 getBestSequencesMiningAlgorithm() . . . 68
6.8.2.7 newAlgorithm() . . . 69
6.9 minerule::AlgorithmsOptions Class Reference . . . 69
6.9.1 Detailed Description . . . 69
6.9.2 Constructor & Destructor Documentation . . . 70
6.9.2.1 AlgorithmsOptions() . . . 70
6.9.3 Member Function Documentation . . . 70
6.9.3.1 getBodyCardinalities() . . . 70
6.9.3.2 getConfidence() . . . 70
6.9.3.3 getHeadCardinalities() . . . 71
6.9.3.4 getMiningAlgorithmsOptions() . . . 71
6.9.3.5 getOutTableName() . . . 71
6.9.3.6 getSourceRowDescription() . . . 71

6.9.3.7 getSupport() . . . 72
6.9.3.8 setBodyCardinalities() . . . 72
6.9.3.9 setConfidence() . . . 72
6.9.3.10 setConnection() . . . 72
6.9.3.11 setHeadCardinalities() . . . 73
6.9.3.12 setMiningAlgorithmsOptions() . . . 73
6.9.3.13 setOutTableName() . . . 73
6.9.3.14 setSourceRowDescription() . . . 74
6.9.3.15 setStatement() . . . 74
6.9.3.16 setSupport() . . . 74
6.10 minerule::Attr_def Struct Reference . . . 75
6.10.1 Detailed Description . . . 75
6.10.2 Constructor & Destructor Documentation . . . 75
6.10.2.1 Attr_def() . . . 75
6.10.3 Field Documentation . . . 75
6.10.3.1 attr_name . . . 75
6.10.3.2 attr_type . . . 76
6.11 Attribute Class Reference . . . 76
6.11.1 Detailed Description . . . 76
6.11.2 Constructor & Destructor Documentation . . . 76
6.11.2.1 Attribute() . . . 76
6.11.3 Member Function Documentation . . . 76
6.11.3.1 operator<() . . . 76
6.11.4 Field Documentation . . . 77
6.11.4.1 name . . . 77
6.12 AttributesUtil Class Reference . . . 77
6.12.1 Detailed Description . . . 77
6.12.2 Constructor & Destructor Documentation . . . 77
6.12.2.1 AttributesUtil() . . . 77
6.12.3 Member Function Documentation . . . 77
6.12.3.1 generatePositions() . . . 78
6.12.3.2 names_to_string() . . . 78
6.13 minerule::Bem_cond Class Reference . . . 78
6.13.1 Detailed Description . . . 79
6.13.2 Constructor & Destructor Documentation . . . 79
6.13.2.1 Bem_cond() [1/2] . . . 79
6.13.2.2 Bem_cond() [2/2] . . . 79
6.13.3 Member Function Documentation . . . 79
6.13.3.1 copyBemCond() . . . 80
6.13.4 Field Documentation . . . 80
6.13.4.1 and_c . . . 80
6.13.4.2 attr . . . 80

6.13.4.3 count_max . . . 80
6.13.4.4 count_min . . . 80
6.13.4.5 op . . . 81
6.13.4.6 type . . . 81
6.13.4.7 val . . . 81
6.14 minerule::BFSWithGidsAndCross Class Reference . . . 81
6.14.1 Detailed Description . . . 82
6.14.2 Constructor & Destructor Documentation . . . 82
6.14.2.1 BFSWithGidsAndCross() . . . 82
6.14.2.2 ∼BFSWithGidsAndCross() . . . 82
6.14.3 Member Function Documentation . . . 82
6.14.3.1 algorithmForType() . . . 83
6.14.3.2 canHandleMinerule() . . . 83
6.14.3.3 execute() . . . 83
6.14.3.4 getMineruleHasSameBodyHead() . . . 83
6.14.3.5 initialize() . . . 84
6.14.3.6 mineRules() . . . 84
6.14.3.7 optimizedMinerule() . . . 85
6.14.3.8 sourceTableRequirements() . . . 85
6.14.4 Field Documentation . . . 85
6.14.4.1 connection . . . 85
6.14.4.2 minerule . . . 86
6.14.4.3 options . . . 86
6.15 minerule::BFSWithGidsNoCross Class Reference . . . 86
6.15.1 Detailed Description . . . 87
6.15.2 Constructor & Destructor Documentation . . . 87
6.15.2.1 BFSWithGidsNoCross() . . . 87
6.15.2.2 ∼BFSWithGidsNoCross() . . . 87
6.15.3 Member Function Documentation . . . 87
6.15.3.1 algorithmForType() . . . 87
6.15.3.2 canHandleMinerule() . . . 88
6.15.3.3 execute() . . . 88
6.15.3.4 getMineruleHasSameBodyHead() . . . 88
6.15.3.5 initialize() . . . 88
6.15.3.6 mineRules() . . . 89
6.15.3.7 optimizedMinerule() . . . 90
6.15.3.8 sourceTableRequirements() . . . 90
6.15.4 Field Documentation . . . 90
6.15.4.1 connection . . . 90
6.15.4.2 minerule . . . 91
6.15.4.3 options . . . 91
6.16 minerule::BitString Class Reference . . . 91

6.16.1 Detailed Description . . . 92
6.16.2 Constructor & Destructor Documentation . . . 92
6.16.2.1 BitString() [1/3] . . . 92
6.16.2.2 BitString() [2/3] . . . 93
6.16.2.3 BitString() [3/3] . . . 93
6.16.3 Member Function Documentation . . . 93
6.16.3.1 clear() [1/2] . . . 93
6.16.3.2 clear() [2/2] . . . 93
6.16.3.3 count() . . . 94
6.16.3.4 gbits() . . . 94
6.16.3.5 getElm() . . . 94
6.16.3.6 getElmA() . . . 94
6.16.3.7 insert() . . . 94
6.16.3.8 invert() [1/2] . . . 95
6.16.3.9 invert() [2/2] . . . 95
6.16.3.10 length() . . . 95
6.16.3.11 moreThan() . . . 95
6.16.3.12 operator"!=() . . . 96
6.16.3.13 operator&() . . . 96
6.16.3.14 operator&=() . . . 96
6.16.3.15 operator=() . . . 97
6.16.3.16 operator==() . . . 97
6.16.3.17 operator[]() . . . 97
6.16.3.18 operator^=() . . . 97
6.16.3.19 operator"|=() . . . 98
6.16.3.20 print() [1/2] . . . 98
6.16.3.21 print() [2/2] . . . 98
6.16.3.22 reset() [1/2] . . . 99
6.16.3.23 reset() [2/2] . . . 99
6.16.3.24 serialize() . . . 99
6.16.3.25 set() [1/2] . . . 99
6.16.3.26 set() [2/2] . . . 100
6.16.3.27 setNBit() . . . 100
6.16.3.28 size() . . . 100
6.16.3.29 ssize() . . . 100
6.16.3.30 test() . . . 100
6.16.3.31 unserialize() . . . 101
6.16.4 Friends And Related Function Documentation . . . 101
6.16.4.1 operator<< . . . 101
6.16.4.2 operator>> . . . 101
6.16.5 Field Documentation . . . 102
6.16.5.1 intersections . . . 102

6.17 minerule::BitVector Class Reference . . . 102
6.17.1 Detailed Description . . . 102
6.17.2 Member Function Documentation . . . 102
6.17.2.1 operator&() . . . 103
6.17.2.2 operator"|() . . . 103
6.17.3 Field Documentation . . . 103
6.17.3.1 elements . . . 103
6.18 minerule::Body Class Reference . . . 103
6.18.1 Detailed Description . . . 104
6.18.2 Member Typedef Documentation . . . 104
6.18.2.1 RowBContainer . . . 104
6.18.3 Constructor & Destructor Documentation . . . 104
6.18.3.1 Body() . . . 105
6.18.3.2 ∼Body() . . . 105
6.18.4 Member Function Documentation . . . 105
6.18.4.1 extractRules() . . . 105
6.18.4.2 findBodiesInTree() . . . 106
6.18.4.3 findChildInTree() . . . 106
6.18.4.4 findRulesInTree() [1/2] . . . 106
6.18.4.5 findRulesInTree() [2/2] . . . 107
6.18.4.6 getAncestor() . . . 108
6.18.4.7 insertItemSetB() . . . 108
6.18.4.8 provaStampaLiv1() . . . 108
6.18.4.9 setAncestor() . . . 108
6.18.5 Field Documentation . . . 109
6.18.5.1 countb . . . 109
6.18.5.2 NRB . . . 109
6.19 minerule::BodyMap Class Reference . . . 109
6.19.1 Detailed Description . . . 110
6.19.2 Constructor & Destructor Documentation . . . 110
6.19.2.1 BodyMap() [1/2] . . . 111
6.19.2.2 BodyMap() [2/2] . . . 111
6.19.3 Member Function Documentation . . . 111
6.19.3.1 add() [1/2] . . . 111
6.19.3.2 add() [2/2] . . . 112
6.19.3.3 buildHead() [1/2] . . . 112
6.19.3.4 buildHead() [2/2] . . . 113
6.19.3.5 buildItemset() . . . 113
6.19.3.6 buildRules() [1/2] . . . 114
6.19.3.7 buildRules() [2/2] . . . 115
6.19.3.8 checkAntiMono() . . . 115
6.19.3.9 checkMono() . . . 115

6.19.3.10 closeOutputFiles() . . . 116
6.19.3.11 generateStartItemSets() . . . 116
6.19.3.12 howManyItemsets() . . . 116
6.19.3.13 nextID() . . . 117
6.19.3.14 openOutputFiles() . . . 117
6.19.3.15 pruneMap() . . . 117
6.19.3.16 saveItemset() . . . 117
6.19.3.17 saveItemsets() . . . 118
6.19.3.18 saveRules() . . . 118
6.19.3.19 setTotalGroups() . . . 118
6.19.3.20 updateCount() . . . 118
6.19.4 Field Documentation . . . 119
6.19.4.1 antiMonoConstr . . . 119
6.19.4.2 elements . . . 119
6.19.4.3 keys . . . 119
6.19.4.4 monoConstr . . . 119
6.20 minerule::BodyMapElement Class Reference . . . 119
6.20.1 Detailed Description . . . 121
6.20.2 Constructor & Destructor Documentation . . . 121
6.20.2.1 BodyMapElement() [1/2] . . . 121
6.20.2.2 BodyMapElement() [2/2] . . . 121
6.20.3 Member Function Documentation . . . 121
6.20.3.1 clear() [1/2] . . . 121
6.20.3.2 clear() [2/2] . . . 122
6.20.3.3 count() . . . 122
6.20.3.4 gbits() . . . 122
6.20.3.5 getElm() . . . 122
6.20.3.6 getElmA() . . . 122
6.20.3.7 insert() [1/3] . . . 123
6.20.3.8 insert() [2/3] . . . 123
6.20.3.9 insert() [3/3] . . . 123
6.20.3.10 invert() [1/2] . . . 123
6.20.3.11 invert() [2/2] . . . 123
6.20.3.12 length() . . . 124
6.20.3.13 moreThan() . . . 124
6.20.3.14 operator"!=() . . . 124
6.20.3.15 operator&() . . . 124
6.20.3.16 operator&=() . . . 125
6.20.3.17 operator==() . . . 125
6.20.3.18 operator[]() . . . 125
6.20.3.19 operator^=() . . . 126
6.20.3.20 operator"|=() . . . 126

6.20.3.21 print() [1/2] . . . 126
6.20.3.22 print() [2/2] . . . 127
6.20.3.23 pruneMap() . . . 127
6.20.3.24 reset() [1/2] . . . 127
6.20.3.25 reset() [2/2] . . . 127
6.20.3.26 serialize() . . . 128
6.20.3.27 set() [1/2] . . . 128
6.20.3.28 set() [2/2] . . . 128
6.20.3.29 setMinMax() . . . 128
6.20.3.30 setNBit() . . . 129
6.20.3.31 size() . . . 129
6.20.3.32 ssize() . . . 129
6.20.3.33 test() . . . 129
6.20.3.34 unserialize() . . . 129
6.20.3.35 updateCount() . . . 130
6.20.4 Field Documentation . . . 130
6.20.4.1 attribute . . . 130
6.20.4.2 counter . . . 130
6.20.4.3 done . . . 130
6.20.4.4 heads . . . 130
6.20.4.5 intersections . . . 131
6.21 minerule::CARE Class Reference . . . 131
6.21.1 Detailed Description . . . 132
6.21.2 Constructor & Destructor Documentation . . . 132
6.21.2.1 CARE() . . . 132
6.21.2.2 ∼CARE() . . . 132
6.21.3 Member Function Documentation . . . 132
6.21.3.1 algorithmForType() . . . 132
6.21.3.2 canHandleMinerule() . . . 133
6.21.3.3 execute() . . . 133
6.21.3.4 initialize() . . . 133
6.21.3.5 mineRules() . . . 134
6.21.3.6 optimizedMinerule() . . . 135
6.21.3.7 sourceTableRequirements() . . . 135
6.21.4 Field Documentation . . . 135
6.21.4.1 connection . . . 135
6.21.4.2 minerule . . . 135
6.21.4.3 options . . . 135
6.22 minerule::OptimizerCatalogue::Catalogue Class Reference . . . 136
6.22.1 Detailed Description . . . 136
6.22.2 Constructor & Destructor Documentation . . . 136
6.22.2.1 Catalogue() [1/2] . . . 136

6.22.2.2 Catalogue() [2/2] . . . 137
6.22.2.3 ~Catalogue() . . . 137
6.22.3 Member Function Documentation . . . 137
6.22.3.1 getSchemaInfo() . . . 137
6.22.3.2 initialize() . . . 137
6.22.3.3 insertMapping() . . . 138
6.22.3.4 stringToOrder() . . . 139
6.22.4 Field Documentation . . . 139
6.22.4.1 elements . . . 139
6.22.4.2 keys . . . 139
6.23 minerule::CatalogueInfo Class Reference . . . 139
6.23.1 Detailed Description . . . 140
6.23.2 Member Function Documentation . . . 140
6.23.2.1 updateQrySize() . . . 140
6.23.3 Field Documentation . . . 140
6.23.3.1 qryName . . . 141
6.23.3.2 qryText . . . 141
6.23.3.3 resName . . . 141
6.23.3.4 resSize . . . 141
6.23.3.5 resTables . . . 141
6.24 minerule::CatalogueInstaller Class Reference . . . 142
6.24.1 Detailed Description . . . 143
6.24.2 Member Enumeration Documentation . . . 143
6.24.2.1 SupportedDbms . . . 143
6.24.3 Constructor & Destructor Documentation . . . 143
6.24.3.1 CatalogueInstaller() . . . 143
6.24.3.2 ~CatalogueInstaller() . . . 143
6.24.4 Member Function Documentation . . . 143
6.24.4.1 dropMRAttList() . . . 144
6.24.4.2 dropMRAutoincrement() . . . 144
6.24.4.3 dropMRDepFun() . . . 144
6.24.4.4 dropMRDepFunCol() . . . 144
6.24.4.5 dropMREqKeys() . . . 144
6.24.4.6 dropMREqKeysCol() . . . 144
6.24.4.7 dropMRQuery() . . . 145
6.24.4.8 initializeAutoincrement() . . . 145
6.24.4.9 install() . . . 145
6.24.4.10 installMRAttList() . . . 145
6.24.4.11 installMRAutoincrement() . . . 145
6.24.4.12 installMRDepFun() . . . 146
6.24.4.13 installMRDepFunCol() . . . 146
6.24.4.14 installMREqKeys() . . . 146

6.24.4.15 installMREqKeysCol() . . . 146
6.24.4.16 installMRQuery() . . . 146
6.24.4.17 newInstaller() [1/2] . . . 146
6.24.4.18 newInstaller() [2/2] . . . 147
6.24.4.19 uninstall() . . . 147
6.24.5 Field Documentation . . . 147
6.24.5.1 _connection . . . 147
6.24.5.2 _statement . . . 147
6.25 minerule::CCSMiner Class Reference . . . 148
6.25.1 Detailed Description . . . 148
6.25.2 Constructor & Destructor Documentation . . . 148
6.25.2.1 CCSMiner() . . . 149
6.25.2.2 ~CCSMiner() . . . 149
6.25.3 Member Function Documentation . . . 149
6.25.3.1 algorithmForType() . . . 149
6.25.3.2 canHandleMinerule() . . . 149
6.25.3.3 combina() . . . 150
6.25.3.4 execute() . . . 153
6.25.3.5 find() . . . 153
6.25.3.6 initialize() . . . 153
6.25.3.7 mineRules() . . . 154
6.25.3.8 optimizedMinerule() . . . 156
6.25.3.9 sourceTableRequirements() . . . 156
6.25.4 Field Documentation . . . 156
6.25.4.1 minerule . . . 157
6.26 minerule::CCSMSequence Class Reference . . . 157
6.26.1 Detailed Description . . . 158
6.26.2 Member Typedef Documentation . . . 158
6.26.2.1 Eid_List . . . 158
6.26.2.2 ResultItems . . . 158
6.26.3 Constructor & Destructor Documentation . . . 158
6.26.3.1 CCSMSequence() [1/3] . . . 158
6.26.3.2 CCSMSequence() [2/3] . . . 159
6.26.3.3 CCSMSequence() [3/3] . . . 159
6.26.3.4 ~CCSMSequence() . . . 159
6.26.4 Member Function Documentation . . . 160
6.26.4.1 add() . . . 160
6.26.4.2 addPrefixSequence() . . . 160
6.26.4.3 addSuffixSequence() . . . 160
6.26.4.4 bitMaptoString() . . . 160
6.26.4.5 canGenerate() . . . 161
6.26.4.6 getCount() . . . 161

6.26.4.7 getFirstItem() . . . 161
6.26.4.8 getLastItem() . . . 161
6.26.4.9 getPrefixSequence() . . . 161
6.26.4.10 getSequenceItems() . . . 162
6.26.4.11 getSuffixSequence() . . . 162
6.26.4.12 isSingleton() . . . 162
6.26.4.13 merge() . . . 162
6.26.4.14 operator<() . . . 163
6.26.4.15 operator==() . . . 164
6.26.4.16 read() . . . 164
6.26.4.17 reduce_tr() . . . 164
6.26.4.18 reduction() . . . 165
6.26.4.19 setEid() . . . 166
6.26.4.20 setLastItem() . . . 166
6.26.4.21 setPresent() . . . 166
6.26.4.22 size() . . . 166
6.26.4.23 stampaEid() . . . 167
6.26.4.24 svuota() . . . 167
6.26.4.25 toStdString() . . . 167
6.27 Optimizations::Combinator Class Reference . . . 167
6.27.1 Detailed Description . . . 168
6.27.2 Constructor & Destructor Documentation . . . 168
6.27.2.1 Combinator() . . . 168
6.27.2.2 ∼Combinator() . . . 168
6.27.3 Member Function Documentation . . . 168
6.27.3.1 className() . . . 169
6.27.3.2 getMaxDisjuncts() . . . 169
6.27.3.3 getMaxDistinctPredicates() . . . 169
6.27.3.4 getMaxQueries() . . . 169
6.27.3.5 getTimeOutThreshold() . . . 169
6.27.3.6 setMaxDisjuncts() . . . 169
6.27.3.7 setMaxDistinctPredicates() . . . 170
6.27.3.8 setMaxQueries() . . . 170
6.27.3.9 setOption() . . . 170
6.27.3.10 setTimeOutThreshold() . . . 170
6.28 minerule::Connection Class Reference . . . 170
6.28.1 Detailed Description . . . 171
6.28.2 Member Enumeration Documentation . . . 171
6.28.2.1 TableKind . . . 171
6.28.3 Constructor & Destructor Documentation . . . 172
6.28.3.1 Connection() . . . 172
6.28.3.2 ∼Connection() . . . 172

6.28.4 Member Function Documentation . . . 172
6.28.4.1 create_tmp_db() . . . 172
6.28.4.2 createResultTables() [1/2] . . . 173
6.28.4.3 createResultTables() [2/2] . . . 173
6.28.4.4 delete_tmp_db() . . . 174
6.28.4.5 deleteDestTables() . . . 174
6.28.4.6 deleteTable() . . . 175
6.28.4.7 finalize() [1/2] . . . 175
6.28.4.8 finalize() [2/2] . . . 175
6.28.4.9 getMRDBConnection() . . . 175
6.28.4.10 getTableName() . . . 176
6.28.4.11 init() . . . 176
6.28.4.12 insert() [1/3] . . . 176
6.28.4.13 insert() [2/3] . . . 177
6.28.4.14 insert() [3/3] . . . 177
6.28.4.15 setBodyCardinalities() . . . 177
6.28.4.16 setHeadCardinalities() . . . 177
6.28.4.17 setOutTableName() . . . 178
6.28.4.18 tableExists() . . . 178
6.28.4.19 useMRDBConnection() . . . 178
6.29 mrdb::Connection Class Reference . . . 178
6.29.1 Detailed Description . . . 179
6.29.2 Constructor & Destructor Documentation . . . 179
6.29.2.1 ∼Connection() . . . 179
6.29.3 Member Function Documentation . . . 179
6.29.3.1 createStatement() . . . 179
6.29.3.2 getMetaData() . . . 179
6.29.3.3 prepareStatement() . . . 180
6.30 minerule::Constraints Struct Reference . . . 180
6.30.1 Detailed Description . . . 180
6.30.2 Constructor & Destructor Documentation . . . 180
6.30.2.1 Constraints() . . . 181
6.30.3 Member Function Documentation . . . 181
6.30.3.1 clear() . . . 181
6.30.4 Field Documentation . . . 181
6.30.4.1 bem . . . 181
6.30.4.2 mid_counters . . . 181
6.31 minerule::ConstrItemSetsExtraction Class Reference . . . 182
6.31.1 Detailed Description . . . 182
6.31.2 Constructor & Destructor Documentation . . . 182
6.31.2.1 ConstrItemSetsExtraction() . . . 182
6.31.2.2 ∼ConstrItemSetsExtraction() . . . 183

6.31.3 Member Function Documentation . . . 183
6.31.3.1 algorithmForType() . . . 183
6.31.3.2 canHandleMinerule() . . . 183
6.31.3.3 execute() . . . 183
6.31.3.4 initialize() . . . 184
6.31.3.5 mineRules() . . . 184
6.31.3.6 optimizedMinerule() . . . 185
6.31.3.7 sourceTableRequirements() . . . 185
6.31.4 Field Documentation . . . 185
6.31.4.1 minerule . . . 185
6.32 minerule::ConstrTree Class Reference . . . 186
6.32.1 Detailed Description . . . 187
6.32.2 Constructor & Destructor Documentation . . . 187
6.32.2.1 ConstrTree() . . . 187
6.32.2.2 ∼ConstrTree() . . . 187
6.32.3 Member Function Documentation . . . 187
6.32.3.1 adjustSupp() . . . 188
6.32.3.2 adjustSuppMIndex() . . . 188
6.32.3.3 adjustSuppRSet() . . . 188
6.32.3.4 buildAttrStr() . . . 189
6.32.3.5 buildQry() . . . 189
6.32.3.6 execute() . . . 190
6.32.3.7 getRoot() . . . 190
6.32.3.8 insertRulesInStructure() . . . 190
6.32.3.9 newIncrementalAlgorithm() . . . 191
6.32.3.10 prepareData() . . . 191
6.32.4 Field Documentation . . . 192
6.32.4.1 bodyDes . . . 192
6.32.4.2 headDes . . . 192
6.32.4.3 minerule . . . 192
6.32.4.4 ngroups . . . 192
6.32.4.5 rb2 . . . 193
6.32.4.6 rh2 . . . 193
6.32.4.7 root . . . 193
6.32.4.8 stateb2 . . . 193
6.32.4.9 stateh2 . . . 193
6.33 minerule::Converter Class Reference . . . 193
6.33.1 Detailed Description . . . 194
6.33.2 Constructor & Destructor Documentation . . . 194
6.33.2.1 Converter() [1/8] . . . 194
6.33.2.2 Converter() [2/8] . . . 194
6.33.2.3 Converter() [3/8] . . . 195

6.33.2.4 Converter() [4/8] . . . 195
6.33.2.5 Converter() [5/8] . . . 195
6.33.2.6 Converter() [6/8] . . . 195
6.33.2.7 Converter() [7/8] . . . 196
6.33.2.8 Converter() [8/8] . . . 196
6.33.3 Member Function Documentation . . . 196
6.33.3.1 isNumber() . . . 196
6.33.3.2 toBool() . . . 196
6.33.3.3 toDouble() . . . 197
6.33.3.4 toLong() . . . 197
6.33.3.5 toString() . . . 198
6.34 minerule::CountingIterator Class Reference . . . 198
6.34.1 Detailed Description . . . 198
6.34.2 Constructor & Destructor Documentation . . . 199
6.34.2.1 CountingIterator() [1/2] . . . 199
6.34.2.2 CountingIterator() [2/2] . . . 199
6.34.3 Member Function Documentation . . . 199
6.34.3.1 delete_list_AND_node() . . . 199
6.34.3.2 new_list_AND_node() . . . 200
6.34.3.3 operator∗() . . . 200
6.34.3.4 operator++() [1/2] . . . 200
6.34.3.5 operator++() [2/2] . . . 200
6.34.3.6 operator=() . . . 201
6.35 mrdb::DatabaseMetaData Class Reference . . . 201
6.35.1 Detailed Description . . . 201
6.35.2 Constructor & Destructor Documentation . . . 201
6.35.2.1 ∼DatabaseMetaData() . . . 201
6.35.3 Member Function Documentation . . . 201
6.35.3.1 getColumns() . . . 202
6.35.3.2 getColumnType() . . . 202
6.35.3.3 getTables() . . . 202
6.36 minerule::OptimizerCatalogue::DepFunCatalogue Class Reference . . . 203
6.36.1 Detailed Description . . . 203
6.36.2 Constructor & Destructor Documentation . . . 203
6.36.2.1 DepFunCatalogue() [1/2] . . . 204
6.36.2.2 DepFunCatalogue() [2/2] . . . 204
6.36.2.3 ∼DepFunCatalogue() . . . 204
6.36.3 Member Function Documentation . . . 204
6.36.3.1 getSchemaInfo() . . . 204
6.36.3.2 initialize() . . . 205
6.36.3.3 insertMapping() . . . 205
6.36.3.4 stringToOrder() . . . 206

6.36.4 Field Documentation . . . 206
6.36.4.1 elements . . . 206
6.36.4.2 keys . . . 207
6.37 minerule::DestrTree Class Reference . . . 207
6.37.1 Detailed Description . . . 208
6.37.2 Constructor & Destructor Documentation . . . 208
6.37.2.1 DestrTree() . . . 208
6.37.2.2 ~DestrTree() . . . 208
6.37.3 Member Function Documentation . . . 209
6.37.3.1 adjustSupp() . . . 209
6.37.3.2 adjustSuppMIndex() . . . 209
6.37.3.3 adjustSuppRSet() . . . 209
6.37.3.4 buildAttrStr() . . . 210
6.37.3.5 buildQry() . . . 211
6.37.3.6 buildQry1NotQry2() . . . 211
6.37.3.7 execute() . . . 211
6.37.3.8 getRoot() . . . 212
6.37.3.9 insertRulesInStructure() . . . 212
6.37.3.10 newIncrementalAlgorithm() . . . 212
6.37.3.11 prepareData() . . . 213
6.37.4 Field Documentation . . . 214
6.37.4.1 bodyDes . . . 214
6.37.4.2 headDes . . . 214
6.37.4.3 minerule . . . 214
6.37.4.4 ngroups . . . 214
6.37.4.5 rb1 . . . 214
6.37.4.6 rb1nb2 . . . 215
6.37.4.7 rh1 . . . 215
6.37.4.8 rh1nh2 . . . 215
6.37.4.9 root . . . 215
6.37.4.10 stateb1 . . . 215
6.37.4.11 stateb1nb2 . . . 215
6.37.4.12 stateh1 . . . 216
6.37.4.13 stateh1nh2 . . . 216
6.38 minerule::Dist_cond Class Reference . . . 216
6.38.1 Detailed Description . . . 216
6.38.2 Constructor & Destructor Documentation . . . 216
6.38.2.1 Dist_cond() [1/2] . . . 217
6.38.2.2 Dist_cond() [2/2] . . . 217
6.38.3 Member Function Documentation . . . 217
6.38.3.1 copyDistCond() . . . 217
6.38.4 Field Documentation . . . 217

6.38.4.1 attr . . . 217
6.38.4.2 function . . . 218
6.38.4.3 range . . . 218
6.39 minerule::QueryNormalizer::SubstEntryHead::Elem Class Reference . . . 218
6.39.1 Detailed Description . . . 218
6.39.2 Constructor & Destructor Documentation . . . 218
6.39.2.1 Elem() [1/2] . . . 219
6.39.2.2 Elem() [2/2] . . . 219
6.39.3 Field Documentation . . . 219
6.39.3.1 colName . . . 219
6.39.3.2 op . . . 219
6.39.3.3 order . . . 219
6.39.3.4 value . . . 220
6.40 minerule::EncodedNF Class Reference . . . 220
6.40.1 Detailed Description . . . 220
6.40.2 Member Enumeration Documentation . . . 220
6.40.2.1 CodesRelationship . . . 220
6.40.3 Member Function Documentation . . . 221
6.40.3.1 computeHammingDistance() . . . 221
6.40.3.2 getCodesRelationship() . . . 222
6.40.3.3 operator==() . . . 223
6.40.4 Field Documentation . . . 223
6.40.4.1 codingLen . . . 223
6.40.4.2 encVector . . . 223
6.40.4.3 numVars . . . 223
6.40.4.4 vecSize . . . 224
6.41 minerule::EncodedNFIterator Class Reference . . . 224
6.41.1 Detailed Description . . . 224
6.41.2 Constructor & Destructor Documentation . . . 224
6.41.2.1 EncodedNFIterator() . . . 224
6.41.3 Member Function Documentation . . . 224
6.41.3.1 ok() . . . 225
6.41.3.2 operator*() . . . 225
6.41.3.3 operator++() . . . 225
6.41.3.4 setOk() . . . 225
6.42 minerule::OptimizerCatalogue::EqKeysCatalogue Class Reference . . . 226
6.42.1 Detailed Description . . . 226
6.42.2 Constructor & Destructor Documentation . . . 226
6.42.2.1 EqKeysCatalogue() [1/2] . . . 227
6.42.2.2 EqKeysCatalogue() [2/2] . . . 227
6.42.2.3 ~EqKeysCatalogue() . . . 227
6.42.3 Member Function Documentation . . . 227

6.42.3.1 getSchemaInfo() . . . 227
6.42.3.2 initialize() . . . 228
6.42.3.3 insertMapping() . . . 228
6.42.3.4 stringToOrder() . . . 229
6.42.4 Field Documentation . . . 229
6.42.4.1 elements . . . 229
6.42.4.2 keys . . . 230
6.43 minerule::EvaluationMeasure Class Reference . . . 230
6.43.1 Detailed Description . . . 230
6.43.2 Member Enumeration Documentation . . . 230
6.43.2.1 MeasureType . . . 230
6.43.2.2 RelOperator . . . 231
6.43.3 Member Function Documentation . . . 231
6.43.3.1 getMeasureType() . . . 231
6.44 mrc::Exception Class Reference . . . 232
6.44.1 Detailed Description . . . 232
6.44.2 Constructor & Destructor Documentation . . . 232
6.44.2.1 Exception() . . . 232
6.44.2.2 ∼Exception() . . . 232
6.44.3 Member Function Documentation . . . 232
6.44.3.1 getResultID() . . . 233
6.44.3.2 what() . . . 233
6.45 minerule::ExpressionNFCoder Class Reference . . . 233
6.45.1 Detailed Description . . . 233
6.45.2 Constructor & Destructor Documentation . . . 234
6.45.2.1 ExpressionNFCoder() . . . 234
6.45.2.2 ∼ExpressionNFCoder() . . . 234
6.45.3 Member Function Documentation . . . 234
6.45.3.1 encode() . . . 234
6.45.4 Field Documentation . . . 235
6.45.4.1 bitsPerCell . . . 235
6.45.4.2 filter . . . 235
6.46 minerule::QueryResult::FastSorter Class Reference . . . 236
6.46.1 Detailed Description . . . 236
6.46.2 Member Function Documentation . . . 236
6.46.2.1 operator()() . . . 236
6.47 minerule::RuleFormatter::FieldWidths Class Reference . . . 236
6.47.1 Detailed Description . . . 237
6.47.2 Constructor & Destructor Documentation . . . 237
6.47.2.1 FieldWidths() . . . 237
6.47.3 Field Documentation . . . 237
6.47.3.1 body . . . 237

6.47.3.2 conf . . . 237
6.47.3.3 head . . . 237
6.47.3.4 supp . . . 238
6.48 minerule::FileUtils Class Reference . . . 238
6.48.1 Detailed Description . . . 238
6.48.2 Member Function Documentation . . . 238
6.48.2.1 fileExists() . . . 238
6.49 RulesMiningAlgorithms::FPGrowth Class Reference . . . 239
6.49.1 Detailed Description . . . 239
6.49.2 Member Enumeration Documentation . . . 239
6.49.2.1 FPAlgoType . . . 239
6.49.3 Constructor & Destructor Documentation . . . 240
6.49.3.1 ∼FPGrowth() . . . 240
6.49.4 Member Function Documentation . . . 240
6.49.4.1 className() . . . 240
6.49.4.2 getAlgoType() . . . 240
6.49.4.3 setAlgoType() . . . 240
6.49.4.4 setOption() . . . 241
6.50 minerule::FSMiner Class Reference . . . 241
6.50.1 Detailed Description . . . 241
6.50.2 Constructor & Destructor Documentation . . . 242
6.50.2.1 FSMiner() . . . 242
6.50.2.2 ∼FSMiner() . . . 242
6.50.3 Member Function Documentation . . . 242
6.50.3.1 algorithmForType() . . . 242
6.50.3.2 canHandleMinerule() . . . 243
6.50.3.3 execute() . . . 243
6.50.3.4 initialize() . . . 243
6.50.3.5 mineRules() . . . 244
6.50.3.6 optimizedMinerule() . . . 244
6.50.3.7 sourceTableRequirements() . . . 245
6.50.4 Field Documentation . . . 245
6.50.4.1 minerule . . . 245
6.51 minerule::FSTree Class Reference . . . 245
6.51.1 Detailed Description . . . 246
6.51.2 Constructor & Destructor Documentation . . . 246
6.51.2.1 FSTree() . . . 246
6.51.2.2 ∼FSTree() . . . 246
6.51.3 Member Function Documentation . . . 247
6.51.3.1 addList() . . . 247
6.51.3.2 addResult() . . . 247
6.51.3.3 addResults() . . . 248

6.51.3.4 appendChild() . . . 248
6.51.3.5 construct_Tree() . . . 249
6.51.3.6 countPath() . . . 250
6.51.3.7 createLinkNSTable() . . . 250
6.51.3.8 createLinkSTable() . . . 251
6.51.3.9 fraziona() . . . 251
6.51.3.10 getN_nodi() . . . 252
6.51.3.11 getResult() . . . 253
6.51.3.12 getThreshold() . . . 253
6.51.3.13 insertLink() . . . 253
6.51.3.14 insertTree() . . . 254
6.51.3.15 mine() . . . 254
6.51.3.16 resetN_nodi() . . . 255
6.51.3.17 setThreshold() . . . 255
6.51.3.18 stampa() . . . 255
6.52 minerule::FSTreeNode Class Reference . . . 256
6.52.1 Detailed Description . . . 256
6.52.2 Constructor & Destructor Documentation . . . 256
6.52.2.1 FSTreeNode() [1/3] . . . 256
6.52.2.2 FSTreeNode() [2/3] . . . 257
6.52.2.3 FSTreeNode() [3/3] . . . 257
6.52.2.4 ~FSTreeNode() . . . 257
6.52.3 Member Function Documentation . . . 258
6.52.3.1 getChild() . . . 258
6.52.3.2 getCount() . . . 258
6.52.3.3 getLabel() . . . 258
6.52.3.4 getParent() . . . 259
6.52.3.5 insertChild() . . . 259
6.52.3.6 setCount() . . . 259
6.52.3.7 setParent() . . . 259
6.53 minerule::FSTreeSequence Class Reference . . . 260
6.53.1 Detailed Description . . . 260
6.53.2 Member Typedef Documentation . . . 261
6.53.2.1 ItemVector . . . 261
6.53.3 Constructor & Destructor Documentation . . . 261
6.53.3.1 FSTreeSequence() [1/3] . . . 261
6.53.3.2 FSTreeSequence() [2/3] . . . 261
6.53.3.3 FSTreeSequence() [3/3] . . . 261
6.53.3.4 ~FSTreeSequence() . . . 262
6.53.4 Member Function Documentation . . . 262
6.53.4.1 add() . . . 262
6.53.4.2 getFSTreeSequence() . . . 263

6.53.4.3 insertHead() . . . 263
6.53.4.4 operator<() . . . 263
6.53.4.5 operator==() . . . 264
6.53.4.6 read() . . . 264
6.53.4.7 removeHead() . . . 265
6.53.4.8 size() . . . 265
6.53.4.9 stampa() . . . 266
6.53.4.10 svuota() . . . 266
6.53.4.11 toStdString() . . . 266
6.54 minerule::GAQueryCombinator Class Reference . . . 267
6.54.1 Detailed Description . . . 267
6.54.2 Member Typedef Documentation . . . 267
6.54.2.1 QueryAndList . . . 268
6.54.2.2 QueryOrList . . . 268
6.54.3 Constructor & Destructor Documentation . . . 268
6.54.3.1 GAQueryCombinator() . . . 268
6.54.3.2 ~GAQueryCombinator() . . . 268
6.54.4 Member Function Documentation . . . 268
6.54.4.1 evaluator() . . . 268
6.54.4.2 evolve() . . . 269
6.54.4.3 getMaxDisjuncts() . . . 270
6.54.4.4 getMaxDistinctPredicates() . . . 270
6.54.4.5 getMaxQueries() . . . 270
6.54.4.6 getResult() . . . 270
6.54.4.7 getTimeOutThreshold() . . . 271
6.54.4.8 setMaxDisjuncts() . . . 271
6.54.4.9 setMaxDistinctPredicates() . . . 271
6.54.4.10 setMaxQueries() . . . 271
6.54.4.11 setTimeOutThreshold() . . . 271
6.54.4.12 TerminationCriterion() . . . 272
6.55 minerule::GenericSourceRowAttribute Class Reference . . . 272
6.55.1 Detailed Description . . . 273
6.55.2 Member Typedef Documentation . . . 273
6.55.2.1 ElementType . . . 273
6.55.3 Constructor & Destructor Documentation . . . 274
6.55.3.1 GenericSourceRowAttribute() [1/3] . . . 274
6.55.3.2 GenericSourceRowAttribute() [2/3] . . . 274
6.55.3.3 GenericSourceRowAttribute() [3/3] . . . 274
6.55.3.4 ~GenericSourceRowAttribute() . . . 274
6.55.4 Member Function Documentation . . . 274
6.55.4.1 asString() . . . 274
6.55.4.2 compareTo() . . . 275

6.55.4.3 copy() . . . 275
6.55.4.4 createAttribute() . . . 276
6.55.4.5 createElement() . . . 276
6.55.4.6 createElementFromType() . . . 277
6.55.4.7 deserialize() . . . 277
6.55.4.8 deserializeElementFromResultSet() . . . 278
6.55.4.9 deserializeElementFromString() . . . 278
6.55.4.10 empty() . . . 279
6.55.4.11 getElementType() . . . 279
6.55.4.12 getFullElementType() . . . 279
6.55.4.13 getSQLData() . . . 279
6.55.4.14 getType() . . . 280
6.55.4.15 operator"!=() . . . 280
6.55.4.16 operator()() . . . 280
6.55.4.17 operator<() . . . 280
6.55.4.18 operator<<() . . . 280
6.55.4.19 operator=() . . . 281
6.55.4.20 operator==() . . . 281
6.55.4.21 serialize() . . . 282
6.55.4.22 serializeElementToString() . . . 282
6.55.4.23 setPreparedStatementParameters() . . . 282
6.55.4.24 setValue() [1/2] . . . 282
6.55.4.25 setValue() [2/2] . . . 283
6.55.5 Friends And Related Function Documentation . . . 283
6.55.5.1 operator<< . . . 283
6.56 mrmatch::GidRulesMatcher Class Reference . . . 284
6.56.1 Detailed Description . . . 284
6.56.2 Constructor & Destructor Documentation . . . 284
6.56.2.1 GidRulesMatcher() . . . 284
6.56.2.2 ~GidRulesMatcher() . . . 285
6.56.3 Member Function Documentation . . . 285
6.56.3.1 addRule() . . . 285
6.56.3.2 match() . . . 285
6.56.3.3 matchItemTransaction() . . . 285
6.56.3.4 matchRuleTransaction() . . . 286
6.56.3.5 matchWithCrossProduct() . . . 286
6.56.3.6 matchWithoutCrossProduct() . . . 286
6.56.3.7 newMatcher() . . . 287
6.56.3.8 printMatches() . . . 287
6.57 minerule::Head Class Reference . . . 288
6.57.1 Detailed Description . . . 288
6.57.2 Member Typedef Documentation . . . 288

6.57.2.1 RowContainer . . . 288
6.57.3 Constructor & Destructor Documentation . . . 289
6.57.3.1 Head() . . . 289
6.57.3.2 ∼Head() . . . 289
6.57.4 Member Function Documentation . . . 289
6.57.4.1 extractRules() . . . 289
6.57.4.2 findHead() . . . 290
6.57.4.3 findHead2() . . . 291
6.57.4.4 findHead2bis() . . . 291
6.57.4.5 getAncestor() . . . 292
6.57.4.6 insertItemSetH() [1/2] . . . 292
6.57.4.7 insertItemSetH() [2/2] . . . 292
6.57.4.8 setAncestor() . . . 293
6.57.5 Field Documentation . . . 293
6.57.5.1 counth . . . 293
6.57.5.2 NR . . . 293
6.58 minerule::HeadBodyPredicatesSeparator Class Reference . . . 293
6.58.1 Detailed Description . . . 293
6.58.2 Member Function Documentation . . . 294
6.58.2.1 separate() . . . 294
6.59 minerule::HeaderQuery Class Reference . . . 294
6.59.1 Detailed Description . . . 295
6.59.2 Constructor & Destructor Documentation . . . 295
6.59.2.1 HeaderQuery() [1/2] . . . 295
6.59.2.2 HeaderQuery() [2/2] . . . 295
6.59.3 Member Function Documentation . . . 295
6.59.3.1 getAttrBody() . . . 296
6.59.3.2 getAttrHead() . . . 296
6.59.3.3 getCardBody() . . . 296
6.59.3.4 getCardHead() . . . 296
6.59.3.5 getName() . . . 296
6.60 minerule::IDIncrementalAlgorithm Class Reference . . . 297
6.60.1 Detailed Description . . . 297
6.60.2 Constructor & Destructor Documentation . . . 297
6.60.2.1 IDIncrementalAlgorithm() . . . 298
6.60.2.2 ∼IDIncrementalAlgorithm() . . . 298
6.60.3 Member Function Documentation . . . 298
6.60.3.1 checkInclusion() . . . 298
6.60.3.2 checkInValidRules() . . . 299
6.60.3.3 execute() . . . 299
6.60.3.4 fillValidItems() . . . 300
6.60.3.5 filterQueries() . . . 300

6.60.3.6 getItemInfos() . . . 301
6.60.3.7 newIncrementalAlgorithm() . . . 301
6.60.4 Field Documentation . . . 302
6.60.4.1 attrList . . . 302
6.60.4.2 minerule . . . 302
6.61 minerule::IncrementalAlgorithm Class Reference . . . 302
6.61.1 Detailed Description . . . 303
6.61.2 Constructor & Destructor Documentation . . . 303
6.61.2.1 IncrementalAlgorithm() . . . 303
6.61.2.2 ∼IncrementalAlgorithm() . . . 303
6.61.3 Member Function Documentation . . . 303
6.61.3.1 execute() . . . 303
6.61.3.2 newIncrementalAlgorithm() . . . 304
6.61.4 Field Documentation . . . 304
6.61.4.1 minerule . . . 304
6.62 minerule::Interval Class Reference . . . 304
6.62.1 Detailed Description . . . 306
6.62.2 Member Enumeration Documentation . . . 306
6.62.2.1 Operator . . . 306
6.62.3 Constructor & Destructor Documentation . . . 306
6.62.3.1 Interval() [1/3] . . . 306
6.62.3.2 Interval() [2/3] . . . 307
6.62.3.3 Interval() [3/3] . . . 307
6.62.4 Member Function Documentation . . . 307
6.62.4.1 compareValues() . . . 308
6.62.4.2 getAttribute() . . . 308
6.62.4.3 getMaxLwr() . . . 309
6.62.4.4 getMinUpp() . . . 309
6.62.4.5 getOperator() . . . 310
6.62.4.6 getValue() . . . 310
6.62.4.7 intersect() . . . 311
6.62.4.8 isEmpty() . . . 311
6.62.4.9 setType() . . . 312
6.62.4.10 update() . . . 312
6.62.5 Friends And Related Function Documentation . . . 312
6.62.5.1 operator<< . . . 312
6.62.6 Field Documentation . . . 312
6.62.6.1 lwr . . . 313
6.62.6.2 lwrOpen . . . 313
6.62.6.3 NEGINFTY . . . 313
6.62.6.4 openPoints . . . 313
6.62.6.5 POSINFTY . . . 313

6.62.6.6 type . . . 314
6.62.6.7 typeOk . . . 314
6.62.6.8 upp . . . 314
6.62.6.9 uppOpen . . . 314
6.63 minerule::IntervalChecker Class Reference . . . 314
6.63.1 Detailed Description . . . 315
6.63.2 Constructor & Destructor Documentation . . . 315
6.63.2.1 IntervalChecker() . . . 315
6.63.3 Member Function Documentation . . . 315
6.63.3.1 impossibleVariableSetting() . . . 315
6.63.3.2 typeForAttribute() . . . 316
6.64 minerule::InvalidConfigurationFilter Class Reference . . . 316
6.64.1 Detailed Description . . . 317
6.64.2 Constructor & Destructor Documentation . . . 317
6.64.2.1 InvalidConfigurationFilter() . . . 317
6.64.3 Member Function Documentation . . . 317
6.64.3.1 operator()() . . . 317
6.65 ItemsetsMiningAlgorithms Class Reference . . . 318
6.65.1 Detailed Description . . . 318
6.65.2 Constructor & Destructor Documentation . . . 318
6.65.2.1 ItemsetsMiningAlgorithms() [1/2] . . . 318
6.65.2.2 ItemsetsMiningAlgorithms() [2/2] . . . 318
6.65.2.3 ∼ItemsetsMiningAlgorithms() . . . 319
6.65.3 Member Function Documentation . . . 319
6.65.3.1 className() . . . 319
6.65.3.2 getPreferredAlgorithm() . . . 319
6.65.3.3 setOption() . . . 319
6.65.3.4 setPreferredAlgorithm() . . . 319
6.65.3.5 subclassForName() . . . 320
6.66 minerule::ItemTransaction< ItemSetType > Class Template Reference . . . 320
6.66.1 Detailed Description . . . 320
6.66.2 Constructor & Destructor Documentation . . . 321
6.66.2.1 ItemTransaction() . . . 321
6.66.3 Member Function Documentation . . . 321
6.66.3.1 findGid() . . . 321
6.66.3.2 loadBody() . . . 321
6.66.3.3 loadHead() . . . 322
6.67 minerule::ItemType Class Reference . . . 322
6.67.1 Detailed Description . . . 323
6.67.2 Constructor & Destructor Documentation . . . 323
6.67.2.1 ItemType() [1/4] . . . 323
6.67.2.2 ∼ItemType() . . . 323

6.67.2.3 ItemType() [2/4] . . . 323
6.67.2.4 ItemType() [3/4] . . . 324
6.67.2.5 ItemType() [4/4] . . . 324
6.67.3 Member Function Documentation . . . 324
6.67.3.1 asString() . . . 324
6.67.3.2 getElement() . . . 324
6.67.3.3 getFullElementType() . . . 325
6.67.3.4 getSQLData() . . . 325
6.67.3.5 lessThan() . . . 325
6.67.3.6 operator"!=() [1/2] . . . 325
6.67.3.7 operator"!=() [2/2] . . . 326
6.67.3.8 operator()() . . . 326
6.67.3.9 operator<() [1/2] . . . 326
6.67.3.10 operator<() [2/2] . . . 326
6.67.3.11 operator=() [1/2] . . . 327
6.67.3.12 operator=() [2/2] . . . 327
6.67.3.13 operator==() [1/2] . . . 327
6.67.3.14 operator==() [2/2] . . . 327
6.67.3.15 operator>() [1/2] . . . 328
6.67.3.16 operator>() [2/2] . . . 328
6.67.3.17 setPreparedStatementParameters() . . . 328
6.67.3.18 setSourceRowElement() . . . 328
6.67.4 Friends And Related Function Documentation . . . 329
6.67.4.1 operator<< . . . 329
6.68 minerule::QueryResult::Iterator Class Reference . . . 329
6.68.1 Detailed Description . . . 329
6.68.2 Constructor & Destructor Documentation . . . 329
6.68.2.1 Iterator() . . . 329
6.68.2.2 ~Iterator() . . . 330
6.68.3 Member Function Documentation . . . 330
6.68.3.1 getRule() . . . 330
6.68.3.2 init() . . . 330
6.68.3.3 next() . . . 331
6.69 minerule::SourceTable::Iterator Class Reference . . . 331
6.69.1 Detailed Description . . . 332
6.69.2 Constructor & Destructor Documentation . . . 332
6.69.2.1 Iterator() [1/2] . . . 332
6.69.2.2 Iterator() [2/2] . . . 332
6.69.2.3 ~Iterator() . . . 332
6.69.3 Member Function Documentation . . . 332
6.69.3.1 get() . . . 332
6.69.3.2 isAfterLast() . . . 333

6.69.3.3 next() . . . 333
6.69.3.4 operator++() . . . 333
6.69.3.5 operator->() . . . 333
6.69.4 Friends And Related Function Documentation . . . 333
6.69.4.1 SourceTable . . . 334
6.70 minerule::FSTreeSequence::less_sequence Struct Reference . . . 334
6.70.1 Detailed Description . . . 334
6.70.2 Member Function Documentation . . . 334
6.70.2.1 comp() . . . 334
6.70.2.2 operator()() . . . 335
6.71 mrc::Options::ListFormat Class Reference . . . 335
6.71.1 Detailed Description . . . 335
6.71.2 Constructor & Destructor Documentation . . . 335
6.71.2.1 ListFormat() . . . 335
6.71.3 Field Documentation . . . 335
6.71.3.1 result . . . 336
6.71.3.2 size . . . 336
6.71.3.3 text . . . 336
6.72 minerule::MapElement Class Reference . . . 336
6.72.1 Detailed Description . . . 337
6.72.2 Constructor & Destructor Documentation . . . 337
6.72.2.1 MapElement() . . . 338
6.72.3 Member Function Documentation . . . 338
6.72.3.1 clear() [1/2] . . . 338
6.72.3.2 clear() [2/2] . . . 338
6.72.3.3 count() . . . 338
6.72.3.4 gbits() . . . 339
6.72.3.5 getElm() . . . 339
6.72.3.6 getElmA() . . . 339
6.72.3.7 insert() [1/2] . . . 339
6.72.3.8 insert() [2/2] . . . 339
6.72.3.9 invert() [1/2] . . . 340
6.72.3.10 invert() [2/2] . . . 340
6.72.3.11 length() . . . 340
6.72.3.12 moreThan() . . . 340
6.72.3.13 operator"!=() . . . 341
6.72.3.14 operator&() . . . 341
6.72.3.15 operator&=() . . . 341
6.72.3.16 operator=() . . . 341
6.72.3.17 operator==() . . . 342
6.72.3.18 operator[]() . . . 342
6.72.3.19 operator^=() . . . 342

6.72.3.20 operator"|=() . . . 342
6.72.3.21 print() [1/2] . . . 343
6.72.3.22 print() [2/2] . . . 343
6.72.3.23 reset() [1/2] . . . 343
6.72.3.24 reset() [2/2] . . . 343
6.72.3.25 serialize() . . . 344
6.72.3.26 set() [1/2] . . . 344
6.72.3.27 set() [2/2] . . . 344
6.72.3.28 setNBit() . . . 344
6.72.3.29 size() . . . 345
6.72.3.30 ssize() . . . 345
6.72.3.31 test() . . . 345
6.72.3.32 unserialize() . . . 345
6.72.4 Field Documentation . . . 346
6.72.4.1 counter . . . 346
6.72.4.2 intersections . . . 346
6.73 mrmatch::Matcher Class Reference . . . 346
6.73.1 Detailed Description . . . 347
6.73.2 Member Function Documentation . . . 347
6.73.2.1 addRule() . . . 347
6.73.2.2 match() . . . 347
6.73.2.3 matchItemTransaction() . . . 347
6.73.2.4 matchRuleTransaction() . . . 348
6.73.2.5 matchWithCrossProduct() . . . 348
6.73.2.6 matchWithoutCrossProduct() . . . 348
6.73.2.7 newMatcher() . . . 349
6.73.2.8 printMatches() . . . 349
6.74 minerule::MemDebugGenericSourceRowAttribute Class Reference . . . 349
6.74.1 Detailed Description . . . 350
6.74.2 Member Typedef Documentation . . . 350
6.74.2.1 ElementType . . . 351
6.74.3 Constructor & Destructor Documentation . . . 351
6.74.3.1 MemDebugGenericSourceRowAttribute() [1/3] . . . 351
6.74.3.2 MemDebugGenericSourceRowAttribute() [2/3] . . . 351
6.74.3.3 MemDebugGenericSourceRowAttribute() [3/3] . . . 351
6.74.3.4 ~MemDebugGenericSourceRowAttribute() . . . 351
6.74.4 Member Function Documentation . . . 352
6.74.4.1 asString() . . . 352
6.74.4.2 compareTo() . . . 352
6.74.4.3 copy() . . . 353
6.74.4.4 createAttribute() . . . 353
6.74.4.5 createElement() . . . 353

6.74.4.6 createElementFromType() . . . 354
6.74.4.7 deserialize() . . . 354
6.74.4.8 deserializeElementFromResultSet() . . . 355
6.74.4.9 deserializeElementFromString() . . . 355
6.74.4.10 empty() . . . 356
6.74.4.11 getElementType() . . . 356
6.74.4.12 getFullElementType() . . . 356
6.74.4.13 getInstanceCounter() . . . 356
6.74.4.14 getSQLData() . . . 357
6.74.4.15 getType() . . . 357
6.74.4.16 operator"!=() . . . 357
6.74.4.17 operator()() . . . 358
6.74.4.18 operator<() . . . 358
6.74.4.19 operator<<() . . . 358
6.74.4.20 operator==() . . . 359
6.74.4.21 serialize() . . . 359
6.74.4.22 serializeElementToString() . . . 359
6.74.4.23 setPreparedStatementParameters() . . . 359
6.74.4.24 setValue() [1/2] . . . 360
6.74.4.25 setValue() [2/2] . . . 360
6.75 minerule::MineruleException Class Reference . . . 361
6.75.1 Detailed Description . . . 361
6.75.2 Constructor & Destructor Documentation . . . 361
6.75.2.1 MineruleException() . . . 361
6.75.2.2 ~MineruleException() . . . 361
6.75.3 Member Function Documentation . . . 362
6.75.3.1 getErrorCode() . . . 362
6.75.3.2 unformattedMessage() . . . 362
6.75.3.3 what() . . . 362
6.76 MineruleOptions Class Reference . . . 362
6.76.1 Detailed Description . . . 364
6.76.2 Constructor & Destructor Documentation . . . 364
6.76.2.1 ~MineruleOptions() . . . 364
6.76.2.2 MineruleOptions() [1/2] . . . 364
6.76.2.3 MineruleOptions() [2/2] . . . 364
6.76.3 Member Function Documentation . . . 365
6.76.3.1 className() . . . 365
6.76.3.2 getDebugStream() . . . 365
6.76.3.3 getDebugStreamObj() [1/2] . . . 365
6.76.3.4 getDebugStreamObj() [2/2] . . . 365
6.76.3.5 getErrStream() . . . 365
6.76.3.6 getErrStreamObj() [1/2] . . . 366

6.76.3.7 getErrStreamObj() [2/2] . . . 366
6.76.3.8 getKnownStreams() . . . 366
6.76.3.9 getLogStream() . . . 366
6.76.3.10 getLogStreamObj() [1/2] . . . 366
6.76.3.11 getLogStreamObj() [2/2] . . . 367
6.76.3.12 getMineruleName() . . . 367
6.76.3.13 getMineruleSourceName() . . . 367
6.76.3.14 getMiningAlgorithms() [1/2] . . . 367
6.76.3.15 getMiningAlgorithms() [2/2] . . . 367
6.76.3.16 getMRDB() [1/2] . . . 368
6.76.3.17 getMRDB() [2/2] . . . 368
6.76.3.18 getOptimizations() [1/2] . . . 368
6.76.3.19 getOptimizations() [2/2] . . . 368
6.76.3.20 getParsers() [1/2] . . . 368
6.76.3.21 getParsers() [2/2] . . . 369
6.76.3.22 getSafety() [1/2] . . . 369
6.76.3.23 getSafety() [2/2] . . . 369
6.76.3.24 getSharedOptions() . . . 369
6.76.3.25 getUserOptions() . . . 369
6.76.3.26 getWarnStream() . . . 370
6.76.3.27 getWarnStreamObj() [1/2] . . . 370
6.76.3.28 getWarnStreamObj() [2/2] . . . 370
6.76.3.29 readFromFile() . . . 370
6.76.3.30 readFromString() . . . 371
6.76.3.31 saveOptions() . . . 371
6.76.3.32 setMineruleName() . . . 373
6.76.3.33 setMineruleSourceName() . . . 374
6.76.3.34 setOption() . . . 374
6.76.3.35 subclassForName() . . . 374
6.76.4 Field Documentation . . . 375
6.76.4.1 DEFAULT_FILE_NAME . . . 375
6.77 minerule::OptimizerCatalogue::MineruleResultInfo Class Reference . . . 375
6.77.1 Detailed Description . . . 376
6.77.2 Member Typedef Documentation . . . 376
6.77.2.1 AttrVector . . . 376
6.77.2.2 bem_c . . . 376
6.77.3 Constructor & Destructor Documentation . . . 377
6.77.3.1 MineruleResultInfo() [1/2] . . . 377
6.77.3.2 MineruleResultInfo() [2/2] . . . 377
6.77.4 Member Function Documentation . . . 377
6.77.4.1 getBEMText() . . . 377
6.77.4.2 getMineitemsetsText() . . . 378

6.77.4.3 getMineruleText() . . . 378
6.77.4.4 getMinesequenceText() . . . 379
6.77.4.5 getText() . . . 379
6.77.4.6 hasCrossConditions() [1/2] . . . 379
6.77.4.7 hasCrossConditions() [2/2] . . . 380
6.77.4.8 hasDisjuctionsInMC() . . . 380
6.77.4.9 hasSameBodyHead() . . . 380
6.77.4.10 init() . . . 381
6.77.4.11 requiresClusters() . . . 381
6.77.5 Field Documentation . . . 381
6.77.5.1 ba . . . 381
6.77.5.2 body_coincident_head . . . 381
6.77.5.3 bodyCardinalities . . . 381
6.77.5.4 c_aggr_list . . . 382
6.77.5.5 ca . . . 382
6.77.5.6 cc . . . 382
6.77.5.7 conf . . . 382
6.77.5.8 distinct . . . 382
6.77.5.9 filter_condition . . . 382
6.77.5.10 ga . . . 383
6.77.5.11 gc . . . 383
6.77.5.12 ha . . . 383
6.77.5.13 headCardinalities . . . 383
6.77.5.14 length . . . 383
6.77.5.15 mc . . . 383
6.77.5.16 miningTask . . . 384
6.77.5.17 oa . . . 384
6.77.5.18 ra . . . 384
6.77.5.19 resultset . . . 384
6.77.5.20 seq_bem_vect . . . 384
6.77.5.21 seq_dist_vect . . . 384
6.77.5.22 sequenceAllowedGaps . . . 385
6.77.5.23 sup . . . 385
6.77.5.24 tab_result . . . 385
6.77.5.25 tab_source . . . 385
6.77.5.26 tautologies . . . 385
6.78 minerule::MiningAlgorithm Class Reference . . . 385
6.78.1 Detailed Description . . . 386
6.78.2 Constructor & Destructor Documentation . . . 386
6.78.2.1 MiningAlgorithm() . . . 386
6.78.3 Member Function Documentation . . . 386
6.78.3.1 algorithmForType() . . . 387

6.78.3.2 canHandleMinerule() . . . 387
6.78.3.3 execute() . . . 387
6.78.3.4 initialize() . . . 388
6.78.3.5 mineRules() . . . 388
6.78.3.6 optimizedMinerule() . . . 388
6.78.3.7 sourceTableRequirements() . . . 389
6.78.4 Field Documentation . . . 389
6.78.4.1 connection . . . 389
6.78.4.2 minerule . . . 389
6.78.4.3 options . . . 389
6.79 minerule::MiningAlgorithmBase Class Reference . . . 389
6.79.1 Detailed Description . . . 390
6.79.2 Constructor & Destructor Documentation . . . 390
6.79.2.1 MiningAlgorithmBase() . . . 390
6.79.2.2 ∼MiningAlgorithmBase() . . . 390
6.79.3 Member Function Documentation . . . 390
6.79.3.1 algorithmForType() . . . 391
6.79.3.2 canHandleMinerule() . . . 391
6.79.3.3 execute() . . . 391
6.79.3.4 optimizedMinerule() . . . 391
6.79.3.5 sourceTableRequirements() . . . 392
6.79.4 Field Documentation . . . 392
6.79.4.1 minerule . . . 392
6.80 MiningAlgorithms Class Reference . . . 392
6.80.1 Detailed Description . . . 393
6.80.2 Constructor & Destructor Documentation . . . 393
6.80.2.1 ∼MiningAlgorithms() . . . 393
6.80.3 Member Function Documentation . . . 393
6.80.3.1 className() . . . 393
6.80.3.2 getItemsetsMiningAlgorithms() . . . 393
6.80.3.3 getRulesMiningAlgorithms() [1/2] . . . 393
6.80.3.4 getRulesMiningAlgorithms() [2/2] . . . 394
6.80.3.5 getSequencesMiningAlgorithms() . . . 394
6.80.3.6 setOption() . . . 394
6.80.3.7 subclassForName() . . . 394
6.81 minerule::MinMax Class Reference . . . 395
6.81.1 Detailed Description . . . 395
6.81.2 Constructor & Destructor Documentation . . . 395
6.81.2.1 MinMax() [1/2] . . . 395
6.81.2.2 MinMax() [2/2] . . . 395
6.81.3 Member Function Documentation . . . 395
6.81.3.1 maxValue() . . . 396

6.81.3.2 minValue() . . . 396
6.81.4 Field Documentation . . . 396
6.81.4.1 max . . . 396
6.81.4.2 min . . . 396
6.82 minerule::MinMaxPair Class Reference . . . 396
6.82.1 Detailed Description . . . 397
6.82.2 Constructor & Destructor Documentation . . . 397
6.82.2.1 MinMaxPair() . . . 397
6.82.2.2 ∼MinMaxPair() . . . 397
6.82.3 Member Function Documentation . . . 397
6.82.3.1 applyConstraints() . . . 398
6.82.3.2 contains() . . . 398
6.82.3.3 getDefaultMax() . . . 398
6.82.3.4 getMax() . . . 398
6.82.3.5 getMin() . . . 399
6.82.3.6 setDefaultMax() . . . 399
6.82.3.7 setMax() . . . 399
6.82.3.8 setMin() . . . 399
6.82.3.9 validate() . . . 399
6.83 Mrdb Class Reference . . . 400
6.83.1 Detailed Description . . . 400
6.83.2 Constructor & Destructor Documentation . . . 400
6.83.2.1 Mrdb() . . . 400
6.83.2.2 ∼Mrdb() . . . 401
6.83.3 Member Function Documentation . . . 401
6.83.3.1 className() . . . 401
6.83.3.2 clearConnection() . . . 401
6.83.3.3 getCacheWrites() . . . 401
6.83.3.4 getDBMS() . . . 401
6.83.3.5 getLibName() . . . 402
6.83.3.6 getMRDBConnection() . . . 402
6.83.3.7 getName() . . . 402
6.83.3.8 getPassword() . . . 402
6.83.3.9 getUsername() . . . 402
6.83.3.10 resetConnection() . . . 403
6.83.3.11 setCacheWrites() . . . 403
6.83.3.12 setConnection() . . . 403
6.83.3.13 setDBMS() . . . 403
6.83.3.14 setName() . . . 404
6.83.3.15 setOption() . . . 404
6.83.3.16 setPassword() . . . 404
6.83.3.17 setUsername() . . . 404

6.84 minerule::MRDebugPusher Class Reference . . . 404
6.84.1 Detailed Description . . . 405
6.84.2 Constructor & Destructor Documentation . . . 405
6.84.2.1 MRDebugPusher() . . . 405
6.84.2.2 ~MRDebugPusher() . . . 405
6.85 minerule::MRErrPusher Class Reference . . . 405
6.85.1 Detailed Description . . . 406
6.85.2 Constructor & Destructor Documentation . . . 406
6.85.2.1 MRErrPusher() . . . 406
6.85.2.2 ~MRErrPusher() . . . 406
6.86 minerule::MRLogger Class Reference . . . 406
6.86.1 Detailed Description . . . 407
6.86.2 Constructor & Destructor Documentation . . . 407
6.86.2.1 MRLogger() [1/2] . . . 407
6.86.2.2 MRLogger() [2/2] . . . 407
6.86.2.3 ~MRLogger() . . . 408
6.86.3 Member Function Documentation . . . 408
6.86.3.1 getCurrentCpuDelta() . . . 408
6.86.3.2 getCurrentCpuSecs() . . . 408
6.86.3.3 getCurrentTimeDelta() . . . 408
6.86.3.4 getCurrentTimeSecs() . . . 409
6.86.3.5 getIndentLen() . . . 409
6.86.3.6 getLogLevel() . . . 409
6.86.3.7 getStream() . . . 409
6.86.3.8 log() . . . 409
6.86.3.9 logMeasurements() . . . 410
6.86.3.10 pop() . . . 410
6.86.3.11 push() . . . 410
6.86.3.12 setLogLevel() . . . 411
6.86.3.13 setStream() . . . 411
6.86.3.14 startMeasuring() . . . 411
6.86.3.15 stopMeasuring() . . . 411
6.86.3.16 updateIndentString() . . . 412
6.87 minerule::MRLogPusher Class Reference . . . 412
6.87.1 Detailed Description . . . 412
6.87.2 Constructor & Destructor Documentation . . . 412
6.87.2.1 MRLogPusher() . . . 412
6.87.2.2 ~MRLogPusher() . . . 413
6.88 MRResultSetIterator Class Reference . . . 413
6.88.1 Detailed Description . . . 413
6.88.2 Constructor & Destructor Documentation . . . 413
6.88.2.1 MRResultSetIterator() . . . 413

6.88.3 Member Function Documentation . . . 413
6.88.3.1 getResultSet() . . . 414
6.88.3.2 next() . . . 414
6.88.3.3 reset() . . . 414
6.89 minerule::MRWarnPusher Class Reference . . . 414
6.89.1 Detailed Description . . . 414
6.89.2 Constructor & Destructor Documentation . . . 415
6.89.2.1 MRWarnPusher() . . . 415
6.89.2.2 ∼MRWarnPusher() . . . 415
6.90 minerule::MySqlCatalogueInstaller Class Reference . . . 415
6.90.1 Detailed Description . . . 416
6.90.2 Member Enumeration Documentation . . . 416
6.90.2.1 SupportedDbms . . . 416
6.90.3 Constructor & Destructor Documentation . . . 417
6.90.3.1 MySqlCatalogueInstaller() . . . 417
6.90.3.2 ∼MySqlCatalogueInstaller() . . . 417
6.90.4 Member Function Documentation . . . 417
6.90.4.1 dropMRAttList() . . . 417
6.90.4.2 dropMRAutoincrement() . . . 417
6.90.4.3 dropMRDepFun() . . . 418
6.90.4.4 dropMRDepFunCol() . . . 418
6.90.4.5 dropMREqKeys() . . . 418
6.90.4.6 dropMREqKeysCol() . . . 418
6.90.4.7 dropMRQuery() . . . 419
6.90.4.8 initializeAutoincrement() . . . 419
6.90.4.9 install() . . . 419
6.90.4.10 installMRAttList() . . . 419
6.90.4.11 installMRAutoincrement() . . . 420
6.90.4.12 installMRDepFun() . . . 420
6.90.4.13 installMRDepFunCol() . . . 420
6.90.4.14 installMREqKeys() . . . 420
6.90.4.15 installMREqKeysCol() . . . 421
6.90.4.16 installMRQuery() . . . 421
6.90.4.17 newInstaller() [1/2] . . . 421
6.90.4.18 newInstaller() [2/2] . . . 422
6.90.4.19 uninstall() . . . 422
6.90.5 Field Documentation . . . 422
6.90.5.1 _connection . . . 422
6.90.5.2 _statement . . . 422
6.91 minerule::NewRule Class Reference . . . 423
6.91.1 Detailed Description . . . 423
6.91.2 Constructor & Destructor Documentation . . . 423

6.91.2.1 NewRule() [1/7] . . . 423
6.91.2.2 NewRule() [2/7] . . . 424
6.91.2.3 NewRule() [3/7] . . . 424
6.91.2.4 NewRule() [4/7] . . . 424
6.91.2.5 NewRule() [5/7] . . . 424
6.91.2.6 NewRule() [6/7] . . . 425
6.91.2.7 NewRule() [7/7] . . . 425
6.91.3 Friends And Related Function Documentation . . . 425
6.91.3.1 operator<< . . . 425
6.91.4 Field Documentation . . . 425
6.91.4.1 body . . . 426
6.91.4.2 bodySupp . . . 426
6.91.4.3 gids . . . 426
6.91.4.4 head . . . 426
6.91.4.5 lastBody . . . 426
6.91.4.6 lastHead . . . 426
6.91.4.7 satisfy . . . 427
6.92 minerule::NewRuleSet Class Reference . . . 427
6.92.1 Detailed Description . . . 427
6.92.2 Field Documentation . . . 427
6.92.2.1 elements . . . 427
6.93 minerule::NodeRow Class Reference . . . 428
6.93.1 Detailed Description . . . 428
6.93.2 Constructor & Destructor Documentation . . . 428
6.93.2.1 NodeRow() [1/2] . . . 428
6.93.2.2 NodeRow() [2/2] . . . 429
6.93.2.3 ∼NodeRow() . . . 429
6.93.3 Member Function Documentation . . . 429
6.93.3.1 decremSupp() . . . 429
6.93.3.2 demark() . . . 429
6.93.3.3 getChild() . . . 429
6.93.3.4 getSupp() . . . 430
6.93.3.5 incremSupp() . . . 430
6.93.3.6 isMarked() . . . 430
6.93.3.7 mark() . . . 430
6.93.3.8 setChild() . . . 430
6.93.3.9 setSupp() . . . 430
6.94 minerule::NodeRowB Class Reference . . . 431
6.94.1 Detailed Description . . . 431
6.94.2 Constructor & Destructor Documentation . . . 431
6.94.2.1 NodeRowB() [1/2] . . . 431
6.94.2.2 NodeRowB() [2/2] . . . 432

6.94.2.3 ~NodeRowB() . . . 432
6.94.3 Member Function Documentation . . . 432
6.94.3.1 decremSupp() . . . 432
6.94.3.2 demark() . . . 432
6.94.3.3 getChild() . . . 432
6.94.3.4 getHead() . . . 433
6.94.3.5 getSupp() . . . 433
6.94.3.6 incremSupp() . . . 433
6.94.3.7 isMarked() . . . 433
6.94.3.8 mark() . . . 433
6.94.3.9 setChild() . . . 433
6.94.3.10 setHead() . . . 434
6.94.3.11 setSupp() . . . 434
6.95 minerule::NumericSourceRowAttribute Class Reference . . . 434
6.95.1 Detailed Description . . . 435
6.95.2 Member Typedef Documentation . . . 436
6.95.2.1 ElementType . . . 436
6.95.3 Constructor & Destructor Documentation . . . 436
6.95.3.1 NumericSourceRowAttribute() [1/3] . . . 436
6.95.3.2 NumericSourceRowAttribute() [2/3] . . . 436
6.95.3.3 NumericSourceRowAttribute() [3/3] . . . 436
6.95.3.4 ~NumericSourceRowAttribute() . . . 436
6.95.4 Member Function Documentation . . . 437
6.95.4.1 asString() . . . 437
6.95.4.2 compareTo() . . . 437
6.95.4.3 copy() . . . 438
6.95.4.4 createAttribute() . . . 438
6.95.4.5 createElement() . . . 438
6.95.4.6 createElementFromType() . . . 439
6.95.4.7 deserialize() . . . 439
6.95.4.8 deserializeElementFromResultSet() . . . 439
6.95.4.9 deserializeElementFromString() . . . 440
6.95.4.10 empty() . . . 440
6.95.4.11 getElementType() . . . 440
6.95.4.12 getFullElementType() . . . 440
6.95.4.13 getSQLData() . . . 441
6.95.4.14 getType() . . . 441
6.95.4.15 operator"!=() . . . 441
6.95.4.16 operator()() . . . 442
6.95.4.17 operator<() . . . 442
6.95.4.18 operator<<() . . . 442
6.95.4.19 operator=() . . . 443

6.95.4.20 operator==() . . . 443
6.95.4.21 serialize() . . . 443
6.95.4.22 serializeElementToString() . . . 444
6.95.4.23 setPreparedStatementParameters() . . . 444
6.95.4.24 setValue() [1/2] . . . 444
6.95.4.25 setValue() [2/2] . . . 444
6.95.5 Friends And Related Function Documentation . . . 445
6.95.5.1 operator<< . . . 445
6.95.6 Field Documentation . . . 445
6.95.6.1 elementType . . . 445
6.96 minerule::OptimizedMinerule::OptimizationInfo Class Reference . . . 446
6.96.1 Detailed Description . . . 446
6.96.2 Field Documentation . . . 446
6.96.2.1 combinationFormula . . . 446
6.96.2.2 minerule . . . 446
6.96.2.3 minerulesToCombine . . . 446
6.96.2.4 relationship . . . 447
6.97 Optimizations Class Reference . . . 447
6.97.1 Detailed Description . . . 448
6.97.2 Member Enumeration Documentation . . . 448
6.97.2.1 PreferredIncrementalAlgorithm . . . 448
6.97.3 Constructor & Destructor Documentation . . . 448
6.97.3.1 Optimizations() . . . 448
6.97.3.2 ~Optimizations() . . . 448
6.97.4 Member Function Documentation . . . 449
6.97.4.1 className() . . . 449
6.97.4.2 getAvoidCombinationDetection() . . . 449
6.97.4.3 getAvoidDominanceDetection() . . . 449
6.97.4.4 getAvoidEquivalenceDetection() . . . 449
6.97.4.5 getCatalogue() . . . 450
6.97.4.6 getCombinator() [1/2] . . . 450
6.97.4.7 getCombinator() [2/2] . . . 450
6.97.4.8 getIncrementalAlgorithm() . . . 450
6.97.4.9 getTryOptimizations() . . . 450
6.97.4.10 setAvoidCombinationDetection() . . . 451
6.97.4.11 setAvoidDominanceDetection() . . . 451
6.97.4.12 setAvoidEquivalenceDetection() . . . 451
6.97.4.13 setIncrementalAlgorithm() . . . 451
6.97.4.14 setOption() . . . 451
6.97.4.15 setTryOptimizations() . . . 452
6.97.4.16 subclassForName() . . . 452
6.98 minerule::OptimizedMinerule Class Reference . . . 452

6.98.1 Detailed Description . . . 453
6.98.2 Member Enumeration Documentation . . . 453
6.98.2.1 MineruleRelationship . . . 453
6.98.3 Constructor & Destructor Documentation . . . 454
6.98.3.1 OptimizedMinerule() . . . 454
6.98.4 Member Function Documentation . . . 454
6.98.4.1 attributesInPredicate() . . . 454
6.98.4.2 checkForCombinedQueries() . . . 454
6.98.4.3 dropTableIfExists() . . . 455
6.98.4.4 firstMineruleDominatesSecondMinerule() . . . 456
6.98.4.5 firstMineruleIncludesSecondMinerule() . . . 456
6.98.4.6 getMineruleRelationship() . . . 457
6.98.4.7 getOptimizationInfo() [1/2] . . . 457
6.98.4.8 getOptimizationInfo() [2/2] . . . 458
6.98.4.9 getParsedMinerule() . . . 458
6.98.4.10 getRelationshipType() . . . 458
6.98.4.11 hasIDConstraints() . . . 458
6.98.4.12 isACandidateQuery() . . . 458
6.98.4.13 optimize() . . . 459
6.98.4.14 predicateAttributesAreIncluded() . . . 460
6.99 minerule::OptimizerCatalogue Class Reference . . . 461
6.99.1 Detailed Description . . . 461
6.99.2 Member Typedef Documentation . . . 462
6.99.2.1 CatalogueEntry . . . 462
6.99.2.2 KeyCols . . . 462
6.99.3 Member Enumeration Documentation . . . 462
6.99.3.1 CatalogueType . . . 462
6.99.3.2 OrderType . . . 462
6.99.4 Constructor & Destructor Documentation . . . 463
6.99.4.1 OptimizerCatalogue() . . . 463
6.99.5 Member Function Documentation . . . 463
6.99.5.1 addDerivedResult() . . . 463
6.99.5.2 addMineruleAttributeList() . . . 464
6.99.5.3 addMineruleResult() . . . 464
6.99.5.4 checkInstallation() . . . 464
6.99.5.5 deleteMinerule() . . . 465
6.99.5.6 existsMinerule() . . . 466
6.99.5.7 getCatalogue() . . . 467
6.99.5.8 getMRQueryInfo() . . . 467
6.99.5.9 getMRQueryInfos() . . . 467
6.99.5.10 getMRQueryName() . . . 468
6.99.5.11 getMRQueryNames() . . . 468

6.99.5.12 getMRQueryResultIterator() . . . 469
6.99.5.13 getNewAutoincrementValue() . . . 469
6.99.5.14 getResultsetName() . . . 470
6.99.5.15 hasIDConstraints() . . . 470
6.99.5.16 install() . . . 471
6.99.5.17 isIDAttribute() . . . 471
6.99.5.18 uninstall() . . . 472
6.100 minerule::OptionBase Class Reference . . . 472
6.100.1 Detailed Description . . . 472
6.100.2 Constructor & Destructor Documentation . . . 473
6.100.2.1 ∼OptionBase() . . . 473
6.100.3 Member Function Documentation . . . 473
6.100.3.1 className() . . . 473
6.100.3.2 setOption() . . . 473
6.100.3.3 subclassForName() . . . 473
6.101 mrc::Options Class Reference . . . 473
6.101.1 Detailed Description . . . 474
6.101.2 Member Enumeration Documentation . . . 474
6.101.2.1 Command . . . 474
6.101.2.2 QryParams . . . 475
6.101.3 Constructor & Destructor Documentation . . . 475
6.101.3.1 Options() . . . 475
6.101.4 Member Function Documentation . . . 475
6.101.4.1 getCommand() . . . 476
6.101.4.2 getDerivedQuery() . . . 476
6.101.4.3 getListFormat() . . . 476
6.101.4.4 getOriginalQuery() . . . 476
6.101.4.5 getSearchParam() . . . 476
6.101.4.6 setAddDerivedQuery() . . . 477
6.101.4.7 setCheckCatalogue() . . . 477
6.101.4.8 setDeleteQry() . . . 477
6.101.4.9 setInstallCatalogue() . . . 477
6.101.4.10 setListFormat() . . . 478
6.101.4.11 setSearchParam() . . . 478
6.101.4.12 setSearchQry() . . . 478
6.101.4.13 setShowList() . . . 479
6.101.4.14 setUninstallCatalogue() . . . 479
6.101.5 Field Documentation . . . 479
6.101.5.1 command . . . 479
6.101.5.2 derivedQuery . . . 479
6.101.5.3 listFormat . . . 479
6.101.5.4 originalQuery . . . 480

6.101.5.5 searchParam . . . 480
6.102 mrmatch::Options Class Reference . . . 480
6.102.1 Detailed Description . . . 480
6.102.2 Constructor & Destructor Documentation . . . 480
6.102.2.1 Options() . . . 481
6.102.2.2 ∼Options() . . . 481
6.102.3 Member Function Documentation . . . 481
6.102.3.1 getMatchOutputTableName() . . . 481
6.102.3.2 initMineruleOptions() . . . 481
6.102.3.3 matcherSpecs() . . . 482
6.102.3.4 parse() . . . 482
6.102.3.5 printUsage() . . . 482
6.102.3.6 queryName() . . . 483
6.102.3.7 setMatchKind() . . . 483
6.102.3.8 setMatchOutput() . . . 484
6.102.3.9 setMatchOutputTableName() . . . 484
6.102.3.10 setQueryName() . . . 484
6.102.3.11 setQueryNumber() . . . 484
6.102.3.12 setTableName() . . . 484
6.102.3.13 tableName() . . . 485
6.103 mrprint::Options Class Reference . . . 485
6.103.1 Detailed Description . . . 485
6.103.2 Constructor & Destructor Documentation . . . 485
6.103.2.1 Options() . . . 485
6.103.2.2 ∼Options() . . . 486
6.103.3 Member Function Documentation . . . 486
6.103.3.1 mroptFileName() . . . 486
6.103.3.2 noLogArtifacts() . . . 486
6.103.3.3 noLowConfidenceFilter() . . . 486
6.103.3.4 parse() . . . 487
6.103.3.5 printHelp() . . . 487
6.103.3.6 queryName() . . . 488
6.103.3.7 queryNumber() . . . 488
6.103.3.8 sortOrder() . . . 488
6.104 OutStream Class Reference . . . 488
6.104.1 Detailed Description . . . 489
6.104.2 Constructor & Destructor Documentation . . . 489
6.104.2.1 OutStream() . . . 489
6.104.2.2 ∼OutStream() . . . 489
6.104.3 Member Function Documentation . . . 489
6.104.3.1 className() . . . 490
6.104.3.2 disable() . . . 490

6.104.3.3 enable() . . . 490
6.104.3.4 getLogger() . . . 490
6.104.3.5 getStream() . . . 490
6.104.3.6 setLogger() . . . 491
6.104.3.7 setLogLevel() . . . 491
6.104.3.8 setOption() . . . 491
6.105 minerule::ParsedMinerule Class Reference . . . 491
6.105.1 Detailed Description . . . 492
6.105.2 Member Typedef Documentation . . . 493
6.105.2.1 AttrVector . . . 493
6.105.2.2 bem_c . . . 493
6.105.3 Constructor & Destructor Documentation . . . 493
6.105.3.1 ParsedMinerule() [1/3] . . . 493
6.105.3.2 ParsedMinerule() [2/3] . . . 493
6.105.3.3 ∼ParsedMinerule() . . . 494
6.105.3.4 ParsedMinerule() [3/3] . . . 494
6.105.4 Member Function Documentation . . . 494
6.105.4.1 getBEMText() . . . 494
6.105.4.2 getMineitemsetsText() . . . 495
6.105.4.3 getMineruleText() . . . 495
6.105.4.4 getMinesequenceText() . . . 496
6.105.4.5 getText() . . . 496
6.105.4.6 hasCrossConditions() [1/2] . . . 496
6.105.4.7 hasCrossConditions() [2/2] . . . 497
6.105.4.8 hasDisjuctionsInMC() . . . 497
6.105.4.9 hasSameBodyHead() . . . 497
6.105.4.10 init() . . . 498
6.105.4.11 requiresClusters() . . . 498
6.105.5 Field Documentation . . . 498
6.105.5.1 ba . . . 498
6.105.5.2 body_coincident_head . . . 498
6.105.5.3 bodyCardinalities . . . 498
6.105.5.4 c_aggr_list . . . 499
6.105.5.5 ca . . . 499
6.105.5.6 cc . . . 499
6.105.5.7 conf . . . 499
6.105.5.8 distinct . . . 499
6.105.5.9 filter_condition . . . 499
6.105.5.10 ga . . . 500
6.105.5.11 gc . . . 500
6.105.5.12 ha . . . 500
6.105.5.13 headCardinalities . . . 500

6.105.5.14 length . . . 500
6.105.5.15 mc . . . 500
6.105.5.16 miningTask . . . 501
6.105.5.17 oa . . . 501
6.105.5.18 ra . . . 501
6.105.5.19 seq_bem_vect . . . 501
6.105.5.20 seq_dist_vect . . . 501
6.105.5.21 sequenceAllowedGaps . . . 501
6.105.5.22 sup . . . 502
6.105.5.23 tab_result . . . 502
6.105.5.24 tab_source . . . 502
6.105.5.25 tautologies . . . 502
6.106 minerule::ParsedPredConjunction Class Reference . . . 502
6.106.1 Detailed Description . . . 503
6.106.2 Constructor & Destructor Documentation . . . 503
6.106.2.1 ParsedPredConjunction() [1/2] . . . 503
6.106.2.2 ParsedPredConjunction() [2/2] . . . 503
6.106.3 Member Function Documentation . . . 504
6.106.3.1 areAllAggr_f() . . . 504
6.106.3.2 areAllBorH() . . . 504
6.106.3.3 atLeastOneBorH() . . . 505
6.106.3.4 cloneInstance() . . . 505
6.106.3.5 copy_and_append() . . . 505
6.106.3.6 find() . . . 506
6.106.3.7 newInstance() . . . 506
6.106.3.8 newPredConjunction() . . . 506
6.106.3.9 newPredicate() . . . 507
6.106.3.10 operator"!() . . . 507
6.106.3.11 operator&=() . . . 507
6.106.4 Field Documentation . . . 507
6.106.4.1 elements . . . 508
6.107 minerule::ParsedPredicate Class Reference . . . 508
6.107.1 Detailed Description . . . 509
6.107.2 Constructor & Destructor Documentation . . . 509
6.107.2.1 ParsedPredicate() [1/2] . . . 509
6.107.2.2 ParsedPredicate() [2/2] . . . 509
6.107.3 Member Function Documentation . . . 509
6.107.3.1 areAllAggr_f() . . . 509
6.107.3.2 areAllBorH() . . . 510
6.107.3.3 atLeastOneBorH() . . . 510
6.107.3.4 cloneInstance() . . . 511
6.107.3.5 convert() . . . 511

6.107.3.6 convert_and_list() . . . 512
6.107.3.7 merge() . . . 512
6.107.3.8 newInstance() . . . 512
6.107.3.9 newPredConjunction() . . . 513
6.107.3.10 newPredicate() . . . 513
6.107.3.11 operator"!() . . . 513
6.107.3.12 operator&=() . . . 514
6.107.3.13 operator"|=() . . . 514
6.107.3.14 stamp() . . . 514
6.107.4 Field Documentation . . . 515
6.107.4.1 elements . . . 515
6.108 minerule::ParsedSimplePredicate Class Reference . . . 515
6.108.1 Detailed Description . . . 516
6.108.2 Constructor & Destructor Documentation . . . 516
6.108.2.1 ParsedSimplePredicate() [1/3] . . . 516
6.108.2.2 ParsedSimplePredicate() [2/3] . . . 516
6.108.2.3 ParsedSimplePredicate() [3/3] . . . 517
6.108.3 Member Function Documentation . . . 517
6.108.3.1 cloneInstance() . . . 517
6.108.3.2 getOp() . . . 518
6.108.3.3 getVal1() . . . 518
6.108.3.4 getVal2() . . . 518
6.108.3.5 isAggr_function() . . . 519
6.108.3.6 isPartOfBorH() . . . 519
6.108.3.7 newInstance() . . . 519
6.108.3.8 newPredConjunction() . . . 520
6.108.3.9 newPredicate() . . . 520
6.108.3.10 operator"!() . . . 520
6.108.3.11 operator"!=() . . . 521
6.108.3.12 operator<() . . . 521
6.108.3.13 operator==() . . . 521
6.109 Parsers Class Reference . . . 522
6.109.1 Detailed Description . . . 522
6.109.2 Constructor & Destructor Documentation . . . 522
6.109.2.1 Parsers() . . . 522
6.109.2.2 ~Parsers() . . . 523
6.109.3 Member Function Documentation . . . 523
6.109.3.1 className() . . . 523
6.109.3.2 getBodyCardinalities() . . . 523
6.109.3.3 getHeadCardinalities() . . . 523
6.109.3.4 getLogFILE() . . . 523
6.109.3.5 setLogFILE() . . . 524

6.109.3.6 setLogOnStderr() . . . 524
6.109.3.7 setLogOnStdout() . . . 524
6.109.3.8 setMaxBodyElems() . . . 524
6.109.3.9 setMaxHeadElems() . . . 524
6.109.3.10 setMinBodyElems() . . . 524
6.109.3.11 setMinHeadElems() . . . 525
6.109.3.12 setOption() . . . 525
6.110 RulesMiningAlgorithms::PartitionBase Class Reference . . . 525
6.110.1 Detailed Description . . . 525
6.110.2 Constructor & Destructor Documentation . . . 525
6.110.2.1 ∼PartitionBase() . . . 526
6.110.3 Member Function Documentation . . . 526
6.110.3.1 className() . . . 526
6.110.3.2 getRowsPerPartition() . . . 526
6.110.3.3 setOption() . . . 526
6.110.3.4 setRowsPerPartition() . . . 526
6.111 RulesMiningAlgorithms::PartitionWithClusters Class Reference . . . 527
6.111.1 Detailed Description . . . 527
6.111.2 Constructor & Destructor Documentation . . . 527
6.111.2.1 ∼PartitionWithClusters() . . . 527
6.111.3 Member Function Documentation . . . 527
6.111.3.1 className() . . . 527
6.111.3.2 getRowsPerPartition() . . . 528
6.111.3.3 setOption() . . . 528
6.111.3.4 setRowsPerPartition() . . . 528
6.112 minerule::PostgresCatalogueInstaller Class Reference . . . 528
6.112.1 Detailed Description . . . 529
6.112.2 Member Enumeration Documentation . . . 529
6.112.2.1 SupportedDbms . . . 529
6.112.3 Constructor & Destructor Documentation . . . 530
6.112.3.1 PostgresCatalogueInstaller() . . . 530
6.112.3.2 ∼PostgresCatalogueInstaller() . . . 530
6.112.4 Member Function Documentation . . . 530
6.112.4.1 dropMRAttList() . . . 530
6.112.4.2 dropMRAutoincrement() . . . 530
6.112.4.3 dropMRDepFun() . . . 531
6.112.4.4 dropMRDepFunCol() . . . 531
6.112.4.5 dropMREqKeys() . . . 531
6.112.4.6 dropMREqKeysCol() . . . 531
6.112.4.7 dropMRQuery() . . . 532
6.112.4.8 initializeAutoincrement() . . . 532
6.112.4.9 install() . . . 532

6.112.4.10 installMRAttList() . . . 532
6.112.4.11 installMRAutoincrement() . . . 533
6.112.4.12 installMRDepFun() . . . 533
6.112.4.13 installMRDepFunCol() . . . 533
6.112.4.14 installMREqKeys() . . . 533
6.112.4.15 installMREqKeysCol() . . . 534
6.112.4.16 installMRQuery() . . . 534
6.112.4.17 newInstaller() [1/2] . . . 534
6.112.4.18 newInstaller() [2/2] . . . 535
6.112.4.19 uninstall() . . . 535
6.112.5 Field Documentation . . . 535
6.112.5.1 _connection . . . 535
6.112.5.2 _statement . . . 535
6.113 minerule::PredConjunction Class Reference . . . 536
6.113.1 Detailed Description . . . 536
6.113.2 Constructor & Destructor Documentation . . . 536
6.113.2.1 PredConjunction() [1/3] . . . 536
6.113.2.2 PredConjunction() [2/3] . . . 537
6.113.2.3 PredConjunction() [3/3] . . . 537
6.113.2.4 ~PredConjunction() . . . 537
6.113.3 Member Function Documentation . . . 537
6.113.3.1 evaluate() . . . 537
6.113.3.2 operator&=() . . . 538
6.113.4 Field Documentation . . . 538
6.113.4.1 elements . . . 538
6.114 minerule::PredConjunctionBase Class Reference . . . 538
6.114.1 Detailed Description . . . 539
6.114.2 Constructor & Destructor Documentation . . . 539
6.114.2.1 PredConjunctionBase() [1/2] . . . 539
6.114.2.2 PredConjunctionBase() [2/2] . . . 539
6.114.2.3 ~PredConjunctionBase() . . . 539
6.114.3 Member Function Documentation . . . 540
6.114.3.1 cloneInstance() . . . 540
6.114.3.2 copy_and_append() . . . 540
6.114.3.3 find() . . . 540
6.114.3.4 newInstance() . . . 541
6.114.3.5 newPredConjunction() . . . 541
6.114.3.6 newPredicate() . . . 541
6.114.3.7 operator"!() . . . 542
6.114.3.8 operator&=() . . . 542
6.114.4 Field Documentation . . . 542
6.114.4.1 elements . . . 542

6.115 minerule::Predicate Class Reference . . . 543
6.115.1 Detailed Description . . . 543
6.115.2 Constructor & Destructor Documentation . . . 543
6.115.2.1 Predicate() [1/3] . . . 543
6.115.2.2 Predicate() [2/3] . . . 544
6.115.2.3 Predicate() [3/3] . . . 544
6.115.2.4 ~Predicate() . . . 544
6.115.3 Member Function Documentation . . . 544
6.115.3.1 evaluate() . . . 544
6.115.3.2 getNumVariables() . . . 545
6.115.3.3 getPredicateList() . . . 545
6.115.3.4 operator&=() . . . 545
6.115.3.5 operator=() . . . 546
6.115.3.6 operator"|=() . . . 546
6.115.3.7 setNumVariables() . . . 546
6.115.4 Field Documentation . . . 546
6.115.4.1 elements . . . 547
6.116 minerule::PredicateBase Class Reference . . . 547
6.116.1 Detailed Description . . . 548
6.116.2 Constructor & Destructor Documentation . . . 548
6.116.2.1 PredicateBase() [1/2] . . . 548
6.116.2.2 PredicateBase() [2/2] . . . 548
6.116.2.3 ~PredicateBase() . . . 548
6.116.3 Member Function Documentation . . . 548
6.116.3.1 cloneInstance() . . . 549
6.116.3.2 merge() . . . 549
6.116.3.3 newInstance() . . . 549
6.116.3.4 newPredConjunction() . . . 550
6.116.3.5 newPredicate() . . . 550
6.116.3.6 operator"!() . . . 550
6.116.3.7 operator&=() . . . 551
6.116.3.8 operator=() . . . 551
6.116.3.9 operator"|=() . . . 551
6.116.3.10 stamp() . . . 552
6.116.4 Field Documentation . . . 552
6.116.4.1 elements . . . 552
6.117 minerule::PredicateUtils Class Reference . . . 552
6.117.1 Detailed Description . . . 553
6.117.2 Member Typedef Documentation . . . 553
6.117.2.1 PredicateRelationship . . . 553
6.117.3 Member Function Documentation . . . 553
6.117.3.1 getPredicateRelationship() . . . 553

6.117.3.2 predicatesAreEquivalent() . . . 554
6.118 minerule::PrepareDataUtils Class Reference . . . 554
6.118.1 Detailed Description . . . 555
6.118.2 Constructor & Destructor Documentation . . . 555
6.118.2.1 PrepareDataUtils() . . . 555
6.118.3 Member Function Documentation . . . 555
6.118.3.1 buildAttrListDescription() . . . 555
6.118.3.2 buildBodyTableQuery() . . . 556
6.118.3.3 buildExtendedSourceTableQuery() . . . 556
6.118.3.4 buildHeadTableQuery() . . . 556
6.118.3.5 buildSourceTableQuery() . . . 557
6.118.3.6 dropTableIfExists() . . . 557
6.118.3.7 evaluateTotGroups() [1/2] . . . 557
6.118.3.8 evaluateTotGroups() [2/2] . . . 558
6.119 mrdb::PreparedStatement Class Reference . . . 558
6.119.1 Detailed Description . . . 558
6.119.2 Constructor & Destructor Documentation . . . 559
6.119.2.1 ~PreparedStatement() . . . 559
6.119.3 Member Function Documentation . . . 559
6.119.3.1 execute() . . . 559
6.119.3.2 executeQuery() . . . 559
6.119.3.3 setDouble() . . . 560
6.119.3.4 setInt() . . . 560
6.119.3.5 setLong() . . . 560
6.119.3.6 setString() . . . 561
6.120 mrc::Printer Class Reference . . . 561
6.120.1 Detailed Description . . . 561
6.120.2 Constructor & Destructor Documentation . . . 561
6.120.2.1 Printer() . . . 561
6.120.3 Member Function Documentation . . . 562
6.120.3.1 print() [1/2] . . . 562
6.120.3.2 print() [2/2] . . . 562
6.121 minerule::Progress Class Reference . . . 562
6.121.1 Detailed Description . . . 563
6.121.2 Member Typedef Documentation . . . 563
6.121.2.1 UpdateHandler . . . 563
6.121.3 Constructor & Destructor Documentation . . . 563
6.121.3.1 Progress() . . . 563
6.121.3.2 ~Progress() . . . 563
6.121.4 Member Function Documentation . . . 564
6.121.4.1 end() . . . 564
6.121.4.2 handleStartStop() . . . 564

6.121.4.3 setDefaultHandler() . . . 564
6.121.4.4 setHandler() . . . 565
6.121.4.5 start() . . . 565
6.121.4.6 tick() . . . 565
6.122 minerule::PtrSimplePredComp Class Reference . . . 565
6.122.1 Detailed Description . . . 565
6.122.2 Member Function Documentation . . . 566
6.122.2.1 operator()() . . . 566
6.123 minerule::QueryNormalizer Class Reference . . . 566
6.123.1 Detailed Description . . . 566
6.123.2 Member Typedef Documentation . . . 566
6.123.2.1 BodyInfo . . . 567
6.123.2.2 HeadInfo . . . 567
6.123.3 Constructor & Destructor Documentation . . . 567
6.123.3.1 QueryNormalizer() . . . 567
6.123.4 Member Function Documentation . . . 567
6.123.4.1 normalize() . . . 567
6.123.4.2 setMinerule() . . . 568
6.124 minerule::QueryResult Class Reference . . . 568
6.124.1 Detailed Description . . . 568
6.125 minerule::ResultCombinator Class Reference . . . 568
6.125.1 Detailed Description . . . 569
6.125.2 Constructor & Destructor Documentation . . . 569
6.125.2.1 ResultCombinator() . . . 569
6.125.2.2 ∼ResultCombinator() . . . 569
6.125.3 Member Function Documentation . . . 569
6.125.3.1 execute() . . . 570
6.125.3.2 newIncrementalAlgorithm() . . . 570
6.125.4 Field Documentation . . . 571
6.125.4.1 minerule . . . 571
6.126 minerule::QueryResult::ResultSet< Sorter > Class Template Reference . . . 571
6.126.1 Detailed Description . . . 572
6.126.2 Constructor & Destructor Documentation . . . 572
6.126.2.1 ResultSet() [1/2] . . . 572
6.126.2.2 ResultSet() [2/2] . . . 572
6.126.3 Member Function Documentation . . . 573
6.126.3.1 inplace_intersect() . . . 573
6.126.3.2 inplace_union() . . . 573
6.126.3.3 load() . . . 573
6.126.3.4 operator&=() . . . 574
6.126.3.5 operator"|=() . . . 574
6.126.3.6 save() . . . 574

6.126.4 Field Documentation . . . 574
6.126.4.1 keys . . . 575
6.127 mrdb::ResultSet Class Reference . . . 575
6.127.1 Detailed Description . . . 575
6.127.2 Constructor & Destructor Documentation . . . 575
6.127.2.1 ~ResultSet() . . . 575
6.127.3 Member Function Documentation . . . 576
6.127.3.1 getBoolean() . . . 576
6.127.3.2 getDouble() . . . 576
6.127.3.3 getFloat() . . . 576
6.127.3.4 getInt() . . . 576
6.127.3.5 getLong() . . . 577
6.127.3.6 getMetaData() . . . 577
6.127.3.7 getString() . . . 577
6.127.3.8 isAfterLast() . . . 577
6.127.3.9 isBeforeFirst() . . . 578
6.127.3.10 next() . . . 578
6.128 mrdb::ResultSetMetaData Class Reference . . . 578
6.128.1 Detailed Description . . . 578
6.128.2 Constructor & Destructor Documentation . . . 578
6.128.2.1 ~ResultSetMetaData() . . . 579
6.128.3 Member Function Documentation . . . 579
6.128.3.1 getColumnCount() . . . 579
6.128.3.2 getColumnName() . . . 579
6.128.3.3 getColumnType() . . . 579
6.128.3.4 getColumnTypeName() . . . 580
6.128.3.5 getPrecision() . . . 580
6.128.3.6 getScale() . . . 580
6.129 minerule::Rule Class Reference . . . 581
6.129.1 Detailed Description . . . 581
6.129.2 Constructor & Destructor Documentation . . . 582
6.129.2.1 Rule() [1/2] . . . 582
6.129.2.2 Rule() [2/2] . . . 582
6.129.2.3 ~Rule() . . . 582
6.129.3 Member Function Documentation . . . 582
6.129.3.1 clear() . . . 582
6.129.3.2 getBody() [1/2] . . . 583
6.129.3.3 getBody() [2/2] . . . 583
6.129.3.4 getBodyId() . . . 583
6.129.3.5 getConfidence() . . . 583
6.129.3.6 getHead() [1/2] . . . 583
6.129.3.7 getHead() [2/2] . . . 584

6.129.3.8 getHeadId() . . . 584
6.129.3.9 getSupport() . . . 584
6.129.3.10 operator"!=() . . . 584
6.129.3.11 operator<() . . . 584
6.129.3.12 operator<=() . . . 585
6.129.3.13 operator=() . . . 585
6.129.3.14 operator==() . . . 585
6.129.3.15 operator>() . . . 585
6.129.3.16 operator>=() . . . 586
6.129.3.17 setBody() . . . 586
6.129.3.18 setBodyId() . . . 586
6.129.3.19 setConfidence() . . . 586
6.129.3.20 setHead() . . . 586
6.129.3.21 setHeadId() . . . 587
6.129.3.22 setSupport() . . . 587
6.130 minerule::RuleFormatter Class Reference . . . 587
6.130.1 Detailed Description . . . 588
6.130.2 Constructor & Destructor Documentation . . . 588
6.130.2.1 RuleFormatter() . . . 588
6.130.2.2 ∼RuleFormatter() . . . 588
6.130.3 Member Function Documentation . . . 588
6.130.3.1 fieldWidths() [1/2] . . . 588
6.130.3.2 fieldWidths() [2/2] . . . 589
6.130.3.3 formatRule() . . . 589
6.130.3.4 postExec() . . . 589
6.130.3.5 printRule() . . . 589
6.130.3.6 quote() . . . 589
6.130.3.7 quoteElems() . . . 590
6.130.3.8 setFieldWidths() . . . 590
6.130.3.9 setSuppressLog() . . . 590
6.130.3.10 suppressLog() . . . 590
6.130.4 Field Documentation . . . 590
6.130.4.1 _bhSep . . . 591
6.130.4.2 _fieldWidths . . . 591
6.130.4.3 _suppressLog . . . 591
6.131 mrmatch::RuleGidsDBMatcher Class Reference . . . 591
6.131.1 Detailed Description . . . 592
6.131.2 Member Typedef Documentation . . . 592
6.131.2.1 Gids . . . 592
6.131.2.2 RuleGids . . . 593
6.131.2.3 RuleGidsVector . . . 593
6.131.3 Constructor & Destructor Documentation . . . 593

6.131.3.1 RuleGidsDBMatcher() . . . 593
6.131.3.2 ~RuleGidsDBMatcher() . . . 593
6.131.4 Member Function Documentation . . . 593
6.131.4.1 addRule() . . . 593
6.131.4.2 createOutputTable() . . . 594
6.131.4.3 createOutputTableIndex() . . . 594
6.131.4.4 formatGids() . . . 594
6.131.4.5 match() . . . 595
6.131.4.6 matchItemTransaction() . . . 595
6.131.4.7 matchRuleTransaction() . . . 595
6.131.4.8 matchWithCrossProduct() . . . 596
6.131.4.9 matchWithoutCrossProduct() . . . 596
6.131.4.10 newMatcher() . . . 596
6.131.4.11 printMatches() . . . 597
6.131.5 Field Documentation . . . 597
6.131.5.1 _minerule . . . 597
6.131.5.2 _outTableName . . . 597
6.131.5.3 _ruleGidsVector . . . 598
6.132 mrmatch::RuleGidsMatcher Class Reference . . . 598
6.132.1 Detailed Description . . . 599
6.132.2 Member Typedef Documentation . . . 599
6.132.2.1 Gids . . . 599
6.132.2.2 RuleGids . . . 599
6.132.2.3 RuleGidsVector . . . 599
6.132.3 Constructor & Destructor Documentation . . . 599
6.132.3.1 RuleGidsMatcher() . . . 600
6.132.3.2 ~RuleGidsMatcher() . . . 600
6.132.4 Member Function Documentation . . . 600
6.132.4.1 addRule() . . . 600
6.132.4.2 formatGids() . . . 600
6.132.4.3 match() . . . 601
6.132.4.4 matchItemTransaction() . . . 601
6.132.4.5 matchRuleTransaction() . . . 601
6.132.4.6 matchWithCrossProduct() . . . 602
6.132.4.7 matchWithoutCrossProduct() . . . 602
6.132.4.8 newMatcher() . . . 602
6.132.4.9 printMatches() . . . 603
6.132.5 Field Documentation . . . 603
6.132.5.1 _ruleGidsVector . . . 603
6.133 minerule::RulesMatcher Class Reference . . . 603
6.133.1 Detailed Description . . . 604
6.133.2 Member Typedef Documentation . . . 604

6.133.2.1 ItemSetType . . . 604
6.133.2.2 RuleSetType . . . 604
6.133.3 Member Function Documentation . . . 604
6.133.3.1 match() [1/2] . . . 604
6.133.3.2 match() [2/2] . . . 605
6.134 RulesMiningAlgorithms Class Reference . . . 605
6.134.1 Detailed Description . . . 606
6.134.2 Constructor & Destructor Documentation . . . 606
6.134.2.1 RulesMiningAlgorithms() [1/2] . . . 606
6.134.2.2 RulesMiningAlgorithms() [2/2] . . . 606
6.134.2.3 ∼RulesMiningAlgorithms() . . . 607
6.134.3 Member Function Documentation . . . 607
6.134.3.1 className() . . . 607
6.134.3.2 getFPGrowth() [1/2] . . . 607
6.134.3.3 getFPGrowth() [2/2] . . . 607
6.134.3.4 getPartitionBase() [1/2] . . . 607
6.134.3.5 getPartitionBase() [2/2] . . . 608
6.134.3.6 getPartitionWithClusters() [1/2] . . . 608
6.134.3.7 getPartitionWithClusters() [2/2] . . . 608
6.134.3.8 getPreferredAlgorithm() . . . 608
6.134.3.9 setOption() . . . 608
6.134.3.10 setPreferredAlgorithm() . . . 609
6.134.3.11 subclassForName() . . . 609
6.134.4 Field Documentation . . . 609
6.134.4.1 fpgrowth . . . 609
6.134.4.2 partitionbase . . . 609
6.134.4.3 partitiongeneralized . . . 609
6.134.4.4 preferredAlgorithm . . . 610
6.135 minerule::RuleTransaction< RuleSetType > Class Template Reference . . . 610
6.135.1 Detailed Description . . . 610
6.135.2 Constructor & Destructor Documentation . . . 610
6.135.2.1 RuleTransaction() . . . 611
6.135.3 Member Function Documentation . . . 611
6.135.3.1 findGid() . . . 611
6.135.3.2 load() . . . 611
6.136 Safety Class Reference . . . 611
6.136.1 Detailed Description . . . 612
6.136.2 Constructor & Destructor Documentation . . . 612
6.136.2.1 Safety() . . . 612
6.136.2.2 ∼Safety() . . . 612
6.136.3 Member Function Documentation . . . 612
6.136.3.1 className() . . . 612

6.136.3.2 getAllowCascadeDeletes() . . . 613
6.136.3.3 getOverwriteHomonymMinerules() . . . 613
6.136.3.4 setAllowCascadeDeletes() . . . 613
6.136.3.5 setOption() . . . 613
6.136.3.6 setOverwriteHomonymMinerules() . . . 613
6.137 minerule::Scmp Class Reference . . . 614
6.137.1 Detailed Description . . . 614
6.137.2 Member Function Documentation . . . 614
6.137.2.1 operator()() . . . 614
6.138 SequencesMiningAlgorithms Class Reference . . . 614
6.138.1 Detailed Description . . . 615
6.138.2 Constructor & Destructor Documentation . . . 615
6.138.2.1 SequencesMiningAlgorithms() [1/2] . . . 615
6.138.2.2 SequencesMiningAlgorithms() [2/2] . . . 615
6.138.2.3 ∼SequencesMiningAlgorithms() . . . 615
6.138.3 Member Function Documentation . . . 615
6.138.3.1 className() . . . 616
6.138.3.2 getPreferredAlgorithm() . . . 616
6.138.3.3 setOption() . . . 616
6.138.3.4 setPreferredAlgorithm() . . . 616
6.138.3.5 subclassForName() . . . 616
6.139 minerule::SimplePredAnalyzer Class Reference . . . 617
6.139.1 Detailed Description . . . 617
6.139.2 Member Function Documentation . . . 617
6.139.2.1 getOperatorIndex() . . . 617
6.139.2.2 getRelation() . . . 618
6.139.2.3 getValuesRelationshipIndex() . . . 618
6.139.2.4 isAttrOpValuePredicate() . . . 618
6.139.3 Field Documentation . . . 619
6.139.3.1 opRelationTable . . . 619
6.140 minerule::SimplePredicate Class Reference . . . 619
6.140.1 Detailed Description . . . 620
6.140.2 Member Function Documentation . . . 620
6.140.2.1 freeSimplePredicatePool() . . . 620
6.140.2.2 getOp() . . . 620
6.140.2.3 getVal1() . . . 620
6.140.2.4 getVal2() . . . 621
6.140.2.5 getVarId() . . . 621
6.140.2.6 newSimplePredicate() [1/2] . . . 621
6.140.2.7 newSimplePredicate() [2/2] . . . 621
6.140.2.8 operator<() . . . 622
6.140.2.9 operator==() . . . 622

6.140.2.10 setVarId() . . . 622
6.141 minerule::SimplePredicateBase Class Reference . . . 622
6.141.1 Detailed Description . . . 623
6.141.2 Constructor & Destructor Documentation . . . 623
6.141.2.1 SimplePredicateBase() [1/3] . . . 623
6.141.2.2 SimplePredicateBase() [2/3] . . . 623
6.141.2.3 SimplePredicateBase() [3/3] . . . 623
6.141.2.4 ∼SimplePredicateBase() . . . 624
6.141.3 Member Function Documentation . . . 624
6.141.3.1 cloneInstance() . . . 624
6.141.3.2 getOp() . . . 625
6.141.3.3 getVal1() . . . 625
6.141.3.4 getVal2() . . . 625
6.141.3.5 newInstance() . . . 626
6.141.3.6 newPredConjunction() . . . 626
6.141.3.7 newPredicate() . . . 626
6.141.3.8 operator"!() . . . 627
6.141.3.9 operator"!=() . . . 627
6.141.3.10 operator<() . . . 627
6.141.3.11 operator==() . . . 628
6.142 minerule::SimpleRuleFormatter Class Reference . . . 628
6.142.1 Detailed Description . . . 629
6.142.2 Constructor & Destructor Documentation . . . 629
6.142.2.1 SimpleRuleFormatter() . . . 629
6.142.2.2 ∼SimpleRuleFormatter() . . . 629
6.142.3 Member Function Documentation . . . 629
6.142.3.1 fieldWidths() [1/2] . . . 629
6.142.3.2 fieldWidths() [2/2] . . . 630
6.142.3.3 formatRule() . . . 630
6.142.3.4 postExec() . . . 630
6.142.3.5 printRule() . . . 630
6.142.3.6 quote() . . . 631
6.142.3.7 quoteElems() . . . 631
6.142.3.8 setFieldWidths() . . . 631
6.142.3.9 setSuppressLog() . . . 631
6.142.3.10 suppressLog() . . . 632
6.142.4 Field Documentation . . . 632
6.142.4.1 _bhSep . . . 632
6.142.4.2 _fieldWidths . . . 632
6.142.4.3 _suppressLog . . . 632
6.143 minerule::QueryResult::SortBodyHead Class Reference . . . 632
6.143.1 Detailed Description . . . 633

6.143.2 Member Function Documentation . . . 633
6.143.2.1 operator()() . . . 633
6.144 minerule::QueryResult::SortBodyHeadSuppConf Class Reference . . . 633
6.144.1 Detailed Description . . . 633
6.144.2 Member Function Documentation . . . 633
6.144.2.1 operator()() . . . 634
6.145 minerule::QueryResult::SortConfBodyHeadSupp Class Reference . . . 634
6.145.1 Detailed Description . . . 634
6.145.2 Member Function Documentation . . . 634
6.145.2.1 operator()() . . . 635
6.146 minerule::QueryResult::SortConfBodySuppHead Class Reference . . . 635
6.146.1 Detailed Description . . . 635
6.146.2 Member Function Documentation . . . 635
6.146.2.1 operator()() . . . 636
6.147 minerule::QueryResult::SortConfSuppBodyHead Class Reference . . . 636
6.147.1 Detailed Description . . . 636
6.147.2 Member Function Documentation . . . 636
6.147.2.1 operator()() . . . 637
6.148 minerule::SortedRuleFormatter< Sorter > Class Template Reference . . . 637
6.148.1 Detailed Description . . . 638
6.148.2 Constructor & Destructor Documentation . . . 638
6.148.2.1 SortedRuleFormatter() . . . 638
6.148.2.2 ~SortedRuleFormatter() . . . 638
6.148.3 Member Function Documentation . . . 638
6.148.3.1 fieldWidths() [1/2] . . . 639
6.148.3.2 fieldWidths() [2/2] . . . 639
6.148.3.3 formatRule() . . . 639
6.148.3.4 postExec() . . . 639
6.148.3.5 printRule() . . . 640
6.148.3.6 quote() . . . 640
6.148.3.7 quoteElems() . . . 640
6.148.3.8 setFieldWidths() . . . 640
6.148.3.9 setSuppressLog() . . . 641
6.148.3.10 suppressLog() . . . 641
6.148.4 Field Documentation . . . 641
6.148.4.1 _bhSep . . . 641
6.148.4.2 _fieldWidths . . . 641
6.148.4.3 _suppressLog . . . 641
6.149 minerule::QueryResult::SortHeadBodySuppConf Class Reference . . . 642
6.149.1 Detailed Description . . . 642
6.149.2 Member Function Documentation . . . 642
6.149.2.1 operator()() . . . 642

6.150 minerule::QueryResult::SortSuppConfBodyHead Class Reference . . . 642
6.150.1 Detailed Description . . . 643
6.150.2 Member Function Documentation . . . 643
6.150.2.1 operator()() . . . 643
6.151 minerule::SourceRow Class Reference . . . 643
6.151.1 Detailed Description . . . 644
6.151.2 Constructor & Destructor Documentation . . . 644
6.151.2.1 SourceRow() [1/3] . . . 644
6.151.2.2 SourceRow() [2/3] . . . 644
6.151.2.3 SourceRow() [3/3] . . . 644
6.151.2.4 ∼SourceRow() . . . 645
6.151.3 Member Function Documentation . . . 645
6.151.3.1 getBody() . . . 645
6.151.3.2 getClusterBody() . . . 645
6.151.3.3 getClusterHead() . . . 645
6.151.3.4 getGroup() . . . 646
6.151.3.5 getHead() . . . 646
6.151.3.6 init() . . . 646
6.151.4 Friends And Related Function Documentation . . . 646
6.151.4.1 operator<< . . . 647
6.152 minerule::SourceRowAttrCollectionDescriptor Class Reference . . . 647
6.152.1 Detailed Description . . . 647
6.152.2 Constructor & Destructor Documentation . . . 647
6.152.2.1 SourceRowAttrCollectionDescriptor() [1/2] . . . 647
6.152.2.2 SourceRowAttrCollectionDescriptor() [2/2] . . . 648
6.152.3 Member Function Documentation . . . 648
6.152.3.1 getColumnsCount() . . . 648
6.152.3.2 getSQLColumnNames() . . . 648
6.152.3.3 getSQLDataDefinition() . . . 648
6.152.3.4 init() . . . 649
6.152.3.5 questionMarks() . . . 649
6.153 minerule::SourceRowAttribute Class Reference . . . 649
6.153.1 Detailed Description . . . 650
6.153.2 Member Typedef Documentation . . . 650
6.153.2.1 ElementType . . . 650
6.153.3 Constructor & Destructor Documentation . . . 651
6.153.3.1 ∼SourceRowAttribute() . . . 651
6.153.4 Member Function Documentation . . . 651
6.153.4.1 asString() . . . 651
6.153.4.2 compareTo() . . . 651
6.153.4.3 copy() . . . 652
6.153.4.4 createAttribute() . . . 652

6.153.4.5 createElement() . . . 652
6.153.4.6 createElementFromType() . . . 653
6.153.4.7 deserialize() . . . 653
6.153.4.8 deserializeElementFromResultSet() . . . 653
6.153.4.9 deserializeElementFromString() . . . 654
6.153.4.10 empty() . . . 654
6.153.4.11 getElementType() . . . 654
6.153.4.12 getFullElementType() . . . 654
6.153.4.13 getSQLData() . . . 655
6.153.4.14 getType() . . . 655
6.153.4.15 operator"!=() . . . 655
6.153.4.16 operator()() . . . 655
6.153.4.17 operator<() . . . 656
6.153.4.18 operator<<() . . . 656
6.153.4.19 operator==() . . . 656
6.153.4.20 serialize() . . . 657
6.153.4.21 serializeElementToString() . . . 657
6.153.4.22 setPreparedStatementParameters() . . . 657
6.153.4.23 setValue() [1/2] . . . 657
6.153.4.24 setValue() [2/2] . . . 658
6.154 minerule::SourceRowAttributeCollection Class Reference . . . 658
6.154.1 Detailed Description . . . 659
6.154.2 Member Typedef Documentation . . . 659
6.154.2.1 ElementType . . . 659
6.154.3 Constructor & Destructor Documentation . . . 659
6.154.3.1 SourceRowAttributeCollection() [1/3] . . . 659
6.154.3.2 SourceRowAttributeCollection() [2/3] . . . 660
6.154.3.3 SourceRowAttributeCollection() [3/3] . . . 660
6.154.3.4 ~SourceRowAttributeCollection() . . . 660
6.154.4 Member Function Documentation . . . 660
6.154.4.1 asString() . . . 661
6.154.4.2 copy() . . . 661
6.154.4.3 createElement() . . . 661
6.154.4.4 createElementFromType() . . . 662
6.154.4.5 deserialize() . . . 662
6.154.4.6 deserializeElementFromResultSet() . . . 663
6.154.4.7 deserializeElementFromString() . . . 663
6.154.4.8 empty() . . . 663
6.154.4.9 getElementType() . . . 663
6.154.4.10 getFullElementType() . . . 664
6.154.4.11 getSQLData() . . . 664
6.154.4.12 operator"!=() . . . 664

6.154.4.13 operator()() . . . 665
6.154.4.14 operator< () . . . 665
6.154.4.15 operator<< () . . . 665
6.154.4.16 operator=() . . . 666
6.154.4.17 operator==() . . . 666
6.154.4.18 serialize() . . . 667
6.154.4.19 serializeElementToString() . . . 667
6.154.4.20 setPreparedStatementParameters() . . . 667
6.155 minerule::SourceRowColumnIds Class Reference . . . 668
6.155.1 Detailed Description . . . 668
6.155.2 Constructor & Destructor Documentation . . . 668
6.155.2.1 SourceRowColumnIds() [1/2] . . . 668
6.155.2.2 SourceRowColumnIds() [2/2] . . . 669
6.155.2.3 ∼SourceRowColumnIds() . . . 669
6.155.3 Member Function Documentation . . . 669
6.155.3.1 setBodyElems() . . . 669
6.155.3.2 setClusterBodyElems() . . . 669
6.155.3.3 setClusterHeadElems() . . . 670
6.155.3.4 setGroupElems() . . . 670
6.155.3.5 setHeadElems() . . . 671
6.155.4 Field Documentation . . . 671
6.155.4.1 bodyElems . . . 671
6.155.4.2 clusterBodyElems . . . 671
6.155.4.3 clusterHeadElems . . . 672
6.155.4.4 groupElems . . . 672
6.155.4.5 headElems . . . 672
6.156 minerule::SourceRowElement Class Reference . . . 672
6.156.1 Detailed Description . . . 673
6.156.2 Member Typedef Documentation . . . 673
6.156.2.1 ElementType . . . 673
6.156.3 Constructor & Destructor Documentation . . . 673
6.156.3.1 ∼SourceRowElement() . . . 674
6.156.4 Member Function Documentation . . . 674
6.156.4.1 asString() . . . 674
6.156.4.2 copy() . . . 674
6.156.4.3 createElement() . . . 674
6.156.4.4 createElementFromType() . . . 675
6.156.4.5 deserialize() . . . 675
6.156.4.6 deserializeElementFromResultSet() . . . 675
6.156.4.7 deserializeElementFromString() . . . 676
6.156.4.8 empty() . . . 676
6.156.4.9 getElementType() . . . 676

6.156.4.10 getFullElementType() . . . 676
6.156.4.11 getSQLData() . . . 677
6.156.4.12 operator"!=() . . . 677
6.156.4.13 operator()() . . . 677
6.156.4.14 operator< () . . . 677
6.156.4.15 operator<< () . . . 677
6.156.4.16 operator=() . . . 678
6.156.4.17 operator==() . . . 678
6.156.4.18 serialize() . . . 678
6.156.4.19 serializeElementToString() . . . 678
6.156.4.20 setPreparedStatementParameters() . . . 678
6.157 minerule::SourceRowEmptyElement Class Reference . . . 679
6.157.1 Detailed Description . . . 679
6.157.2 Member Typedef Documentation . . . 680
6.157.2.1 ElementType . . . 680
6.157.3 Constructor & Destructor Documentation . . . 680
6.157.3.1 SourceRowEmptyElement() . . . 680
6.157.3.2 ∼SourceRowEmptyElement() . . . 680
6.157.4 Member Function Documentation . . . 680
6.157.4.1 asString() . . . 680
6.157.4.2 copy() . . . 681
6.157.4.3 createElement() . . . 681
6.157.4.4 createElementFromType() . . . 681
6.157.4.5 deserialize() . . . 682
6.157.4.6 deserializeElementFromResultSet() . . . 682
6.157.4.7 deserializeElementFromString() . . . 682
6.157.4.8 empty() . . . 683
6.157.4.9 getElementType() . . . 683
6.157.4.10 getFullElementType() . . . 683
6.157.4.11 getSQLData() . . . 683
6.157.4.12 operator"!=() . . . 683
6.157.4.13 operator()() . . . 684
6.157.4.14 operator< () . . . 684
6.157.4.15 operator<< () . . . 684
6.157.4.16 operator=() . . . 685
6.157.4.17 operator==() . . . 685
6.157.4.18 serialize() . . . 685
6.157.4.19 serializeElementToString() . . . 685
6.157.4.20 setPreparedStatementParameters() . . . 686
6.158 minerule::SourceRowMetaInfo Class Reference . . . 686
6.158.1 Detailed Description . . . 686
6.158.2 Constructor & Destructor Documentation . . . 686

6.158.2.1 SourceRowMetaInfo() [1/2] . . . 686
6.158.2.2 SourceRowMetaInfo() [2/2] . . . 687
6.158.3 Member Function Documentation . . . 687
6.158.3.1 getBody() . . . 687
6.158.3.2 getClusterBody() . . . 687
6.158.3.3 getClusterHead() . . . 688
6.158.3.4 getGroup() . . . 688
6.158.3.5 getHead() . . . 688
6.159 minerule::SourceTable Class Reference . . . 688
6.159.1 Detailed Description . . . 689
6.159.2 Member Enumeration Documentation . . . 689
6.159.2.1 IteratorKind . . . 689
6.159.3 Constructor & Destructor Documentation . . . 689
6.159.3.1 SourceTable() [1/2] . . . 689
6.159.3.2 SourceTable() [2/2] . . . 690
6.159.3.3 ∼SourceTable() . . . 690
6.159.4 Member Function Documentation . . . 690
6.159.4.1 getTotGroups() . . . 690
6.159.4.2 init() . . . 690
6.159.4.3 newIterator() . . . 691
6.159.4.4 usesCrossProduct() . . . 691
6.160 minerule::SourceTableRequirements Class Reference . . . 691
6.160.1 Detailed Description . . . 692
6.160.2 Member Typedef Documentation . . . 692
6.160.2.1 RequirementsSet . . . 692
6.160.3 Member Enumeration Documentation . . . 692
6.160.3.1 Requirements . . . 692
6.160.4 Constructor & Destructor Documentation . . . 692
6.160.4.1 SourceTableRequirements() [1/2] . . . 693
6.160.4.2 SourceTableRequirements() [2/2] . . . 693
6.160.4.3 ∼SourceTableRequirements() . . . 693
6.160.5 Member Function Documentation . . . 693
6.160.5.1 crossProduct() . . . 693
6.160.5.2 set() . . . 693
6.160.5.3 sortedGids() . . . 694
6.161 mrdb::SQLException Class Reference . . . 694
6.161.1 Detailed Description . . . 694
6.161.2 Constructor & Destructor Documentation . . . 694
6.161.2.1 SQLException() . . . 694
6.161.2.2 ∼SQLException() . . . 695
6.162 minerule::SQLUtils Class Reference . . . 695
6.162.1 Detailed Description . . . 695

6.162.2 Member Enumeration Documentation . . . 695
6.162.2.1 Type . . . 695
6.162.3 Member Function Documentation . . . 696
6.162.3.1 getType() [1/3] . . . 696
6.162.3.2 getType() [2/3] . . . 696
6.162.3.3 getType() [3/3] . . . 697
6.162.3.4 isAttribute() . . . 697
6.162.3.5 isBinaryType() . . . 697
6.162.3.6 isBitType() . . . 698
6.162.3.7 isDateTimeType() . . . 698
6.162.3.8 isNumericType() . . . 698
6.162.3.9 isStringType() . . . 698
6.162.3.10 quote() . . . 699
6.162.3.11 removeHeadBodyFromAttrName() . . . 699
6.163 mrdb::Statement Class Reference . . . 699
6.163.1 Detailed Description . . . 700
6.163.2 Constructor & Destructor Documentation . . . 700
6.163.2.1 ~Statement() . . . 700
6.163.3 Member Function Documentation . . . 700
6.163.3.1 execute() . . . 700
6.163.3.2 executeQuery() . . . 700
6.164 minerule::StringCompare Class Reference . . . 701
6.164.1 Detailed Description . . . 701
6.164.2 Member Function Documentation . . . 701
6.164.2.1 operator()() . . . 701
6.165 minerule::StringUtils Class Reference . . . 701
6.165.1 Detailed Description . . . 702
6.165.2 Member Function Documentation . . . 702
6.165.2.1 join() . . . 702
6.165.2.2 setColorsEnabled() . . . 703
6.165.2.3 split() . . . 703
6.165.2.4 splitToLength() . . . 703
6.165.2.5 toBlue() . . . 704
6.165.2.6 toBold() . . . 704
6.165.2.7 toBoldRed() . . . 704
6.165.2.8 toGreen() . . . 704
6.165.2.9 toOrange() . . . 704
6.165.2.10 toRed() . . . 705
6.165.2.11 toWhite() . . . 705
6.165.2.12 toYellow() . . . 705
6.165.3 Field Documentation . . . 705
6.165.3.1 BLUE . . . 705

6.165.3.2 BOLD . . . 705
6.165.3.3 CLOSE . . . 706
6.165.3.4 GREEN . . . 706
6.165.3.5 RED . . . 706
6.165.3.6 WHITE . . . 706
6.165.3.7 YELLOW . . . 706
6.166 minerule::STSMiner Class Reference . . . 707
6.166.1 Detailed Description . . . 707
6.166.2 Constructor & Destructor Documentation . . . 707
6.166.2.1 STSMiner() . . . 708
6.166.2.2 ∼STSMiner() . . . 709
6.166.3 Member Function Documentation . . . 709
6.166.3.1 algorithmForType() . . . 709
6.166.3.2 canHandleMinerule() . . . 710
6.166.3.3 execute() . . . 710
6.166.3.4 initialize() . . . 710
6.166.3.5 mineRules() . . . 711
6.166.3.6 optimizedMinerule() . . . 712
6.166.3.7 sourceTableRequirements() . . . 712
6.166.4 Field Documentation . . . 712
6.166.4.1 connection . . . 712
6.166.4.2 minerule . . . 713
6.167 minerule::QueryNormalizer::SubstEntryBody Class Reference . . . 713
6.167.1 Detailed Description . . . 713
6.167.2 Constructor & Destructor Documentation . . . 713
6.167.2.1 SubstEntryBody() [1/2] . . . 713
6.167.2.2 SubstEntryBody() [2/2] . . . 714
6.167.3 Member Function Documentation . . . 714
6.167.3.1 operator<() . . . 714
6.167.4 Field Documentation . . . 714
6.167.4.1 pos . . . 714
6.167.4.2 pred . . . 714
6.168 minerule::QueryNormalizer::SubstEntryHead Class Reference . . . 714
6.168.1 Detailed Description . . . 715
6.168.2 Constructor & Destructor Documentation . . . 715
6.168.2.1 SubstEntryHead() [1/2] . . . 715
6.168.2.2 SubstEntryHead() [2/2] . . . 715
6.168.3 Member Function Documentation . . . 715
6.168.3.1 operator<() . . . 716
6.168.4 Field Documentation . . . 716
6.168.4.1 elems . . . 716
6.168.4.2 refKey . . . 716

6.169 minerule::SumGreaterEq Class Reference . . . 716
6.169.1 Detailed Description . . . 717
6.169.2 Constructor & Destructor Documentation . . . 717
6.169.2.1 SumGreaterEq() [1/3] . . . 717
6.169.2.2 SumGreaterEq() [2/3] . . . 717
6.169.2.3 SumGreaterEq() [3/3] . . . 717
6.169.3 Member Function Documentation . . . 718
6.169.3.1 check() [1/2] . . . 718
6.169.3.2 check() [2/2] . . . 718
6.169.4 Field Documentation . . . 718
6.169.4.1 attributeIndex . . . 718
6.170 minerule::SumGreaterThan Class Reference . . . 719
6.170.1 Detailed Description . . . 719
6.170.2 Constructor & Destructor Documentation . . . 719
6.170.2.1 SumGreaterThan() [1/3] . . . 719
6.170.2.2 SumGreaterThan() [2/3] . . . 720
6.170.2.3 SumGreaterThan() [3/3] . . . 720
6.170.3 Member Function Documentation . . . 720
6.170.3.1 check() [1/2] . . . 720
6.170.3.2 check() [2/2] . . . 720
6.170.4 Field Documentation . . . 721
6.170.4.1 attributeIndex . . . 721
6.171 minerule::SumLessEq Class Reference . . . 721
6.171.1 Detailed Description . . . 721
6.171.2 Constructor & Destructor Documentation . . . 722
6.171.2.1 SumLessEq() [1/3] . . . 722
6.171.2.2 SumLessEq() [2/3] . . . 722
6.171.2.3 SumLessEq() [3/3] . . . 722
6.171.3 Member Function Documentation . . . 722
6.171.3.1 check() [1/2] . . . 722
6.171.3.2 check() [2/2] . . . 723
6.171.4 Field Documentation . . . 723
6.171.4.1 attributeIndex . . . 723
6.172 minerule::SumLessThan Class Reference . . . 723
6.172.1 Detailed Description . . . 724
6.172.2 Constructor & Destructor Documentation . . . 724
6.172.2.1 SumLessThan() [1/3] . . . 724
6.172.2.2 SumLessThan() [2/3] . . . 724
6.172.2.3 SumLessThan() [3/3] . . . 724
6.172.3 Member Function Documentation . . . 724
6.172.3.1 check() [1/2] . . . 725
6.172.3.2 check() [2/2] . . . 725

6.172.4 Field Documentation . . . 725
6.172.4.1 attributeIndex . . . 725
6.173 minerule::SupportMeasure Class Reference . . . 726
6.173.1 Detailed Description . . . 726
6.173.2 Member Enumeration Documentation . . . 726
6.173.2.1 MeasureType . . . 726
6.173.2.2 RelOperator . . . 727
6.173.3 Constructor & Destructor Documentation . . . 727
6.173.3.1 SupportMeasure() . . . 727
6.173.4 Member Function Documentation . . . 727
6.173.4.1 evalSupport() . . . 727
6.173.4.2 getMeasureType() . . . 728
6.173.4.3 getThreshold() . . . 728
6.174 minerule::TimeOutException Class Reference . . . 728
6.174.1 Detailed Description . . . 729
6.175 minerule::Transaction Class Reference . . . 729
6.175.1 Detailed Description . . . 729
6.175.2 Member Function Documentation . . . 729
6.175.2.1 findGid() . . . 730
6.175.2.2 loadBody() . . . 730
6.175.2.3 loadHead() . . . 730
6.175.3 Field Documentation . . . 730
6.175.3.1 keys . . . 730
6.175.3.2 values . . . 731
6.176 minerule::TransactionBase< SetType > Class Template Reference . . . 731
6.176.1 Detailed Description . . . 731
6.176.2 Constructor & Destructor Documentation . . . 731
6.176.2.1 TransactionBase() . . . 731
6.176.3 Member Function Documentation . . . 732
6.176.3.1 findGid() . . . 732
6.177 minerule::TranslatedTable Class Reference . . . 732
6.177.1 Detailed Description . . . 732
6.177.2 Member Function Documentation . . . 733
6.177.2.1 getOriginalName() . . . 733
6.177.2.2 getOriginalValue() . . . 733
6.177.2.3 getTranslatedColumnName() . . . 733
6.177.2.4 getTranslatedName() . . . 734
6.177.2.5 getTranslatedValue() . . . 734
6.178 minerule::TranslatedTableStandardSQL Class Reference . . . 734
6.178.1 Detailed Description . . . 735
6.178.2 Constructor & Destructor Documentation . . . 735
6.178.2.1 TranslatedTableStandardSQL() . . . 735

6.178.2.2 ~TranslatedTableStandardSQL() . . . 735
6.178.3 Member Function Documentation . . . 736
6.178.3.1 getOriginalName() . . . 736
6.178.3.2 getOriginalValue() . . . 736
6.178.3.3 getTranslatedColumnName() . . . 736
6.178.3.4 getTranslatedName() . . . 737
6.178.3.5 getTranslatedValue() . . . 737
6.178.3.6 setOriginalTableName() . . . 737
6.178.3.7 setTranslatedTableName() . . . 738
6.178.3.8 updateColumnsToTranslate() . . . 738
6.178.4 Friends And Related Function Documentation . . . 738
6.178.4.1 TranslationManagerStandardSQL . . . 738
6.179 minerule::TranslationManager Class Reference . . . 738
6.179.1 Detailed Description . . . 739
6.179.2 Member Function Documentation . . . 739
6.179.2.1 alreadyTranslated() . . . 739
6.179.2.2 getOriginalColumnNameForColumn() . . . 740
6.179.2.3 getTranslatedColumnNameForColumn() . . . 740
6.179.2.4 getTranslationTableNameForColumn() . . . 740
6.179.2.5 getTranslationTableNameForTable() . . . 741
6.179.2.6 translateTable() . . . 741
6.180 minerule::TranslationManagerStandardSQL Class Reference . . . 741
6.180.1 Detailed Description . . . 742
6.180.2 Constructor & Destructor Documentation . . . 742
6.180.2.1 TranslationManagerStandardSQL() . . . 742
6.180.3 Member Function Documentation . . . 742
6.180.3.1 alreadyTranslated() . . . 743
6.180.3.2 getOriginalColumnNameForColumn() . . . 743
6.180.3.3 getTranslatedColumnNameForColumn() . . . 743
6.180.3.4 getTranslationTableNameForColumn() . . . 743
6.180.3.5 getTranslationTableNameForTable() . . . 744
6.180.3.6 translateTable() . . . 744
6.180.4 Friends And Related Function Documentation . . . 744
6.180.4.1 TranslatedTableStandardSQL . . . 744
6.181 mrdb::Types Struct Reference . . . 745
6.181.1 Detailed Description . . . 745
6.181.2 Member Enumeration Documentation . . . 745
6.181.2.1 SQLType . . . 745
6.182 minerule::VarSet Class Reference . . . 746
6.182.1 Detailed Description . . . 746
6.182.2 Constructor & Destructor Documentation . . . 746
6.182.2.1 VarSet() [1/2] . . . 747

6.182.2.2 VarSet() [2/2] . . . 747
6.182.3 Member Function Documentation . . . 747
6.182.3.1 getVar() . . . 747
6.182.3.2 operator++() . . . 747
6.182.3.3 operator<() . . . 748
6.182.3.4 operator=() . . . 748
6.182.3.5 operator==() . . . 748
6.182.3.6 operator>() . . . 748
6.182.3.7 setSize() . . . 749
6.182.3.8 setVar() . . . 749
6.182.3.9 size() . . . 749
6.183 minerule::VarSetEnumerator Class Reference . . . 749
6.183.1 Detailed Description . . . 750
6.183.2 Constructor & Destructor Documentation . . . 750
6.183.2.1 VarSetEnumerator() . . . 750
6.183.3 Member Function Documentation . . . 750
6.183.3.1 enumerationSize() . . . 750
6.183.3.2 operator*() . . . 750
6.183.3.3 operator++() . . . 750

7 File Documentation **751**

7.1 /Users/esposito/Software/minerule/include/minerule/Algorithms/Algorithms.hpp File Reference . . . 751
7.2 Algorithms.hpp . . . 751
7.3 /Users/esposito/Software/minerule/include/minerule/Algorithms/AlgorithmsOptions.hpp File Reference 752
7.4 AlgorithmsOptions.hpp . . . 753
7.5 /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsAndCross.hpp File Reference . . . 754
7.6 BFSWithGidsAndCross.hpp . . . 754
7.7 /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsNoCross.hpp File Reference . . . 756
7.8 BFSWithGidsNoCross.hpp . . . 757
7.9 /Users/esposito/Software/minerule/include/minerule/Algorithms/BitVector.hpp File Reference . . . 759
7.10 BitVector.hpp . . . 759
7.11 /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp File Reference . . . 760
7.12 Bodymap.hpp . . . 761
7.13 /Users/esposito/Software/minerule/include/minerule/Algorithms/Care.hpp File Reference . . . 763
7.14 Care.hpp . . . 763
7.15 /Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMiner.hpp File Reference . . . 764
7.16 CCSMiner.hpp . . . 765
7.17 /Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMSequence.hpp File Reference 765
7.17.1 Macro Definition Documentation . . . 766
7.17.1.1 ABSENT . . . 766
7.17.1.2 PRESENT . . . 766

7.18 CCSMSequence.hpp . . . 767
7.19 /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrItemSetsExtraction.hpp File Reference . . . 771
7.20 ConstrItemSetsExtraction.hpp . . . 771
7.21 /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrTree.hpp File Reference . . . 772
7.22 ConstrTree.hpp . . . 772
7.23 /Users/esposito/Software/minerule/include/minerule/Algorithms/DestrTree.hpp File Reference . . . 773
7.24 DestrTree.hpp . . . 774
7.25 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSMiner.hpp File Reference . . . 775
7.26 FSMiner.hpp . . . 775
7.27 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTree.hpp File Reference . . . 776
7.28 FSTree.hpp . . . 776
7.29 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeNode.hpp File Reference . . . 778
7.30 FSTreeNode.hpp . . . 778
7.31 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeSequence.hpp File Reference 779
7.32 FSTreeSequence.hpp . . . 779
7.33 /Users/esposito/Software/minerule/include/minerule/Algorithms/IDIncrementalAlgorithm.hpp File Reference . . . 780
7.34 IDIncrementalAlgorithm.hpp . . . 781
7.35 /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp File Reference 781
7.36 IncrAlgoClasses.hpp . . . 782
7.37 /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrementalAlgorithm.hpp File Reference . . . 784
7.38 IncrementalAlgorithm.hpp . . . 784
7.39 /Users/esposito/Software/minerule/include/minerule/Algorithms/MiningAlgorithmBase.hpp File Reference . . . 785
7.40 MiningAlgorithmBase.hpp . . . 785
7.41 /Users/esposito/Software/minerule/include/minerule/Algorithms/ResultCombinator.hpp File Reference 786
7.42 ResultCombinator.hpp . . . 787
7.43 /Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp File Reference . . . 787
7.44 STSMiner.hpp . . . 788
7.45 /Users/esposito/Software/minerule/include/minerule/Database/Connection.hpp File Reference . . . 797
7.46 Connection.hpp . . . 797
7.47 /Users/esposito/Software/minerule/include/minerule/mrdb/Connection.hpp File Reference . . . 799
7.47.1 Typedef Documentation . . . 800
7.47.1.1 MRDBConnectFun . . . 800
7.48 Connection.hpp . . . 800
7.49 /Users/esposito/Software/minerule/include/minerule/Database/ItemType.hpp File Reference . . . 801
7.50 ItemType.hpp . . . 801
7.51 /Users/esposito/Software/minerule/include/minerule/Database/MRResultSet.hpp File Reference . . . 804
7.52 MRResultSet.hpp . . . 804
7.53 /Users/esposito/Software/minerule/include/minerule/Database/PrepareDataUtils.hpp File Reference 805
7.54 PrepareDataUtils.hpp . . . 805

7.55 /Users/esposito/Software/minerule/include/minerule/Database/SourceRow.hpp File Reference . . . 806
7.55.1 Macro Definition Documentation . . . 807
7.55.1.1 ELEM_OR_EMPTY . . . 807
7.56 SourceRow.hpp . . . 807
7.57 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp File Reference 808
7.58 SourceRowAttribute.hpp . . . 809
7.59 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttributeCollection.hpp File Reference . . . 811
7.60 SourceRowAttributeCollection.hpp . . . 812
7.61 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowColumnIds.hpp File Reference . . . 813
7.62 SourceRowColumnIds.hpp . . . 814
7.63 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowElement.hpp File Reference 815
7.64 SourceRowElement.hpp . . . 815
7.65 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowMetaInfo.hpp File Reference 818
7.66 SourceRowMetaInfo.hpp . . . 818
7.67 /Users/esposito/Software/minerule/include/minerule/Database/SourceTable.hpp File Reference . . . 819
7.68 SourceTable.hpp . . . 820
7.69 /Users/esposito/Software/minerule/include/minerule/Database/SourceTableRequirements.hpp File Reference . . . 821
7.70 SourceTableRequirements.hpp . . . 821
7.71 /Users/esposito/Software/minerule/include/minerule/Database/Transaction.hpp File Reference . . . 822
7.72 Transaction.hpp . . . 822
7.73 /Users/esposito/Software/minerule/include/minerule/mrdb/DatabaseMetaData.hpp File Reference . 823
7.74 DatabaseMetaData.hpp . . . 824
7.75 /Users/esposito/Software/minerule/include/minerule/mrdb/PreparedStatement.hpp File Reference . 824
7.76 PreparedStatement.hpp . . . 825
7.77 /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSet.hpp File Reference . . . 825
7.78 ResultSet.hpp . . . 825
7.79 /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSetMetaData.hpp File Reference . 826
7.80 ResultSetMetaData.hpp . . . 826
7.81 /Users/esposito/Software/minerule/include/minerule/mrdb/SQLException.hpp File Reference . . . 827
7.81.1 Macro Definition Documentation . . . 827
7.81.1.1 _NOEXCEPT . . . 827
7.82 SQLException.hpp . . . 827
7.83 /Users/esposito/Software/minerule/include/minerule/mrdb/Statement.hpp File Reference . . . 828
7.84 Statement.hpp . . . 828
7.85 /Users/esposito/Software/minerule/include/minerule/mrdb/Types.hpp File Reference . . . 828
7.86 Types.hpp . . . 829
7.87 /Users/esposito/Software/minerule/include/minerule/Optimizer/CatalogueInstaller.hpp File Reference 829
7.88 CatalogueInstaller.hpp . . . 829
7.89 /Users/esposito/Software/minerule/include/minerule/Optimizer/GAQueryCombinator.hpp File Reference . . . 830

7.90 GAQueryCombinator.hpp . . . 831
7.91 /Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/MySqlCatalogueInstaller.hpp File Reference . . . 832
7.92 MySqlCatalogueInstaller.hpp . . . 832
7.93 /Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp File Reference . . . 833
7.94 PostgresCatalogueInstaller.hpp . . . 833
7.95 /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizedMinerule.hpp File Reference 834
7.96 OptimizedMinerule.hpp . . . 834
7.97 /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp File Reference 836
7.98 OptimizerCatalogue.hpp . . . 836
7.99 /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp File Reference 839
7.100 QueryNormalizer.hpp . . . 839
7.101 /Users/esposito/Software/minerule/include/minerule/Parsers/EvaluationMeasure.hpp File Reference 842
7.102 EvaluationMeasure.hpp . . . 842
7.103 /Users/esposito/Software/minerule/include/minerule/Parsers/HeaderQuery.hpp File Reference . . . 843
7.104 HeaderQuery.hpp . . . 843
7.105 /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp File Reference . 844
7.105.1 Typedef Documentation . . . 845
7.105.1.1 list_AND_node . . . 845
7.105.1.2 list_OR_node . . . 845
7.105.1.3 mineruletag . . . 845
7.105.1.4 simple_pred . . . 845
7.105.2 Function Documentation . . . 845
7.105.2.1 clone_l_AND() . . . 846
7.105.2.2 clone_l_OR() . . . 846
7.106 ParsedMinerule.hpp . . . 846
7.107 /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedPredicate.hpp File Reference . 850
7.108 ParsedPredicate.hpp . . . 850
7.109 /Users/esposito/Software/minerule/include/minerule/Parsers/ParserLibrary.hpp File Reference . . . 852
7.110 ParserLibrary.hpp . . . 853
7.111 /Users/esposito/Software/minerule/include/minerule/Parsers/PredicateBase.hpp File Reference . . 853
7.112 PredicateBase.hpp . . . 853
7.113 /Users/esposito/Software/minerule/include/minerule/Parsers/SupportMeasure.hpp File Reference . 856
7.114 SupportMeasure.hpp . . . 856
7.115 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp File Reference . . . 857
7.116 ExpressionNFCoder.hpp . . . 858
7.117 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp File Reference . . . 859
7.118 HeadBodyPredicatesSeparator.hpp . . . 860
7.119 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp File Reference . . 860
7.120 Interval.hpp . . . 861

7.121 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/InvalidConfigurationFilter.hpp File Reference . . . 863
7.122 InvalidConfigurationFilter.hpp . . . 863
7.123 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp File Reference . . . 864
7.124 Predicate.hpp . . . 864
7.125 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/PredicateUtils.hpp File Reference 867
7.126 PredicateUtils.hpp . . . 867
7.127 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/SimplePredAnalyzer.hpp File Reference . . . 869
7.128 SimplePredAnalyzer.hpp . . . 869
7.129 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/VarSet.hpp File Reference . . . 870
7.130 VarSet.hpp . . . 871
7.131 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtable.hpp File Reference 873
7.132 translatedtable.hpp . . . 873
7.133 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtablestandardsql.hpp File Reference . . . 874
7.133.1 Macro Definition Documentation . . . 874
7.133.1.1 __TRANSLATEDTABLESTANDARDSQL_H_ . . . 874
7.134 translatedtablestandardsql.hpp . . . 874
7.135 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanager.hpp File Reference . . . 876
7.135.1 Macro Definition Documentation . . . 876
7.135.1.1 BBLOG . . . 876
7.136 translationmanager.hpp . . . 877
7.137 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanagerstandardsql.hpp File Reference . . . 877
7.138 translationmanagerstandardsql.hpp . . . 878
7.139 /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp File Reference 879
7.140 QueryResult-header.hpp . . . 880
7.141 /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-impl.hpp File Reference . . . 884
7.142 QueryResult-impl.hpp . . . 885
7.143 /Users/esposito/Software/minerule/include/minerule/Result/QueryResult.hpp File Reference . . . 886
7.144 QueryResult.hpp . . . 886
7.145 /Users/esposito/Software/minerule/include/minerule/Result/Rule.hpp File Reference . . . 887
7.146 Rule.hpp . . . 887
7.147 /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp File Reference . . . 889
7.148 RuleFormatter.hpp . . . 889
7.149 /Users/esposito/Software/minerule/include/minerule/Result/RulesMatcher.hpp File Reference . . . 891
7.150 RulesMatcher.hpp . . . 891
7.151 /Users/esposito/Software/minerule/include/minerule/Utils/AlgorithmTypes.hpp File Reference . . . 892
7.152 AlgorithmTypes.hpp . . . 892
7.153 /Users/esposito/Software/minerule/include/minerule/Utils/Bitstring.hpp File Reference . . . 893
7.153.1 Macro Definition Documentation . . . 894

7.153.1.1 CHAR_BIT . . . 894
7.154 Bitstring.hpp . . . 894
7.155 /Users/esposito/Software/minerule/include/minerule/Utils/common.hpp File Reference . . . 895
7.156 common.hpp . . . 895
7.157 /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp File Reference . . . 896
7.158 Constraints.hpp . . . 896
7.159 /Users/esposito/Software/minerule/include/minerule/Utils/Converter.hpp File Reference . . . 898
7.160 Converter.hpp . . . 898
7.161 /Users/esposito/Software/minerule/include/minerule/Utils/FileUtils.hpp File Reference . . . 900
7.162 FileUtils.hpp . . . 900
7.163 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleErrors.hpp File Reference . . . 901
7.164 MineruleErrors.hpp . . . 901
7.165 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleException.hpp File Reference . . . 902
7.165.1 Macro Definition Documentation . . . 902
7.165.1.1 _NOEXCEPT . . . 902
7.165.1.2 MineruleException . . . 903
7.166 MineruleException.hpp . . . 903
7.167 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp File Reference . . . 903
7.168 MineruleLogs.hpp . . . 904
7.169 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions.hpp File Reference . . . 906
7.170 MineruleOptions.hpp . . . 906
7.171 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms.hpp File Reference . . . 907
7.172 miningalgorithms.hpp . . . 907
7.173 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/itemsetminin File Reference . . . 908
7.174 itemsetmining.hpp . . . 908
7.175 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.h File Reference . . . 909
7.176 rulemining.hpp . . . 909
7.177 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/sequencemi File Reference . . . 911
7.178 sequencemining.hpp . . . 911
7.179 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/mrdb.hpp File Reference . . . 912
7.180 mrdb.hpp . . . 912
7.181 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/optimizations.hpp File Reference . . . 914
7.182 optimizations.hpp . . . 914
7.183 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/outstream.hpp File Reference . . . 916
7.184 outstream.hpp . . . 916
7.185 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/parsers.hpp File Reference . . . 917

7.186 parsers.hpp . . . 917
7.187 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/root.hpp File Reference . . . 918
7.188 root.hpp . . . 918
7.189 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/safety.hpp File Reference . . . 922
7.190 safety.hpp . . . 922
7.191 /Users/esposito/Software/minerule/include/minerule/Utils/MinMaxPair.hpp File Reference . . . 923
7.192 MinMaxPair.hpp . . . 923
7.193 /Users/esposito/Software/minerule/include/minerule/Utils/MRLogger.hpp File Reference . . . 924
7.194 MRLogger.hpp . . . 925
7.195 /Users/esposito/Software/minerule/include/minerule/Utils/OptionParserLib.hpp File Reference . . . 927
7.196 OptionParserLib.hpp . . . 927
7.197 /Users/esposito/Software/minerule/include/minerule/Utils/Progress.hpp File Reference . . . 928
7.198 Progress.hpp . . . 928
7.199 /Users/esposito/Software/minerule/include/minerule/Utils/SQLUtils.hpp File Reference . . . 929
7.200 SQLUtils.hpp . . . 929
7.201 /Users/esposito/Software/minerule/include/minerule/Utils/StringUtils.hpp File Reference . . . 931
7.202 StringUtils.hpp . . . 931
7.203 /Users/esposito/Software/minerule/src/Algorithms/Algorithms.cpp File Reference . . . 932
7.204 Algorithms.cpp . . . 932
7.205 /Users/esposito/Software/minerule/src/Algorithms/AlgorithmsOptions.cpp File Reference . . . 936
7.206 AlgorithmsOptions.cpp . . . 936
7.207 /Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsAndCross.cpp File Reference . . . 937
7.208 BFSWithGidsAndCross.cpp . . . 937
7.209 /Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsNoCross.cpp File Reference . . . 941
7.210 BFSWithGidsNoCross.cpp . . . 942
7.211 /Users/esposito/Software/minerule/src/Algorithms/Bodymap.cpp File Reference . . . 946
7.212 Bodymap.cpp . . . 946
7.213 /Users/esposito/Software/minerule/src/Algorithms/Care.cpp File Reference . . . 952
7.213.1 Macro Definition Documentation . . . 952
7.213.1.1 MROPTIONFILE . . . 952
7.214 Care.cpp . . . 952
7.215 /Users/esposito/Software/minerule/src/Algorithms/CCSMiner.cpp File Reference . . . 954
7.216 CCSMiner.cpp . . . 954
7.217 /Users/esposito/Software/minerule/src/Algorithms/CCSMSequence.cpp File Reference . . . 960
7.218 CCSMSequence.cpp . . . 961
7.219 /Users/esposito/Software/minerule/src/Algorithms/ConstrItemSetsExtraction.cpp File Reference . . . 964
7.219.1 Macro Definition Documentation . . . 964
7.219.1.1 MROPTIONFILE . . . 964
7.220 ConstrItemSetsExtraction.cpp . . . 964
7.221 /Users/esposito/Software/minerule/src/Algorithms/ConstrTree.cpp File Reference . . . 966
7.222 ConstrTree.cpp . . . 966

7.223 /Users/esposito/Software/minerule/src/Algorithms/DestrTree.cpp File Reference 969
7.224 DestrTree.cpp 969
7.225 /Users/esposito/Software/minerule/src/Algorithms/FSMiner.cpp File Reference 973
7.226 FSMiner.cpp 974
7.227 /Users/esposito/Software/minerule/src/Algorithms/FSTree.cpp File Reference 975
7.228 FSTree.cpp 975
7.229 /Users/esposito/Software/minerule/src/Algorithms/FSTreeNode.cpp File Reference 980
7.230 FSTreeNode.cpp 980
7.231 /Users/esposito/Software/minerule/src/Algorithms/FSTreeSequence.cpp File Reference 981
7.232 FSTreeSequence.cpp 981
7.233 /Users/esposito/Software/minerule/src/Algorithms/IDIncrementalAlgorithm.cpp File Reference . . . 983
7.234 IDIncrementalAlgorithm.cpp 984
7.235 /Users/esposito/Software/minerule/src/Algorithms/IncrAlgoClasses.cpp File Reference 986
7.236 IncrAlgoClasses.cpp 986
7.237 /Users/esposito/Software/minerule/src/Algorithms/IncrementalAlgorithm.cpp File Reference 992
7.238 IncrementalAlgorithm.cpp 993
7.239 /Users/esposito/Software/minerule/src/Algorithms/MiningAlgorithmBase.cpp File Reference 993
7.240 MiningAlgorithmBase.cpp 994
7.241 /Users/esposito/Software/minerule/src/Algorithms/ResultCombinator.cpp File Reference 994
7.242 ResultCombinator.cpp 995
7.243 /Users/esposito/Software/minerule/src/Algorithms/STSMiner.cpp File Reference 996
7.244 STSMiner.cpp 996
7.245 /Users/esposito/Software/minerule/src/Database/Connection.cpp File Reference 1015
7.246 Connection.cpp 1016
7.247 /Users/esposito/Software/minerule/src/Database/PrepareDataUtils.cpp File Reference 1022
7.247.1 Macro Definition Documentation 1022
7.247.1.1 ADDCOLALIAS [1/2] 1022
7.247.1.2 ADDCOLALIAS [2/2] 1023
7.248 PrepareDataUtils.cpp 1023
7.249 /Users/esposito/Software/minerule/src/Database/SourceRow.cpp File Reference 1027
7.249.1 Macro Definition Documentation 1027
7.249.1.1 CREATE 1027
7.249.1.2 DESTROY [1/2] 1028
7.249.1.3 DESTROY [2/2] 1028
7.250 SourceRow.cpp 1028
7.251 /Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp File Reference 1029
7.252 SourceRowAttribute.cpp 1029
7.253 /Users/esposito/Software/minerule/src/Database/SourceRowAttributeCollection.cpp File Reference 1032
7.254 SourceRowAttributeCollection.cpp 1032
7.255 /Users/esposito/Software/minerule/src/Database/SourceRowColumnIds.cpp File Reference 1036
7.256 SourceRowColumnIds.cpp 1036
7.257 /Users/esposito/Software/minerule/src/Database/SourceRowElement.cpp File Reference 1037

7.258 SourceRowElement.cpp . . . 1037
7.259 /Users/esposito/Software/minerule/src/Database/SourceRowMetaInfo.cpp File Reference . . . 1038
7.260 SourceRowMetaInfo.cpp . . . 1038
7.261 /Users/esposito/Software/minerule/src/Database/SourceTable.cpp File Reference . . . 1041
7.262 SourceTable.cpp . . . 1041
7.263 /Users/esposito/Software/minerule/src/Optimizer/CatalogueInstaller.cpp File Reference . . . 1042
7.264 CatalogueInstaller.cpp . . . 1043
7.265 /Users/esposito/Software/minerule/src/Optimizer/GAQueryCombinator.cpp File Reference . . . 1043
7.265.1 Macro Definition Documentation . . . 1044
7.265.1.1 bit_vector . . . 1044
7.266 GAQueryCombinator.cpp . . . 1044
7.267 /Users/esposito/Software/minerule/src/Optimizer/Installers/PostgresCatalogueInstaller.cpp File Reference . . . 1046
7.268 PostgresCatalogueInstaller.cpp . . . 1047
7.269 /Users/esposito/Software/minerule/src/Optimizer/OptimizedMinerule.cpp File Reference . . . 1048
7.270 OptimizedMinerule.cpp . . . 1049
7.271 /Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp File Reference . . . 1054
7.272 OptimizerCatalogue.cpp . . . 1054
7.273 /Users/esposito/Software/minerule/src/Optimizer/QueryNormalizer.cpp File Reference . . . 1062
7.273.1 Macro Definition Documentation . . . 1062
7.273.1.1 HEAD . . . 1063
7.273.1.2 TAIL . . . 1063
7.274 QueryNormalizer.cpp . . . 1063
7.275 /Users/esposito/Software/minerule/src/Parsers/MineruleParser.ypp File Reference . . . 1074
7.276 MineruleParser.ypp . . . 1074
7.277 /Users/esposito/Software/minerule/src/Parsers/ParsedMinerule.cpp File Reference . . . 1086
7.277.1 Function Documentation . . . 1087
7.277.1.1 clone_l_AND() . . . 1087
7.277.1.2 clone_l_OR() . . . 1088
7.277.1.3 find_in_list() . . . 1088
7.277.1.4 init_analisi_minerule() . . . 1088
7.277.1.5 mrerror() . . . 1088
7.277.1.6 print_AND_list() . . . 1089
7.277.1.7 print_OR_list() . . . 1089
7.278 ParsedMinerule.cpp . . . 1089
7.279 /Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp File Reference . . . 1095
7.280 ParsedPredicate.cpp . . . 1095
7.281 /Users/esposito/Software/minerule/src/Parsers/ParserLibrary.cpp File Reference . . . 1096
7.281.1 Typedef Documentation . . . 1097
7.281.1.1 YY_BUFFER_STATE . . . 1097
7.281.2 Function Documentation . . . 1097
7.281.2.1 mr_delete_buffer() . . . 1097

7.281.2.2 mr_scan_string() . . . 1097
7.281.2.3 mrparse() . . . 1097
7.281.3 Variable Documentation . . . 1097
7.281.3.1 mrout . . . 1097
7.282 ParserLibrary.cpp . . . 1098
7.283 /Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp File Reference . . . 1098
7.284 PredicateBase.cpp . . . 1098
7.285 /Users/esposito/Software/minerule/src/Parsers/SupportMeasure.cpp File Reference . . . 1101
7.286 SupportMeasure.cpp . . . 1102
7.287 /Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp File Reference . . . 1102
7.288 ExpressionNFCoder.cpp . . . 1103
7.289 /Users/esposito/Software/minerule/src/PredicateUtils/HeadBodyPredicatesSeparator.cpp File Reference . . . 1107
7.290 HeadBodyPredicatesSeparator.cpp . . . 1107
7.291 /Users/esposito/Software/minerule/src/PredicateUtils/Interval.cpp File Reference . . . 1108
7.292 Interval.cpp . . . 1109
7.293 /Users/esposito/Software/minerule/src/PredicateUtils/Predicate.cpp File Reference . . . 1112
7.294 Predicate.cpp . . . 1112
7.295 /Users/esposito/Software/minerule/src/PredicateUtils/PredicateUtils.cpp File Reference . . . 1114
7.296 PredicateUtils.cpp . . . 1115
7.297 /Users/esposito/Software/minerule/src/PredicateUtils/SimplePredAnalyzer.cpp File Reference . . . 1117
7.298 SimplePredAnalyzer.cpp . . . 1117
7.299 /Users/esposito/Software/minerule/src/Programs/Minerule/minerule.cpp File Reference . . . 1119
7.299.1 Function Documentation . . . 1119
7.299.1.1 buildPath() . . . 1119
7.299.1.2 checkSubCommands() . . . 1120
7.299.1.3 execSubCommand() . . . 1120
7.299.1.4 expandSubCommand() . . . 1120
7.299.1.5 knownCommands() . . . 1121
7.299.1.6 main() . . . 1121
7.299.1.7 printMRHelp() . . . 1121
7.300 minerule.cpp . . . 1122
7.301 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/ErrorCodes.hpp File Reference . . . 1123
7.302 ErrorCodes.hpp . . . 1124
7.303 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Exception.hpp File Reference . . . 1124
7.303.1 Macro Definition Documentation . . . 1124
7.303.1.1 _NOEXCEPT . . . 1125
7.304 Exception.hpp . . . 1125
7.305 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/mrcatalogue.cpp File Reference . . . 1125
7.305.1 Function Documentation . . . 1126
7.305.1.1 main() . . . 1126
7.306 mrcatalogue.cpp . . . 1126

7.307 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/MRCUtils.cpp File Reference . . . 1127
7.308 MRCUtils.cpp . . . 1128
7.309 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/MRCUtils.hpp File Reference . . . 1132
7.310 MRCUtils.hpp . . . 1132
7.311 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.cpp File Reference . . . 1133
7.312 Printer.cpp . . . 1133
7.313 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.hpp File Reference . . . 1134
7.314 Printer.hpp . . . 1134
7.315 /Users/esposito/Software/minerule/src/Programs/MRDefaultOptions/MRDefaultOptions.cpp File Reference . . . 1135
7.315.1 Function Documentation . . . 1135
7.315.1.1 main() . . . 1135
7.315.1.2 parseOptions() . . . 1136
7.315.1.3 printHelp() . . . 1136
7.315.1.4 printVersion() . . . 1137
7.316 MRDefaultOptions.cpp . . . 1137
7.317 /Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.cpp File Reference . 1138
7.318 GidRulesMatcher.cpp . . . 1138
7.319 /Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.hpp File Reference . 1139
7.320 GidRulesMatcher.hpp . . . 1140
7.321 /Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.cpp File Reference . . . 1140
7.322 Matcher.cpp . . . 1141
7.323 /Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.hpp File Reference . . . 1142
7.324 Matcher.hpp . . . 1142
7.325 /Users/esposito/Software/minerule/src/Programs/MRMatch/mrmatch.cpp File Reference . . . 1143
7.325.1 Function Documentation . . . 1143
7.325.1.1 main() . . . 1143
7.326 mrmatch.cpp . . . 1144
7.327 /Users/esposito/Software/minerule/src/Programs/MRMatch/mrmatch.hpp File Reference . . . 1145
7.328 mrmatch.hpp . . . 1146
7.329 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.cpp File Reference . . . 1146
7.330 Options.cpp . . . 1147
7.331 /Users/esposito/Software/minerule/src/Programs/MRMatch/Options.cpp File Reference . . . 1147
7.332 Options.cpp . . . 1147
7.333 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.cpp File Reference . . . 1150
7.334 Options.cpp . . . 1150
7.335 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.hpp File Reference . . . 1151
7.336 Options.hpp . . . 1152
7.337 /Users/esposito/Software/minerule/src/Programs/MRMatch/Options.hpp File Reference . . . 1154
7.338 Options.hpp . . . 1154
7.339 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.hpp File Reference . . . 1155
7.340 Options.hpp . . . 1155

7.341 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.cpp File Reference 1156
7.342 RuleGidsDBMatcher.cpp . . . 1157
7.343 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.hpp File Reference 1158
7.344 RuleGidsDBMatcher.hpp . . . 1158
7.345 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.cpp File Reference . 1159
7.346 RuleGidsMatcher.cpp . . . 1159
7.347 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.hpp File Reference . 1160
7.348 RuleGidsMatcher.hpp . . . 1160
7.349 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/PrintRules.cpp File Reference . . 1161
7.349.1 Function Documentation . . . 1161
7.349.1.1 main() . . . 1162
7.350 PrintRules.cpp . . . 1162
7.351 /Users/esposito/Software/minerule/src/Programs/MRQuery/mrquery.cpp File Reference . . . 1164
7.351.1 Function Documentation . . . 1164
7.351.1.1 execQuery() . . . 1165
7.351.1.2 main() . . . 1165
7.351.1.3 parseMineruleName() . . . 1165
7.351.1.4 parseOptions() . . . 1166
7.351.1.5 printHelp() . . . 1168
7.351.1.6 printVersion() . . . 1168
7.352 mrquery.cpp . . . 1168
7.353 /Users/esposito/Software/minerule/src/Result/QueryResult.cpp File Reference . . . 1172
7.354 QueryResult.cpp . . . 1172
7.355 /Users/esposito/Software/minerule/src/Result/RuleFormatter.cpp File Reference . . . 1173
7.356 RuleFormatter.cpp . . . 1174
7.357 /Users/esposito/Software/minerule/src/Result/RulesMatcher.cpp File Reference . . . 1175
7.358 RulesMatcher.cpp . . . 1175
7.359 /Users/esposito/Software/minerule/src/Utils/AlgorithmTypes.cpp File Reference . . . 1176
7.360 AlgorithmTypes.cpp . . . 1176
7.361 /Users/esposito/Software/minerule/src/Utils/Bitstring.cpp File Reference . . . 1177
7.362 Bitstring.cpp . . . 1178
7.363 /Users/esposito/Software/minerule/src/Utils/Constraints.cpp File Reference . . . 1183
7.364 Constraints.cpp . . . 1183
7.365 /Users/esposito/Software/minerule/src/Utils/FileUtils.cpp File Reference . . . 1184
7.366 FileUtils.cpp . . . 1184
7.367 /Users/esposito/Software/minerule/src/Utils/MineruleErrors.cpp File Reference . . . 1185
7.368 MineruleErrors.cpp . . . 1185
7.369 /Users/esposito/Software/minerule/src/Utils/MineruleException.cpp File Reference . . . 1186
7.370 MineruleException.cpp . . . 1186
7.371 /Users/esposito/Software/minerule/src/Utils/MineruleLogs.cpp File Reference . . . 1187
7.371.1 Variable Documentation . . . 1188
7.371.1.1 MAX_LOG_LENGTH . . . 1188

7.372 MineruleLogs.cpp . . . 1188
7.373 /Users/esposito/Software/minerule/src/Utils/MineruleOptions.cpp File Reference . . . 1190
7.373.1 Variable Documentation . . . 1190
7.373.1.1 MINERULE_OPTIONS_PARSING_ERROR . . . 1190
7.374 MineruleOptions.cpp . . . 1190
7.375 /Users/esposito/Software/minerule/src/Utils/MRLogger.cpp File Reference . . . 1199
7.376 MRLogger.cpp . . . 1199
7.377 /Users/esposito/Software/minerule/src/Utils/OptionParser.ypp File Reference . . . 1201
7.378 OptionParser.ypp . . . 1201
7.379 /Users/esposito/Software/minerule/src/Utils/OptionParserLib.cpp File Reference . . . 1203
7.379.1 Function Documentation . . . 1203
7.379.1.1 OP_delete_buffer() . . . 1203
7.379.1.2 OP_scan_string() . . . 1203
7.379.1.3 OP_switch_to_buffer() . . . 1204
7.380 OptionParserLib.cpp . . . 1204
7.381 /Users/esposito/Software/minerule/src/Utils/Progress.cpp File Reference . . . 1205
7.382 Progress.cpp . . . 1205
7.383 /Users/esposito/Software/minerule/src/Utils/SQLUtils.cpp File Reference . . . 1207
7.384 SQLUtils.cpp . . . 1207
7.385 /Users/esposito/Software/minerule/src/Utils/StringUtils.cpp File Reference . . . 1209
7.386 StringUtils.cpp . . . 1209

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

minerule 15
mrc 41
mrdb 48
mrmatch 48
mrprint 51

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

__taglist_AND_node . . . 55
__taglist_OR_node . . . 56
__tagminerule . . . 56
__tagsimple_pred . . . 58
minerule::AggregateConstraint . . . 60
  minerule::AggregateAntiMonoConstraint . . . 59
    minerule::SumLessEq . . . 721
    minerule::SumLessThan . . . 723
  minerule::AggregateMonoConstraint . . . 63
    minerule::SumGreaterEq . . . 716
    minerule::SumGreaterThan . . . 719
minerule::Algorithms . . . 64
minerule::AlgorithmsOptions . . . 69
minerule::Attr_def . . . 75
Attribute . . . 76
AttributesUtil . . . 77
minerule::Bem_cond . . . 78
minerule::BitString . . . 91
  minerule::MapElement . . . 336
    minerule::BodyMapElement . . . 119
minerule::Body . . . 103
minerule::CatalogueInfo . . . 139
minerule::CatalogueInstaller . . . 142
  minerule::PostgresCatalogueInstaller . . . 528
    minerule::MySqlCatalogueInstaller . . . 415
minerule::CCSMSequence . . . 157
minerule::Connection . . . 170
mrdb::Connection . . . 178
minerule::Constraints . . . 180
minerule::Converter . . . 193
mrdb::DatabaseMetaData . . . 201
minerule::Dist_cond . . . 216
minerule::QueryNormalizer::SubstEntryHead::Elem . . . 218
minerule::EncodedNF . . . 220

minerule::EncodedNFIterator 224
minerule::EvaluationMeasure 230
    minerule::SupportMeasure 726
std::exception
    minerule::MineruleException 361
    minerule::TimeOutException 728
    mrc::Exception 232
    std::runtime_error
        mrdb::SQLException 694
minerule::ExpressionNFCoder 233
minerule::QueryResult::FastSorter 236
minerule::RuleFormatter::FieldWidths 236
minerule::FileUtils 238
minerule::FSTree 245
minerule::FSTreeNode 256
minerule::FSTreeSequence 260
minerule::GAQueryCombinator 267
minerule::Head 288
minerule::HeadBodyPredicatesSeparator 293
minerule::HeaderQuery 294
minerule::IncrementalAlgorithm 302
    minerule::ConstrTree 186
    minerule::DestrTree 207
    minerule::IDIncrementalAlgorithm 297
    minerule::ResultCombinator 568
minerule::Interval 304
minerule::IntervalChecker 314
minerule::InvalidConfigurationFilter 316
ItemSetType
    minerule::TransactionBase< ItemSetType > 731
        minerule::ItemTransaction< ItemSetType > 320
minerule::ItemType 322
minerule::QueryResult::Iterator 329
minerule::SourceTable::Iterator 331
std::iterator
    minerule::CountingIterator 198
minerule::FSTreeSequence::less_sequence 334
mrc::Options::ListFormat 335
std::map< K, T >
    minerule::BodyMap 109
    minerule::OptimizerCatalogue::Catalogue 136
        minerule::OptimizerCatalogue::DepFunCatalogue 203
        minerule::OptimizerCatalogue::EqKeysCatalogue 226
mrmatch::Matcher 346
    mrmatch::GidRulesMatcher 284
    mrmatch::RuleGidsMatcher 598
        mrmatch::RuleGidsDBMatcher 591
minerule::MiningAlgorithmBase 389
    minerule::MiningAlgorithm 385
        minerule::BFSWithGidsAndCross 81
        minerule::BFSWithGidsNoCross 86
        minerule::CARE 131
        minerule::CCSMiner 148
        minerule::ConstrItemSetsExtraction 182
        minerule::FSMiner 241
        minerule::STSMiner 707
minerule::MinMax 395

minerule::MinMaxPair . . . 396
minerule::MRDebugPusher . . . 404
minerule::MRErrPusher . . . 405
minerule::MRLogger . . . 406
minerule::MRLogPusher . . . 412
MRResultSetIterator . . . 413
minerule::MRWarnPusher . . . 414
minerule::NewRule . . . 423
minerule::NodeRow . . . 428
  minerule::NodeRowB . . . 431
minerule::OptimizedMinerule::OptimizationInfo . . . 446
minerule::OptimizedMinerule . . . 452
minerule::OptimizerCatalogue . . . 461
minerule::OptionBase . . . 472
OptionBase
  ItemsetsMiningAlgorithms . . . 318
  MineruleOptions . . . 362
  MiningAlgorithms . . . 392
  Mrdb . . . 400
  Optimizations . . . 447
  Optimizations::Combinator . . . 167
  OutStream . . . 488
  Parsers . . . 522
  RulesMiningAlgorithms . . . 605
  RulesMiningAlgorithms::FPGrowth . . . 239
  RulesMiningAlgorithms::PartitionBase . . . 525
  RulesMiningAlgorithms::PartitionWithClusters . . . 527
  Safety . . . 611
  SequencesMiningAlgorithms . . . 614
mrc::Options . . . 473
mrmatch::Options . . . 480
mrprint::Options . . . 485
minerule::ParsedMinerule . . . 491
  minerule::OptimizerCatalogue::MineruleResultInfo . . . 375
minerule::PredicateUtils . . . 552
minerule::PrepareDataUtils . . . 554
mrdb::PreparedStatement . . . 558
mrc::Printer . . . 561
minerule::Progress . . . 562
minerule::PtrSimplePredComp . . . 565
minerule::QueryNormalizer . . . 566
minerule::QueryResult . . . 568
mrdb::ResultSet . . . 575
mrdb::ResultSetMetaData . . . 578
minerule::Rule . . . 581
minerule::RuleFormatter . . . 587
  minerule::SimpleRuleFormatter . . . 628
    minerule::SortedRuleFormatter< Sorter > . . . 637
RuleSetType
  minerule::TransactionBase< RuleSetType > . . . 731
    minerule::RuleTransaction< RuleSetType > . . . 610
minerule::RulesMatcher . . . 603
minerule::Scmp . . . 614
std::set< K >
  minerule::TransactionBase< std::set< ItemType > > . . . 731
    minerule::ItemTransaction< std::set< ItemType > > . . . 320
      minerule::Transaction . . . 729

minerule::QueryResult::ResultSet< Sorter > . . . 571
SetType
minerule::TransactionBase< SetType > . . . 731
minerule::SimplePredAnalyzer . . . 617
minerule::SimplePredicate . . . 619
minerule::SimplePredicateBase . . . 622
minerule::ParsedSimplePredicate . . . 515
minerule::QueryResult::SortBodyHead . . . 632
minerule::QueryResult::SortBodyHeadSuppConf . . . 633
minerule::QueryResult::SortConfBodyHeadSupp . . . 634
minerule::QueryResult::SortConfBodySuppHead . . . 635
minerule::QueryResult::SortConfSuppBodyHead . . . 636
minerule::QueryResult::SortHeadBodySuppConf . . . 642
minerule::QueryResult::SortSuppConfBodyHead . . . 642
minerule::SourceRow . . . 643
minerule::SourceRowAttrCollectionDescriptor . . . 647
minerule::SourceRowColumnIds . . . 668
minerule::SourceRowElement . . . 672
minerule::SourceRowAttribute . . . 649
minerule::GenericSourceRowAttribute . . . 272
minerule::MemDebugGenericSourceRowAttribute . . . 349
minerule::NumericSourceRowAttribute . . . 434
minerule::SourceRowAttributeCollection . . . 658
minerule::SourceRowEmptyElement . . . 679
minerule::SourceRowMetaInfo . . . 686
minerule::SourceTable . . . 688
minerule::SourceTableRequirements . . . 691
minerule::SQLUtils . . . 695
mrdb::Statement . . . 699
minerule::StringCompare . . . 701
minerule::StringUtils . . . 701
minerule::QueryNormalizer::SubstEntryBody . . . 713
minerule::QueryNormalizer::SubstEntryHead . . . 714
minerule::TranslatedTable . . . 732
minerule::TranslatedTableStandardSQL . . . 734
minerule::TranslationManager . . . 738
minerule::TranslationManagerStandardSQL . . . 741
mrdb::Types . . . 745
minerule::VarSet . . . 746
minerule::VarSetEnumerator . . . 749
std::vector< T >
minerule::BitVector . . . 102
minerule::NewRuleSet . . . 427
minerule::PredConjunction . . . 536
minerule::PredConjunctionBase . . . 538
minerule::ParsedPredConjunction . . . 502
minerule::Predicate . . . 543
minerule::PredicateBase . . . 547
minerule::ParsedPredicate . . . 508

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

__taglist_AND_node . . . 55
__taglist_OR_node . . . 56
__tagminerule . . . 56
__tagsimple_pred . . . 58
minerule::AggregateAntiMonoConstraint . . . 59
minerule::AggregateConstraint . . . 60
minerule::AggregateMonoConstraint . . . 63
minerule::Algorithms . . . 64
minerule::AlgorithmsOptions . . . 69
minerule::Attr_def . . . 75
Attribute . . . 76
AttributesUtil . . . 77
minerule::Bem_cond . . . 78
minerule::BFSWithGidsAndCross . . . 81
minerule::BFSWithGidsNoCross . . . 86
minerule::BitString . . . 91
minerule::BitVector . . . 102
minerule::Body . . . 103
minerule::BodyMap . . . 109
minerule::BodyMapElement . . . 119
minerule::CARE . . . 131
minerule::OptimizerCatalogue::Catalogue . . . 136
minerule::CatalogueInfo . . . 139
minerule::CatalogueInstaller . . . 142
minerule::CCSMiner . . . 148
minerule::CCSMSequence . . . 157
Optimizations::Combinator . . . 167
minerule::Connection . . . 170
mrdb::Connection . . . 178
minerule::Constraints . . . 180
minerule::ConstrItemSetsExtraction . . . 182
minerule::ConstrTree . . . 186
minerule::Converter . . . 193
minerule::CountingIterator . . . 198
mrdb::DatabaseMetaData . . . 201

minerule::OptimizerCatalogue::DepFunCatalogue . . . 203
minerule::DestrTree . . . 207
minerule::Dist_cond . . . 216
minerule::QueryNormalizer::SubstEntryHead::Elem . . . 218
minerule::EncodedNF . . . 220
minerule::EncodedNFIterator . . . 224
minerule::OptimizerCatalogue::EqKeysCatalogue . . . 226
minerule::EvaluationMeasure . . . 230
mrc::Exception . . . 232
minerule::ExpressionNFCoder . . . 233
minerule::QueryResult::FastSorter . . . 236
minerule::RuleFormatter::FieldWidths . . . 236
minerule::FileUtils . . . 238
RulesMiningAlgorithms::FPGrowth . . . 239
minerule::FSMiner . . . 241
minerule::FSTree . . . 245
minerule::FSTreeNode . . . 256
minerule::FSTreeSequence . . . 260
minerule::GAQueryCombinator . . . 267
minerule::GenericSourceRowAttribute . . . 272
mrmatch::GidRulesMatcher . . . 284
minerule::Head . . . 288
minerule::HeadBodyPredicatesSeparator . . . 293
minerule::HeaderQuery . . . 294
minerule::IDIncrementalAlgorithm . . . 297
minerule::IncrementalAlgorithm . . . 302
minerule::Interval . . . 304
minerule::IntervalChecker . . . 314
minerule::InvalidConfigurationFilter . . . 316
ItemsetsMiningAlgorithms . . . 318
minerule::ItemTransaction< ItemSetType > . . . 320
minerule::ItemType . . . 322
minerule::QueryResult::Iterator . . . 329
minerule::SourceTable::Iterator . . . 331
minerule::FSTreeSequence::less_sequence . . . 334
mrc::Options::ListFormat . . . 335
minerule::MapElement . . . 336
mrmatch::Matcher . . . 346
minerule::MemDebugGenericSourceRowAttribute . . . 349
minerule::MineruleException . . . 361
MineruleOptions . . . 362
minerule::OptimizerCatalogue::MineruleResultInfo . . . 375
minerule::MiningAlgorithm . . . 385
minerule::MiningAlgorithmBase . . . 389
MiningAlgorithms . . . 392
minerule::MinMax . . . 395
minerule::MinMaxPair . . . 396
Mrdb . . . 400
minerule::MRDebugPusher . . . 404
minerule::MRErrPusher . . . 405
minerule::MRLogger . . . 406
minerule::MRLogPusher . . . 412
MRResultSetIterator . . . 413
minerule::MRWarnPusher . . . 414
minerule::MySqlCatalogueInstaller . . . 415
minerule::NewRule . . . 423
minerule::NewRuleSet . . . 427
minerule::NodeRow . . . 428

minerule::NodeRowB . . . 431
minerule::NumericSourceRowAttribute . . . 434
minerule::OptimizedMinerule::OptimizationInfo . . . 446
Optimizations . . . 447
minerule::OptimizedMinerule . . . 452
minerule::OptimizerCatalogue . . . 461
minerule::OptionBase . . . 472
mrc::Options . . . 473
mrmatch::Options . . . 480
mrprint::Options . . . 485
OutStream . . . 488
minerule::ParsedMinerule . . . 491
minerule::ParsedPredConjunction . . . 502
minerule::ParsedPredicate . . . 508
minerule::ParsedSimplePredicate . . . 515
Parsers . . . 522
RulesMiningAlgorithms::PartitionBase . . . 525
RulesMiningAlgorithms::PartitionWithClusters . . . 527
minerule::PostgresCatalogueInstaller . . . 528
minerule::PredConjunction . . . 536
minerule::PredConjunctionBase . . . 538
minerule::Predicate . . . 543
minerule::PredicateBase . . . 547
minerule::PredicateUtils . . . 552
minerule::PrepareDataUtils . . . 554
mrdb::PreparedStatement . . . 558
mrc::Printer . . . 561
minerule::Progress . . . 562
minerule::PtrSimplePredComp . . . 565
minerule::QueryNormalizer . . . 566
minerule::QueryResult . . . 568
minerule::ResultCombinator . . . 568
minerule::QueryResult::ResultSet< Sorter > . . . 571
mrdb::ResultSet . . . 575
mrdb::ResultSetMetaData . . . 578
minerule::Rule . . . 581
minerule::RuleFormatter . . . 587
mrmatch::RuleGidsDBMatcher . . . 591
mrmatch::RuleGidsMatcher . . . 598
minerule::RulesMatcher . . . 603
RulesMiningAlgorithms . . . 605
minerule::RuleTransaction< RuleSetType > . . . 610
Safety . . . 611
minerule::Scmp . . . 614
SequencesMiningAlgorithms . . . 614
minerule::SimplePredAnalyzer . . . 617
minerule::SimplePredicate . . . 619
minerule::SimplePredicateBase . . . 622
minerule::SimpleRuleFormatter . . . 628
minerule::QueryResult::SortBodyHead . . . 632
minerule::QueryResult::SortBodyHeadSuppConf . . . 633
minerule::QueryResult::SortConfBodyHeadSupp . . . 634
minerule::QueryResult::SortConfBodySuppHead . . . 635
minerule::QueryResult::SortConfSuppBodyHead . . . 636
minerule::SortedRuleFormatter< Sorter > . . . 637
minerule::QueryResult::SortHeadBodySuppConf . . . 642
minerule::QueryResult::SortSuppConfBodyHead . . . 642
minerule::SourceRow . . . 643

minerule::SourceRowAttrCollectionDescriptor 647
minerule::SourceRowAttribute 649
minerule::SourceRowAttributeCollection 658
minerule::SourceRowColumnIds 668
minerule::SourceRowElement 672
minerule::SourceRowEmptyElement 679
minerule::SourceRowMetaInfo 686
minerule::SourceTable 688
minerule::SourceTableRequirements 691
mrdb::SQLException 694
minerule::SQLUtils 695
mrdb::Statement 699
minerule::StringCompare 701
minerule::StringUtils 701
minerule::STSMiner 707
minerule::QueryNormalizer::SubstEntryBody 713
minerule::QueryNormalizer::SubstEntryHead 714
minerule::SumGreaterEq 716
minerule::SumGreaterThan 719
minerule::SumLessEq 721
minerule::SumLessThan 723
minerule::SupportMeasure 726
minerule::TimeOutException 728
minerule::Transaction 729
minerule::TransactionBase< SetType > 731
minerule::TranslatedTable 732
minerule::TranslatedTableStandardSQL 734
minerule::TranslationManager 738
minerule::TranslationManagerStandardSQL 741
mrdb::Types 745
minerule::VarSet 746
minerule::VarSetEnumerator 749

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/Users/esposito/Software/minerule/include/minerule/Algorithms/Algorithms.hpp . . . 751
/Users/esposito/Software/minerule/include/minerule/Algorithms/AlgorithmsOptions.hpp . . . 752
/Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsAndCross.hpp . . . 754
/Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsNoCross.hpp . . . 756
/Users/esposito/Software/minerule/include/minerule/Algorithms/BitVector.hpp . . . 759
/Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp . . . 760
/Users/esposito/Software/minerule/include/minerule/Algorithms/Care.hpp . . . 763
/Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMiner.hpp . . . 764
/Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMSequence.hpp . . . 765
/Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrItemSetsExtraction.hpp . . . 771
/Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrTree.hpp . . . 772
/Users/esposito/Software/minerule/include/minerule/Algorithms/DestrTree.hpp . . . 773
/Users/esposito/Software/minerule/include/minerule/Algorithms/FSMiner.hpp . . . 775
/Users/esposito/Software/minerule/include/minerule/Algorithms/FSTree.hpp . . . 776
/Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeNode.hpp . . . 778
/Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeSequence.hpp . . . 779
/Users/esposito/Software/minerule/include/minerule/Algorithms/IDIncrementalAlgorithm.hpp . . . 780
/Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp . . . 781
/Users/esposito/Software/minerule/include/minerule/Algorithms/IncrementalAlgorithm.hpp . . . 784
/Users/esposito/Software/minerule/include/minerule/Algorithms/MiningAlgorithmBase.hpp . . . 785
/Users/esposito/Software/minerule/include/minerule/Algorithms/ResultCombinator.hpp . . . 786
/Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp . . . 787
/Users/esposito/Software/minerule/include/minerule/Database/Connection.hpp . . . 797
/Users/esposito/Software/minerule/include/minerule/Database/ItemType.hpp . . . 801
/Users/esposito/Software/minerule/include/minerule/Database/MRResultSet.hpp . . . 804
/Users/esposito/Software/minerule/include/minerule/Database/PrepareDataUtils.hpp . . . 805
/Users/esposito/Software/minerule/include/minerule/Database/SourceRow.hpp . . . 806
/Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp . . . 808
/Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttributeCollection.hpp . . . 811
/Users/esposito/Software/minerule/include/minerule/Database/SourceRowColumnIds.hpp . . . 813
/Users/esposito/Software/minerule/include/minerule/Database/SourceRowElement.hpp . . . 815
/Users/esposito/Software/minerule/include/minerule/Database/SourceRowMetaInfo.hpp . . . 818
/Users/esposito/Software/minerule/include/minerule/Database/SourceTable.hpp . . . 819
/Users/esposito/Software/minerule/include/minerule/Database/SourceTableRequirements.hpp . . . 821
/Users/esposito/Software/minerule/include/minerule/Database/Transaction.hpp . . . 822

/Users/esposito/Software/minerule/include/minerule/mrdb/Connection.hpp . . . 799
/Users/esposito/Software/minerule/include/minerule/mrdb/DatabaseMetaData.hpp . . . 823
/Users/esposito/Software/minerule/include/minerule/mrdb/PreparedStatement.hpp . . . 824
/Users/esposito/Software/minerule/include/minerule/mrdb/ResultSet.hpp . . . 825
/Users/esposito/Software/minerule/include/minerule/mrdb/ResultSetMetaData.hpp . . . 826
/Users/esposito/Software/minerule/include/minerule/mrdb/SQLException.hpp . . . 827
/Users/esposito/Software/minerule/include/minerule/mrdb/Statement.hpp . . . 828
/Users/esposito/Software/minerule/include/minerule/mrdb/Types.hpp . . . 828
/Users/esposito/Software/minerule/include/minerule/Optimizer/CatalogueInstaller.hpp . . . 829
/Users/esposito/Software/minerule/include/minerule/Optimizer/GAQueryCombinator.hpp . . . 830
/Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizedMinerule.hpp . . . 834
/Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp . . . 836
/Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp . . . 839
/Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/MySqlCatalogueInstaller.hpp . . . 832
/Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp 833
/Users/esposito/Software/minerule/include/minerule/Parsers/EvaluationMeasure.hpp . . . 842
/Users/esposito/Software/minerule/include/minerule/Parsers/HeaderQuery.hpp . . . 843
/Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp . . . 844
/Users/esposito/Software/minerule/include/minerule/Parsers/ParsedPredicate.hpp . . . 850
/Users/esposito/Software/minerule/include/minerule/Parsers/ParserLibrary.hpp . . . 852
/Users/esposito/Software/minerule/include/minerule/Parsers/PredicateBase.hpp . . . 853
/Users/esposito/Software/minerule/include/minerule/Parsers/SupportMeasure.hpp . . . 856
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp . . . 857
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp . 859
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp . . . 860
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/InvalidConfigurationFilter.hpp . . . 863
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp . . . 864
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/PredicateUtils.hpp . . . 867
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/SimplePredAnalyzer.hpp . . . 869
/Users/esposito/Software/minerule/include/minerule/PredicateUtils/VarSet.hpp . . . 870
/Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtable.hpp . . . 873
/Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtablestandardsql.hpp . . . 874
/Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanager.hpp . . . 876
/Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanagerstandardsql.hpp . 877
/Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp . . . 879
/Users/esposito/Software/minerule/include/minerule/Result/QueryResult-impl.hpp . . . 884
/Users/esposito/Software/minerule/include/minerule/Result/QueryResult.hpp . . . 886
/Users/esposito/Software/minerule/include/minerule/Result/Rule.hpp . . . 887
/Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp . . . 889
/Users/esposito/Software/minerule/include/minerule/Result/RulesMatcher.hpp . . . 891
/Users/esposito/Software/minerule/include/minerule/Utils/AlgorithmTypes.hpp . . . 892
/Users/esposito/Software/minerule/include/minerule/Utils/Bitstring.hpp . . . 893
/Users/esposito/Software/minerule/include/minerule/Utils/common.hpp . . . 895
/Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp . . . 896
/Users/esposito/Software/minerule/include/minerule/Utils/Converter.hpp . . . 898
/Users/esposito/Software/minerule/include/minerule/Utils/FileUtils.hpp . . . 900
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleErrors.hpp . . . 901
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleException.hpp . . . 902
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp . . . 903
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions.hpp . . . 906
/Users/esposito/Software/minerule/include/minerule/Utils/MinMaxPair.hpp . . . 923
/Users/esposito/Software/minerule/include/minerule/Utils/MRLogger.hpp . . . 924
/Users/esposito/Software/minerule/include/minerule/Utils/OptionParserLib.hpp . . . 927
/Users/esposito/Software/minerule/include/minerule/Utils/Progress.hpp . . . 928
/Users/esposito/Software/minerule/include/minerule/Utils/SQLUtils.hpp . . . 929
/Users/esposito/Software/minerule/include/minerule/Utils/StringUtils.hpp . . . 931
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms.hpp
907

/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/mrdb.hpp . 912
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/optimizations.hpp 914
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/outstream.hpp 916
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/parsers.hpp 917
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/root.hpp . . 918
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/safety.hpp . 922
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/itemsetmining.hpp 908
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp 909
/Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/sequencemining.hp 911
/Users/esposito/Software/minerule/src/Algorithms/Algorithms.cpp 932
/Users/esposito/Software/minerule/src/Algorithms/AlgorithmsOptions.cpp 936
/Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsAndCross.cpp 937
/Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsNoCross.cpp 941
/Users/esposito/Software/minerule/src/Algorithms/Bodymap.cpp 946
/Users/esposito/Software/minerule/src/Algorithms/Care.cpp 952
/Users/esposito/Software/minerule/src/Algorithms/CCSMiner.cpp 954
/Users/esposito/Software/minerule/src/Algorithms/CCSMSequence.cpp 960
/Users/esposito/Software/minerule/src/Algorithms/ConstrItemSetsExtraction.cpp 964
/Users/esposito/Software/minerule/src/Algorithms/ConstrTree.cpp 966
/Users/esposito/Software/minerule/src/Algorithms/DestrTree.cpp 969
/Users/esposito/Software/minerule/src/Algorithms/FSMiner.cpp 973
/Users/esposito/Software/minerule/src/Algorithms/FSTree.cpp 975
/Users/esposito/Software/minerule/src/Algorithms/FSTreeNode.cpp 980
/Users/esposito/Software/minerule/src/Algorithms/FSTreeSequence.cpp 981
/Users/esposito/Software/minerule/src/Algorithms/IDIncrementalAlgorithm.cpp 983
/Users/esposito/Software/minerule/src/Algorithms/IncrAlgoClasses.cpp 986
/Users/esposito/Software/minerule/src/Algorithms/IncrementalAlgorithm.cpp 992
/Users/esposito/Software/minerule/src/Algorithms/MiningAlgorithmBase.cpp 993
/Users/esposito/Software/minerule/src/Algorithms/ResultCombinator.cpp 994
/Users/esposito/Software/minerule/src/Algorithms/STSMiner.cpp 996
/Users/esposito/Software/minerule/src/Database/Connection.cpp 1015
/Users/esposito/Software/minerule/src/Database/PrepareDataUtils.cpp 1022
/Users/esposito/Software/minerule/src/Database/SourceRow.cpp 1027
/Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp 1029
/Users/esposito/Software/minerule/src/Database/SourceRowAttributeCollection.cpp 1032
/Users/esposito/Software/minerule/src/Database/SourceRowColumnIds.cpp 1036
/Users/esposito/Software/minerule/src/Database/SourceRowElement.cpp 1037
/Users/esposito/Software/minerule/src/Database/SourceRowMetaInfo.cpp 1038
/Users/esposito/Software/minerule/src/Database/SourceTable.cpp 1041
/Users/esposito/Software/minerule/src/Optimizer/CatalogueInstaller.cpp 1042
/Users/esposito/Software/minerule/src/Optimizer/GAQueryCombinator.cpp 1043
/Users/esposito/Software/minerule/src/Optimizer/OptimizedMinerule.cpp 1048
/Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp 1054
/Users/esposito/Software/minerule/src/Optimizer/QueryNormalizer.cpp 1062
/Users/esposito/Software/minerule/src/Optimizer/Installers/PostgresCatalogueInstaller.cpp 1046
/Users/esposito/Software/minerule/src/Parsers/MineruleParser.ypp 1074
/Users/esposito/Software/minerule/src/Parsers/ParsedMinerule.cpp 1086
/Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp 1095
/Users/esposito/Software/minerule/src/Parsers/ParserLibrary.cpp 1096
/Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp 1098
/Users/esposito/Software/minerule/src/Parsers/SupportMeasure.cpp 1101
/Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp 1102
/Users/esposito/Software/minerule/src/PredicateUtils/HeadBodyPredicatesSeparator.cpp 1107

/Users/esposito/Software/minerule/src/PredicateUtils/Interval.cpp . . . 1108
/Users/esposito/Software/minerule/src/PredicateUtils/Predicate.cpp . . . 1112
/Users/esposito/Software/minerule/src/PredicateUtils/PredicateUtils.cpp . . . 1114
/Users/esposito/Software/minerule/src/PredicateUtils/SimplePredAnalyzer.cpp . . . 1117
/Users/esposito/Software/minerule/src/Programs/Minerule/minerule.cpp . . . 1119
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/ErrorCodes.hpp . . . 1123
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/Exception.hpp . . . 1124
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/mrcatalogue.cpp . . . 1125
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/MRCUtils.cpp . . . 1127
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/MRCUtils.hpp . . . 1132
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.cpp . . . 1146
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.hpp . . . 1151
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.cpp . . . 1133
/Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.hpp . . . 1134
/Users/esposito/Software/minerule/src/Programs/MRDefaultOptions/MRDefaultOptions.cpp . . . 1135
/Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.cpp . . . 1138
/Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.hpp . . . 1139
/Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.cpp . . . 1140
/Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.hpp . . . 1142
/Users/esposito/Software/minerule/src/Programs/MRMatch/mrmatch.cpp . . . 1143
/Users/esposito/Software/minerule/src/Programs/MRMatch/mrmatch.hpp . . . 1145
/Users/esposito/Software/minerule/src/Programs/MRMatch/Options.cpp . . . 1147
/Users/esposito/Software/minerule/src/Programs/MRMatch/Options.hpp . . . 1154
/Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.cpp . . . 1156
/Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.hpp . . . 1158
/Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.cpp . . . 1159
/Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.hpp . . . 1160
/Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.cpp . . . 1150
/Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.hpp . . . 1155
/Users/esposito/Software/minerule/src/Programs/MRPrintRules/PrintRules.cpp . . . 1161
/Users/esposito/Software/minerule/src/Programs/MRQuery/mrquery.cpp . . . 1164
/Users/esposito/Software/minerule/src/Result/QueryResult.cpp . . . 1172
/Users/esposito/Software/minerule/src/Result/RuleFormatter.cpp . . . 1173
/Users/esposito/Software/minerule/src/Result/RulesMatcher.cpp . . . 1175
/Users/esposito/Software/minerule/src/Utils/AlgorithmTypes.cpp . . . 1176
/Users/esposito/Software/minerule/src/Utils/Bitstring.cpp . . . 1177
/Users/esposito/Software/minerule/src/Utils/Constraints.cpp . . . 1183
/Users/esposito/Software/minerule/src/Utils/FileUtils.cpp . . . 1184
/Users/esposito/Software/minerule/src/Utils/MineruleErrors.cpp . . . 1185
/Users/esposito/Software/minerule/src/Utils/MineruleException.cpp . . . 1186
/Users/esposito/Software/minerule/src/Utils/MineruleLogs.cpp . . . 1187
/Users/esposito/Software/minerule/src/Utils/MineruleOptions.cpp . . . 1190
/Users/esposito/Software/minerule/src/Utils/MRLogger.cpp . . . 1199
/Users/esposito/Software/minerule/src/Utils/OptionParser.ypp . . . 1201
/Users/esposito/Software/minerule/src/Utils/OptionParserLib.cpp . . . 1203
/Users/esposito/Software/minerule/src/Utils/Progress.cpp . . . 1205
/Users/esposito/Software/minerule/src/Utils/SQLUtils.cpp . . . 1207
/Users/esposito/Software/minerule/src/Utils/StringUtils.cpp . . . 1209

Chapter 5

Namespace Documentation

5.1 minerule Namespace Reference

Data Structures

- class AggregateAntiMonoConstraint
- class AggregateConstraint
- class AggregateMonoConstraint
- class Algorithms
- class AlgorithmsOptions
- struct Attr_def
- class Bem_cond
- class BFSWithGidsAndCross
- class BFSWithGidsNoCross
- class BitString
- class BitVector
- class Body
- class BodyMap
- class BodyMapElement
- class CARE
- class CatalogueInfo
- class CatalogueInstaller
- class CCSMiner
- class CCSMSequence
- class Connection
- struct Constraints
- class ConstrItemSetsExtraction
- class ConstrTree
- class Converter
- class CountingIterator
- class DestrTree
- class Dist_cond
- class EncodedNF
- class EncodedNFIterator
- class EvaluationMeasure
- class ExpressionNFCoder
- class FileUtils
- class FSMiner

- class FSTree
- class FSTreeNode
- class FSTreeSequence
- class GAQueryCombinator
- class GenericSourceRowAttribute
- class Head
- class HeadBodyPredicatesSeparator
- class HeaderQuery
- class IDIncrementalAlgorithm
- class IncrementalAlgorithm
- class Interval
- class IntervalChecker
- class InvalidConfigurationFilter
- class ItemTransaction
- class ItemType
- class MapElement
- class MemDebugGenericSourceRowAttribute
- class MineruleException
- class MiningAlgorithm
- class MiningAlgorithmBase
- class MinMax
- class MinMaxPair
- class MRDebugPusher
- class MRErrPusher
- class MRLogger
- class MRLogPusher
- class MRWarnPusher
- class MySqlCatalogueInstaller
- class NewRule
- class NewRuleSet
- class NodeRow
- class NodeRowB
- class NumericSourceRowAttribute
- class OptimizedMinerule
- class OptimizerCatalogue
- class OptionBase
- class ParsedMinerule
- class ParsedPredConjunction
- class ParsedPredicate
- class ParsedSimplePredicate
- class PostgresCatalogueInstaller
- class PredConjunction
- class PredConjunctionBase
- class Predicate
- class PredicateBase
- class PredicateUtils
- class PrepareDataUtils
- class Progress
- class PtrSimplePredComp
- class QueryNormalizer
- class QueryResult
- class ResultCombinator
- class Rule
- class RuleFormatter
- class RulesMatcher

- class RuleTransaction
- class Scmp
- class SimplePredAnalyzer
- class SimplePredicate
- class SimplePredicateBase
- class SimpleRuleFormatter
- class SortedRuleFormatter
- class SourceRow
- class SourceRowAttrCollectionDescriptor
- class SourceRowAttribute
- class SourceRowAttributeCollection
- class SourceRowColumnIds
- class SourceRowElement
- class SourceRowEmptyElement
- class SourceRowMetaInfo
- class SourceTable
- class SourceTableRequirements
- class SQLUtils
- class StringCompare
- class StringUtils
- class STSMiner
- class SumGreaterEq
- class SumGreaterThan
- class SumLessEq
- class SumLessThan
- class SupportMeasure
- class TimeOutException
- class Transaction
- class TransactionBase
- class TranslatedTable
- class TranslatedTableStandardSQL
- class TranslationManager
- class TranslationManagerStandardSQL
- class VarSet
- class VarSetEnumerator

Typedefs

- typedef MapElement GidList
- typedef std::map< long, int > GroupMap
- typedef std::pair< long, int > GroupPair
- typedef std::vector< ItemType > ItemSet
- typedef unsigned int EncodingBaseType
- typedef int boolean
- typedef unsigned int Bits

Enumerations

- enum MiningTasks { MTMineRules =0 , MTMineItemsets , MTMineSequences , MTEnd }
- enum AlgorithmTypes {
  ATNone =0 , ATBFSWithGidsNoCross , ATBFSWithGidsAndCross , ATCare ,
  ATConstrainedItemsets , ATEnd }
- enum MineruleErrors {
  MR_ERROR_NO_ERROR =0 , MR_ERROR_UNKNOWN , MR_ERROR_INTERNAL , MR_ERROR_INPUT_FILE_NOT_FOUN
  ,
  MR_ERROR_OUTPUT_FILE_PROBLEM , MR_ERROR_NO_MINERULE_SPECIFIED , MR_ERROR_NO_OPTIONFILE_SPE
  , MR_ERROR_OPTION_CONFIGURATION ,
  MR_ERROR_MINERULE_ALREADY_EXISTS , MR_ERROR_DATABASE_ERROR , MR_ERROR_OPTION_PARSING
  , MR_ERROR_MINERULETEXT_PARSING ,
  MR_ERROR_CATALOGUE_ERROR , MR_ERROR_OPTIMIZER_ERROR , MR_ERROR_INSTALLATION_PROBLEM
  , MR_ERROR_SAFETY_PROBLEM }

Functions

- std::ostream & operator<< (std::ostream &os, const SourceRow &sr)
- void removeNode (list_AND_node **l)
- void removeAllNodes (list_AND_node *&l)
- std::ostream & operator<< (std::ostream &os, const ParsedMinerule &mr)
- float floatFromText (char *text)
- std::string trim (std::string text)
- void init_mrparser ()
- void parseMinerule (std::string minerule_text, ParsedMinerule &output)
- ParsedMinerule & getParserOutputObj ()
- std::ostream & operator<< (std::ostream &os, const EncodedNF &nf)
- std::ostream & operator<< (std::ostream &os, const Interval &i)
- const std::string & miningTaskToString (MiningTasks mt)
- MiningTasks stringToMiningTask (const std::string &s)
- std::string buildStringWithListOfMiningTasks ()
- const std::string & stringWithListOfMiningTasks ()
- const std::string & algorithmTypeToString (AlgorithmTypes t)
- AlgorithmTypes stringToAlgorithmType (const std::string &s)
- std::string buildStringWithListOfAlgorithmTypes ()
- std::string stringWithListOfAlgorithmTypes ()
- std::ostream & operator<< (std::ostream &out, const BitString &bs)
- std::istream & operator>> (std::istream &in, BitString &bs)
- const char * me_error_name (MineruleErrors me)
- int me_error_begin ()
- int me_error_end ()
- void logRespectingMaxLogLength (std::ostream &(*logger)(void), size_t indentLen, const std::string &msg)
- std::ostream & MRLog ()
- void MRLog (const std::string &msg)
- std::ostream & MRErr ()
- void MRErr (const std::string &msg)
- std::ostream & MRWarn ()
- void MRWarn (const std::string &msg)
- std::ostream & MRDebug ()
- void MRDebug (const std::string &msg)
- void MRLogPush (const std::string &descr)
- void MRLogPop ()
- void MRErrPush (const std::string &descr)

- void MRErrPop ()
- void MRWarnPush (const std::string &descr)
- void MRWarnPop ()
- void MRDebugPush (const std::string &descr)
- void MRDebugPop ()
- void MRLogStartMeasuring (const std::string &description)
- void MRLogStopMeasuring (const std::string &description)
- void MRLogShowMeasurements ()
- void initializeOptionsFromFile (MineruleOptions &mrOpts, FILE *file)
- void initializeOptionsFromString (MineruleOptions &mrOpts, std::string str)
- void pushOptionClassIntoContext (const std::string &oclass)
- void popOptionClassFromContext ()
- void setOption (const std::string &name, const std::string &value)
- void abort_handler (int s)
- void install_abort_handler ()
- void angledSpeenWheelUpdateHandler (int n, int count, int num)
- void barSpeenWheelUpdateHandler (int n, int count, int num)
- void tickNumberUpdateHandler (int n, int count, int num)
- void clockHandsUpdateHandler (int n, int count, int num)
- std::ostream & operator<< (std::ostream &os, const ItemType &it)
- std::ostream & operator<< (std::ostream &os, const SourceRowAttribute &attr)
- std::ostream & operator<< (std::ostream &o, const SourceRowColumnIds &rowDes)
- std::ostream & operator<< (std::ostream &os, const SourceRowElement &el)
- std::ostream & operator<< (std::ostream &os, const SimplePredicate &s)
- std::ostream & operator<< (std::ostream &os, const PredConjunction &p)
- std::ostream & operator<< (std::ostream &os, const Predicate &p)
- std::ostream & operator<< (std::ostream &os, const VarSet &vset)
- int MRLogGetNewID ()
- std::ostream & operator<< (std::ostream &os, const MinMaxPair &p)

Variables

- const int HAM_LOSS = 10
- const int DISJ_LOSS = 1
- const int QUERY_LOSS = 1
- ParsedMinerule * outputObj
- const char * MineruleErrorsNames [ ]
- std::vector< MineruleOptions::OptionBase * > context
- const int DEBUG_LEVEL = 0
- const int DB_LEVEL_ALL = 1

5.1.1 Detailed Description

This module provides an algorithm that searches for a combination of queries which yelds a result equivalent to a fixed one. The search algorithm is a genetic one having the following characteristics: GENOMA: bistd::string of n*m bits. It is interpreted as a boolean formula having m disjunctions, each disjunct has n conjuncts. If a disjunct does not have any bit set to one, then it is empty and is not considered. For instance, let us consider the n=4 queries: Q1, Q2, Q3, Q4 and let us assume that m=2. We need 4*8 bits in order to represent this genoma. A possible configuration is 0101 1100 corresponding to the formula: Q2 and Q4 or Q1 and Q2. FITTING FUNCTION: The fitting function is the editing distance between the truth table of the formula represented in the genoma and the truth table of the target query. PENALIZATIONS: A number of penalizations can be proposed, one that seems plausible is one which penalizes: the use of disjunctions, and the use of many different queries. For instance, let us denote with k<=m the number of disjunctions in the formula and with t<=n the number of different queries exploited. A

possible fitting function may be: -P0∗ham - P1∗k - P2∗t where ham is the Hamming distance, and P0,P1,P2 are the penalities associated to ham,k, and t respectively.

TODO:

1) Memorizzare da qualche parte (ParsedMinerule?) le sostituzioni effettuate sulla rule attribute delle minerule processate. Servira' per poi "tornare" indietro quando restituiremo i risultati.

2) Eliminare la cal quando inclusa nella gal. Le eventuali cluster conditions, vengono poste in and con le mining cond. (nota: bisogna ri-eseguire

this file contains the basic class for modelling Predicate in disjunction form

As a way to optimize queries, we translate each value in the DB to numerical identifiers. This class perform the translation and provide a way to access to the resulting tables and values by means of the translateTable method.

We refer to tables containing untranslated values as "original tables" while the table generated by this class will be referred to as "translated tables". Analogously, we refer to untranslated values as "original values" and to new ones created by this class as "translated values".

In order to fully abstract the access to the new tables, we provide a translation function for the table column names as well (at the present time the functions implementing this functionality simply returns its input, but this behavior may change in the future).

Author

Roberto Esposito (esposito@di.unito.it)

Provides an implementation of a TranslationManager using only standard SQL primitives.

Author

Roberto Esposito (esposito@di.unito.it)

Here the enumerated type AlgorithmTypes is defined along with few utils to work with it. Even though Algorithm↩ Types should be located in the directory Algorithms, this is not so for a very good reason: MineruleOptions depends on it. If it were in Algorithms then anyone needing libUtils (i.e., everybody) would be forced to link Algorithms (which is not always the case). Two alternatives are possible (better said: ``I can see two alternatives''): 1) to duplicate the information and then convert the types back and forth in order to keep the variables refferring to them updated. 2) to move this header and the associated implementation file into Utils/

5.1.2 Typedef Documentation

5.1.2.1 Bits

```
typedef unsigned int minerule::Bits
```

Definition at line 43 of file Bitstring.hpp.

5.1.2.2 boolean

```
typedef int minerule::boolean
```

Definition at line 42 of file Bitstring.hpp.

5.1.2.3 EncodingBaseType

```
typedef unsigned int minerule::EncodingBaseType
```

Definition at line 28 of file ExpressionNFCoder.hpp.

5.1.2.4 GidList

```
typedef MapElement minerule::GidList
```

Definition at line 56 of file Bodymap.hpp.

5.1.2.5 GroupMap

```
typedef std::map<long, int> minerule::GroupMap
```

Definition at line 19 of file STSMiner.hpp.

5.1.2.6 GroupPair

```
typedef std::pair<long, int> minerule::GroupPair
```

Definition at line 20 of file STSMiner.hpp.

5.1.2.7 ItemSet

```
typedef std::vector<ItemType> minerule::ItemSet
```

this type is the one used by procedures which deal with itemsets and rules

Definition at line 37 of file ItemType.hpp.

5.1.3 Enumeration Type Documentation

5.1.3.1 AlgorithmTypes

```
enum minerule::AlgorithmTypes
```

Enumerator

ATNone	
ATBFSWithGidsNoCross	
ATBFSWithGidsAndCross	
ATCare	
ATConstrainedItemsets	
ATEnd	

Definition at line 44 of file AlgorithmTypes.hpp.

```
00044                {
00045     ATNone=0, // dummy algorithm type
00046     ATBFSWithGidsNoCross,
00047     ATBFSWithGidsAndCross,
00048     ATCare,
00049     ATConstrainedItemsets,
00050     ATEnd // dummy algorithm type do not add algorithms
00051           // below this element
00052   } AlgorithmTypes;
```

5.1.3.2 MineruleErrors

```
enum minerule::MineruleErrors
```

Enumerator

MR_ERROR_NO_ERROR	
MR_ERROR_UNKNOWN	
MR_ERROR_INTERNAL	
MR_ERROR_INPUT_FILE_NOT_FOUND	
MR_ERROR_OUTPUT_FILE_PROBLEM	
MR_ERROR_NO_MINERULE_SPECIFIED	
MR_ERROR_NO_OPTIONFILE_SPECIFIED	
MR_ERROR_OPTION_CONFIGURATION	
MR_ERROR_MINERULE_ALREADY_EXISTS	
MR_ERROR_DATABASE_ERROR	
MR_ERROR_OPTION_PARSING	
MR_ERROR_MINERULETEXT_PARSING	
MR_ERROR_CATALOGUE_ERROR	
MR_ERROR_OPTIMIZER_ERROR	
MR_ERROR_INSTALLATION_PROBLEM	
MR_ERROR_SAFETY_PROBLEM	

Definition at line 20 of file MineruleErrors.hpp.

```
00020                {
00021     MR_ERROR_NO_ERROR=0,
00022     MR_ERROR_UNKNOWN,
00023     MR_ERROR_INTERNAL,
00024     MR_ERROR_INPUT_FILE_NOT_FOUND,
00025     MR_ERROR_OUTPUT_FILE_PROBLEM,
00026     MR_ERROR_NO_MINERULE_SPECIFIED,
00027     MR_ERROR_NO_OPTIONFILE_SPECIFIED,
00028     MR_ERROR_OPTION_CONFIGURATION,
00029     MR_ERROR_MINERULE_ALREADY_EXISTS,
00030     MR_ERROR_DATABASE_ERROR,
00031     MR_ERROR_OPTION_PARSING,
```

```
00032     MR_ERROR_MINERULETEXT_PARSING,
00033     MR_ERROR_CATALOGUE_ERROR,
00034     MR_ERROR_OPTIMIZER_ERROR,
00035     MR_ERROR_INSTALLATION_PROBLEM,
00036     MR_ERROR_SAFETY_PROBLEM,
00037   } MineruleErrors;
```

5.1.3.3 MiningTasks

```
enum minerule::MiningTasks
```

Enumerator

MTMineRules	
MTMineItemsets	
MTMineSequences	
MTEnd	

Definition at line 36 of file AlgorithmTypes.hpp.

```
00036               {
00037     MTMineRules=0,
00038     MTMineItemsets,
00039     MTMineSequences,
00040     MTEnd
00041   } MiningTasks;
```

5.1.4 Function Documentation

5.1.4.1 abort_handler()

```
void minerule::abort_handler (
            int s )
```

Definition at line 11 of file Progress.cpp.

```
00011                       {
00012         std::string msg = "Ctrl-C or Ctrl-Z caught \e[?25h";
00013         write(1, msg.c_str(), msg.size());
00014   exit(1);
00015 }
```

5.1.4.2 algorithmTypeToString()

```
const std::string & minerule::algorithmTypeToString (
            AlgorithmTypes t )
```

Definition at line 73 of file AlgorithmTypes.cpp.

```
00073                                                    {
00074     return algoNames[t];
00075   }
```

5.1.4.3 angledSpeenWheelUpdateHandler()

```
void minerule::angledSpeenWheelUpdateHandler (
            int n,
            int count,
            int num )
```

Definition at line 51 of file Progress.cpp.

```
00051                                                                    {
00052   static unsigned int step = 0;
00053   static char spinWheel[] = { '^', '>', 'v', '<' };
00054
00055         static std::ostream& logger =
00056                 *MineruleOptions::getSharedOptions()
00057                         .getLogStreamObj()
00058                         .getLogger()
00059                         .getStream();
00060
00061         if(Progress::handleStartStop(logger, n))
00062                 return;
00063
00064   if((count+n) % num) {
00065     logger « "\b" « spinWheel[step++ % 4];
00066                 logger.flush();
00067   }
00068 }
```

5.1.4.4 barSpeenWheelUpdateHandler()

```
void minerule::barSpeenWheelUpdateHandler (
            int n,
            int count,
            int num )
```

Definition at line 70 of file Progress.cpp.

```
00070                                                                    {
00071   static unsigned int step = 0;
00072   static char spinWheel[] = { '|', '/', '-', '\\' };
00073
00074         static std::ostream& logger =
00075                 *MineruleOptions::getSharedOptions()
00076                         .getLogStreamObj()
00077                         .getLogger()
00078                         .getStream();
00079
00080         if(Progress::handleStartStop(logger, n))
00081                 return;
00082
00083   if((count+n) % num) {
00084     logger « "\b" « spinWheel[step++ % 4];
00085                 logger.flush();
00086   }
00087 }
```

5.1.4.5 buildStringWithListOfAlgorithmTypes()

```
std::string minerule::buildStringWithListOfAlgorithmTypes ( )
```

Definition at line 87 of file AlgorithmTypes.cpp.

```
00087                                                {
00088     std::string l;
00089     AlgorithmTypes algoType;
00090
00091     for(algoType=ATNone; algoType<ATEnd; algoType=static_cast<AlgorithmTypes>(algoType+1)) {
```

```
00092       if(algoType!=ATNone)
00093                   l+=",";
00094
00095       l+=algoNames[algoType];
00096     }
00097
00098     return l;
00099   }
```

5.1.4.6 buildStringWithListOfMiningTasks()

```
std::string minerule::buildStringWithListOfMiningTasks ( )
```

Definition at line 51 of file AlgorithmTypes.cpp.

```
00051                                                {
00052     std::string l;
00053     MiningTasks task;
00054
00055     for(task=MTMineRules; task<MTEnd; task=static_cast<MiningTasks>(task+1)) {
00056       if(task!=MTMineRules)
00057                   l+=",";
00058       l+=taskNames[task];
00059     }
00060
00061     return l;
00062   }
```

5.1.4.7 clockHandsUpdateHandler()

```
void minerule::clockHandsUpdateHandler (
            int n,
            int count,
            int num )
```

Definition at line 99 of file Progress.cpp.

```
00099                                                                {
00100   static unsigned long int step = 0;
00101   static std::string spinWheel[] =
00102           { "", "", "", "",
00103                   "", "", "", "",
00104                   "", "", "", "" };
00105
00106         static std::ostream& logger =
00107                 *MineruleOptions::getSharedOptions()
00108                         .getLogStreamObj()
00109                         .getLogger()
00110                         .getStream();
00111
00112         if(Progress::handleStartStop(logger, n)) {
00113                 return;
00114         }
00115
00116   if((count+n) % num == 0) {
00117     logger « "\b\b" « spinWheel[step++ % 12] « " ";
00118                 logger.flush();
00119   }
00120 }
```

5.1.4.8 floatFromText()

```
float minerule::floatFromText (
            char * text )
```

Definition at line 202 of file ParsedMinerule.cpp.

```
00202                                   {
00203                 char* endptr;
00204                 float tmp = strtof(text, &endptr);
00205
00206                 assert( endptr!=text );
00207                 assert( errno!=ERANGE );
00208
00209                 return tmp;
00210         }
```

5.1.4.9 getParserOutputObj()

```
ParsedMinerule & minerule::getParserOutputObj ( )
```

Definition at line 42 of file ParserLibrary.cpp.

```
00042                                           {
00043     assert(outputObj!=NULL);
00044     return *outputObj;
00045   }
```

5.1.4.10 init_mrparser()

```
void minerule::init_mrparser ( )
```

5.1.4.11 initializeOptionsFromFile()

```
void minerule::initializeOptionsFromFile (
            MineruleOptions & mrOpts,
            FILE * file )
```

Definition at line 33 of file OptionParserLib.cpp.

```
00033                                                                           {
00034     YY_BUFFER_STATE buf = yy_new_buffer(file,YY_BUF_SIZE);
00035     OP_switch_to_buffer(buf);
00036     context.push_back(&mrOpts);
00037     OPparse();
00038     OP_delete_buffer(buf);
00039   }
```

5.1.4.12 initializeOptionsFromString()

```
void minerule::initializeOptionsFromString (
            MineruleOptions & mrOpts,
            std::string str )
```

Definition at line 41 of file OptionParserLib.cpp.

```
00041                                                                          {
00042     YY_BUFFER_STATE buf = OP_scan_string(str.c_str());
00043     context.push_back(&mrOpts);
00044     OPparse();
00045     OP_delete_buffer(buf);
00046   }
```

5.1.4.13 install_abort_handler()

```
void minerule::install_abort_handler ( )
```

Definition at line 18 of file Progress.cpp.

```
00018                                {
00019         static bool handlerInstalled = false;
00020         if(handlerInstalled) {
00021                 return;
00022         }
00023
00024   struct sigaction sigIntHandler;
00025
00026   sigIntHandler.sa_handler = abort_handler;
00027   sigemptyset(&sigIntHandler.sa_mask);
00028   sigIntHandler.sa_flags = 0;
00029
00030    handlerInstalled = !sigaction(SIGINT, &sigIntHandler, NULL);
00031 }
```

5.1.4.14 logRespectingMaxLogLength()

```
void minerule::logRespectingMaxLogLength (
            std::ostream &(*)(void) logger,
            size_t indentLen,
            const std::string & msg )
```

Definition at line 24 of file MineruleLogs.cpp.

```
00024
          {
00025
00026                 std::vector<std::string>* chunks = StringUtils::splitToLength(msg,
      MAX_LOG_LENGTH-indentLen);
00027
00028                 std::vector<std::string>::const_iterator it= chunks->begin();
00029                 if( it != chunks->end() ) {
00030                         logger() « *it « std::endl;
00031                 }
00032
00033                 ++it;
00034
00035                 for( ; it!=chunks->end(); ++it ) {
00036                         logger() « StringUtils::toGreen("  +") « *it « std::endl;
00037                 }
00038
00039                 delete chunks;
00040         }
```

5.1.4.15 me_error_begin()

```
int minerule::me_error_begin ( )
```

Definition at line 45 of file MineruleErrors.cpp.

```
00045                          {
00046   return MR_ERROR_NO_ERROR;
00047 }
```

5.1.4.16 me_error_end()

```
int minerule::me_error_end ( )
```

Definition at line 49 of file MineruleErrors.cpp.

```
00049                        {
00050   return (int)MR_ERROR_SAFETY_PROBLEM + 1;
00051 }
```

5.1.4.17 me_error_name()

```
const char * minerule::me_error_name (
            MineruleErrors me )
```

Definition at line 40 of file MineruleErrors.cpp.

```
00040                                                {
00041   assert( me>=me_error_begin() && me<me_error_end() );
00042   return MineruleErrorsNames[me];
00043 }
```

5.1.4.18 miningTaskToString()

```
const std::string & minerule::miningTaskToString (
            MiningTasks mt )
```

Definition at line 35 of file AlgorithmTypes.cpp.

```
00035                                                    {
00036     return taskNames[mt];
00037   }
```

5.1.4.19 MRDebug() [1/2]

```
std::ostream & minerule::MRDebug ( )
```

Definition at line 72 of file MineruleLogs.cpp.

```
00072                              {
00073       return MineruleOptions::getSharedOptions().getDebugStream();
00074     }
```

5.1.4.20 MRDebug() [2/2]

```
void minerule::MRDebug (
            const std::string & msg )
```

Definition at line 76 of file MineruleLogs.cpp.

```
00076                                           {
00077         size_t indentLength =
      MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().getIndentLen();
00078                 logRespectingMaxLogLength(MRDebug, indentLength, msg);
00079         }
```

5.1.4.21 MRDebugPop()

```
void minerule::MRDebugPop ( )
```

Definition at line 111 of file MineruleLogs.cpp.

```
00111                       {
00112       MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().pop();
00113     }
```

5.1.4.22 MRDebugPush()

```
void minerule::MRDebugPush (
            const std::string & descr )
```

Definition at line 107 of file MineruleLogs.cpp.

```
00107                                            {
00108       MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().push(descr);
00109     }
```

5.1.4.23 MRErr() [1/2]

```
std::ostream & minerule::MRErr ( )
```

Definition at line 52 of file MineruleLogs.cpp.

```
00052                         {
00053       return MineruleOptions::getSharedOptions().getErrStream();
00054     }
```

5.1.4.24 MRErr() [2/2]

```
void minerule::MRErr (
            const std::string & msg )
```

Definition at line 56 of file MineruleLogs.cpp.

```
00056                                         {
00057                 size_t indentLength =
      MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().getIndentLen();
00058                 logRespectingMaxLogLength(MRErr, indentLength, msg);
00059         }
```

5.1.4.25 MRErrPop()

```
void minerule::MRErrPop ( )
```

Definition at line 95 of file MineruleLogs.cpp.

```
00095                     {
00096       MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().pop();
00097     }
```

5.1.4.26 MRErrPush()

```
void minerule::MRErrPush (
            const std::string & descr )
```

Definition at line 91 of file MineruleLogs.cpp.

```
00091                                          {
00092       MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().push(descr);
00093     }
```

5.1.4.27 MRLog() [1/2]

```
std::ostream & minerule::MRLog ( )
```

Definition at line 43 of file MineruleLogs.cpp.

```
00043                           {
00044       return MineruleOptions::getSharedOptions().getLogStream();
00045     }
```

5.1.4.28 MRLog() [2/2]

```
void minerule::MRLog (
            const std::string & msg )
```

Definition at line 47 of file MineruleLogs.cpp.

```
00047                                      {
00048                 size_t indentLength =
      MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().getIndentLen();
00049                 logRespectingMaxLogLength(MRLog, indentLength ,msg);
00050         }
```

5.1.4.29 MRLogGetNewID()

```
int minerule::MRLogGetNewID ( )
```

5.1.4.30 MRLogPop()

```
void minerule::MRLogPop ( )
```

Definition at line 87 of file MineruleLogs.cpp.

```
00087                     {
00088       MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().pop();
00089     }
```

5.1.4.31 MRLogPush()

```
void minerule::MRLogPush (
             const std::string & descr )
```

Definition at line 83 of file MineruleLogs.cpp.

```
00083                                          {
00084       MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().push(descr);
00085     }
```

5.1.4.32 MRLogShowMeasurements()

```
void minerule::MRLogShowMeasurements ( )
```

Definition at line 123 of file MineruleLogs.cpp.

```
00123                                  {
00124               MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().logMeasurements();
00125         }
```

5.1.4.33 MRLogStartMeasuring()

```
void minerule::MRLogStartMeasuring (
             const std::string & description )
```

Definition at line 116 of file MineruleLogs.cpp.

```
00116                                                    {
00117         MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().startMeasuring(description);
00118     }
```

5.1.4.34 MRLogStopMeasuring()

```
void minerule::MRLogStopMeasuring (
             const std::string & description )
```

Definition at line 119 of file MineruleLogs.cpp.

```
00119                                                   {
00120         MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().stopMeasuring(description);

00121     }
```

5.1.4.35 MRWarn() [1/2]

```
std::ostream & minerule::MRWarn ( )
```

Definition at line 62 of file MineruleLogs.cpp.

```
00062                           {
00063       return MineruleOptions::getSharedOptions().getWarnStream();
00064     }
```

5.1.4.36 MRWarn() [2/2]

```
void minerule::MRWarn (
            const std::string & msg )
```

Definition at line 66 of file MineruleLogs.cpp.

```
00066                                       {
00067             size_t indentLength =
       MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().getIndentLen();
00068                 logRespectingMaxLogLength(MRWarn, indentLength, msg);
00069         }
```

5.1.4.37 MRWarnPop()

```
void minerule::MRWarnPop ( )
```

Definition at line 103 of file MineruleLogs.cpp.

```
00103                       {
00104       MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().pop();
00105     }
```

5.1.4.38 MRWarnPush()

```
void minerule::MRWarnPush (
            const std::string & descr )
```

Definition at line 99 of file MineruleLogs.cpp.

```
00099                                         {
00100       MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().push(descr);
00101     }
```

5.1.4.39 operator<<() [1/14]

```
std::ostream & minerule::operator<< (
            std::ostream & o,
            const SourceRowColumnIds & rowDes )  [inline]
```

Definition at line 117 of file SourceRowColumnIds.hpp.

```
00117                                                                                {
00118   // o « "test";
00119         // o « "head:";
00120   copy( rowDes.headElems.begin(), rowDes.headElems.end(), std::ostream_iterator<int>(o, " ") );
00121         // o « " - ";
00122         // o « "body:";
00123   copy( rowDes.bodyElems.begin(), rowDes.bodyElems.end(), std::ostream_iterator<int>(o, " ") );
00124         return o;
00125 }
```

5.1.4.40 operator<<() [2/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const EncodedNF & nf )
```

Definition at line 225 of file ExpressionNFCoder.cpp.

```
00225                                                                   {
00226     EncodedNFIterator it(nf);
00227     it++;
00228     while(it.ok()) {
00229       os « *it « ".";
00230       it++;
00231     }
00232
00233     return os;
00234   }
```

5.1.4.41 operator<<() [3/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const Interval & i )
```

Definition at line 30 of file Interval.cpp.

```
00030                                                                   {
00031     if( i.lwrOpen || i.openPoints.find(i.lwr)!=i.openPoints.end()) os « "("; else os « "[";
00032     os « i.lwr « "," « i.upp;
00033     if( i.uppOpen || i.openPoints.find(i.upp)!=i.openPoints.end()) os « ")"; else os « "]";
00034
00035     return os;
00036   }
```

5.1.4.42 operator<<() [4/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const ItemType & it )  [inline]
```

Definition at line 215 of file ItemType.hpp.

```
00215                                                                   {
00216                 if(it.el==NULL)
00217                         os«"NULL";
00218                 else
00219                         os « *it.el;
00220
00221                 return os ;
00222         }
```

5.1.4.43 operator<<() [5/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const MinMaxPair & p )  [inline]
```

Definition at line 104 of file MinMaxPair.hpp.

```
00104                                                                   {
00105     os « "MinMax(" « p.getMin() « "," « p.getMax() « ")";
00106     return os;
00107   }
```

5.1.4.44 operator<<() [6/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const ParsedMinerule & mr )
```

Definition at line 128 of file ParsedMinerule.cpp.

```
00128                                                                   {
00129                 os « "ParsedMinerule" « std::endl;
00130                 os « " - ba:";
00131                 copy( mr.ba.begin(), mr.ba.end(), std::ostream_iterator<std::string>(os, " "));
00132                 os « std::endl;
00133
00134         os « " - oa:";
00135         copy( mr.oa.begin(), mr.oa.end(), std::ostream_iterator<std::string>(os, " "));
00136         os « std::endl;
00137
00138                 os « " - ha:";
00139                 copy( mr.ha.begin(), mr.ha.end(), std::ostream_iterator<std::string>(os, " "));
00140                 os « std::endl;
00141
00142                 os « " - ga:";
00143                 copy( mr.ga.begin(), mr.ga.end(), std::ostream_iterator<std::string>(os, " "));
00144                 os « std::endl
00145                         « " - ca:";
00146                 copy( mr.ca.begin(), mr.ca.end(), std::ostream_iterator<std::string>(os, " "));
00147                 os « std::endl
00148                         « " - ra:";
00149                 copy( mr.ra.begin(), mr.ra.end(), std::ostream_iterator<std::string>(os, " "));
00150                 os « std::endl
00151                         « " = mc:";
00152                 print_OR_list(os,mr.mc);
00153                 os « std::endl
00154                         « " = gc:";
00155                 print_OR_list(os,mr.gc);
00156                 os « std::endl
00157                         « " = cc:";
00158                 print_OR_list(os,mr.cc);
00159                 os « std::endl;
00160
00161                 os « " ! clust. agg. list:";
00162                 copy( mr.c_aggr_list.begin(),
00163                         mr.c_aggr_list.end(),
00164                                 std::ostream_iterator<std::string>(os, " "));
00165                 os « std::endl;
00166
00167                 os « " * sup:" « mr.sup « std::endl;
00168                 os « " * conf:" « mr.conf « std::endl;
00169                 os « " & tautologies: "«mr.tautologies « std::endl;
00170                 os « " & body coinc head:"«mr.body_coincident_head « std::endl;
00171                 os « " tab_source:" « mr.tab_source « std::endl;
00172                 os « " tab_result:" « mr.tab_result « std::endl;
00173                 os « "ParsedMinerule - end" « std::endl;
00174
00175                 return os;
00176         }
```

5.1.4.45 operator<<() [7/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const PredConjunction & p )  [inline]
```

Definition at line 226 of file Predicate.hpp.

```
00226                                                                   {
00227     PredConjunction::const_iterator it2;
00228     for(it2=p.begin(); it2!=p.end(); it2++) {
00229       if(it2!=p.begin()) {
00230         os « " AND ";
00231       }
00232
00233       os « **it2;
00234     }
00235
00236     return os;
00237   }
```

5.1.4.46 operator<<() [8/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const Predicate & p )  [inline]
```

Definition at line 239 of file Predicate.hpp.

```
00239                                                                    {
00240     Predicate::const_iterator it;
00241     for( it=p.begin(); it!=p.end(); it++) {
00242       if(it!=p.begin()) {
00243         os « " OR ";
00244       }
00245
00246       os « **it;
00247     }
00248
00249     return os;
00250   }
```

5.1.4.47 operator<<() [9/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const SimplePredicate & s )  [inline]
```

Definition at line 221 of file Predicate.hpp.

```
00221                                                                          {
00222     os « s.getVal1() « s.getOp() « s.getVal2();
00223     return os;
00224   }
```

5.1.4.48 operator<<() [10/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const SourceRow & sr )
```

Definition at line 93 of file SourceRow.cpp.

```
00094                                                {
00095   os « "group:" « sr.getGroup() « std::endl;
00096   os « " clusterBody:" « sr.getClusterBody() « std::endl;
00097   os « " body:" « sr.getBody() « std::endl;
00098   os « " clusterHead:" « sr.getClusterHead() « std::endl;
00099   os « " head:" « sr.getHead() « std::endl;
00100
00101   return os;
00102 }
```

5.1.4.49 operator<<() [11/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const SourceRowAttribute & attr )  [inline]
```

Definition at line 313 of file SourceRowAttribute.hpp.

```
00314                                                                {
00315   attr.operator«(os);
00316   return os;
00317 }
```

5.1.4.50 operator<<() [12/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const SourceRowElement & el )  [inline]
```

Definition at line 198 of file SourceRowElement.hpp.

```
00198                                                                    {
00199   el.operator«(os);
00200   return os;
00201 }
```

5.1.4.51 operator<<() [13/14]

```
std::ostream & minerule::operator<< (
            std::ostream & os,
            const VarSet & vset )  [inline]
```

Definition at line 154 of file VarSet.hpp.

```
00154                                                          {
00155   for(size_t i=0; i<vset.size(); i++) {
00156     if(vset.getVar(i))
00157       os « "1";
00158     else
00159       os « "0";
00160   }
00161   return os;
00162 }
```

5.1.4.52 operator<<() [14/14]

```
std::ostream & minerule::operator<< (
            std::ostream & out,
            const BitString & bs )
```

Definition at line 318 of file Bitstring.cpp.

```
00319 {
00320 for (int i=0; i<bs.n; i++) out « (bs.test(i) ? '1' : '0');
00321 return out;
00322 }
```

5.1.4.53 operator>>()

```
std::istream & minerule::operator>> (
            std::istream & in,
            BitString & bs )
```

Definition at line 328 of file Bitstring.cpp.

```
00329 {
00330 std::ios::iostate St = std::ios::goodbit;
00331 boolean Chg = false;
00332 int C = in.rdbuf()->sgetc();
00333 bs.reset();
00334 while (C != EOF && C == ' ') C = in.rdbuf()->snextc();
00335 for (size_t M = 0; true ; C = in.rdbuf()->snextc(), ++M)
```

```
00336 {
00337         if (C == EOF) {St |= std::ios::eofbit; break; }
00338         else if (C != '0' && C != '1') break;
00339         //else if (_X.allocation() <= _X.length())
00340         //      {_St |= ios::failbit;
00341         //      break; }
00342         else { bs.set(M,C=='1'); Chg = true; }
00343 }
00344 //if (!_Chg)  _St |= ios::failbit;
00345 in.setstate(St);
00346 return (in);
00347 }
```

5.1.4.54 parseMinerule()

```
void minerule::parseMinerule (
             std::string minerule_text,
             ParsedMinerule & output )
```

Definition at line 18 of file ParserLibrary.cpp.

```
00018                                                                    {
00019     outputObj = &output;
00020
00021     const FILE* logfile =
00022       minerule::MineruleOptions::getSharedOptions().getParsers().getLogFILE();
00023
00024     mrout=logfile;
00025     YY_BUFFER_STATE buf = mr_scan_string(minerule_text.c_str());
00026
00027     init_mrparser();
00028
00029     try {
00030        mrparse();
00031     } catch (MineruleException& m) {
00032      std::string error;
00033       error = std::string(m.unformattedMessage())+ " Original minerule was:"+minerule_text;
00034       throw MineruleException(MR_ERROR_MINERULETEXT_PARSING, error);
00035     }
00036
00037     mr_delete_buffer(buf);
00038
00039     outputObj = NULL;
00040   }
```

5.1.4.55 popOptionClassFromContext()

```
void minerule::popOptionClassFromContext ( )
```

Definition at line 54 of file OptionParserLib.cpp.

```
00054                                          {
00055     assert(context.size()>=1);
00056     context.pop_back();
00057   }
```

5.1.4.56 pushOptionClassIntoContext()

```
void minerule::pushOptionClassIntoContext (
             const std::string & oclass )
```

Definition at line 48 of file OptionParserLib.cpp.

```
00048                                                                {
00049     /* Here we have to push_back OptionBase pointers */
00050     assert(context.size()>0);
00051     context.push_back( &context.back()->subclassForName(oclass) );
00052   }
```

5.1.4.57 removeAllNodes()

```
void minerule::removeAllNodes (
            list_AND_node *& l )
```

Definition at line 853 of file QueryNormalizer.cpp.

```
00853                                                {
00854         while( l!=NULL ) {
00855             removeNode(&l);
00856         }
00857     }
```

5.1.4.58 removeNode()

```
void minerule::removeNode (
            list_AND_node ** l )
```

Definition at line 843 of file QueryNormalizer.cpp.

```
00843                                                {
00844         list_AND_node* node = *l;
00845         *l = node->next;
00846         delete node->sp->val1;
00847         delete node->sp->val2;
00848         delete node->sp->op;
00849         delete node->sp;
00850         delete node;
00851     }
```

5.1.4.59 setOption()

```
void minerule::setOption (
            const std::string & name,
            const std::string & value )
```

Definition at line 59 of file OptionParserLib.cpp.

```
00059                                                                {
00060     assert(context.size()>=1);
00061
00062     context.back()->setOption(name,value);
00063   }
```

5.1.4.60 stringToAlgorithmType()

```
AlgorithmTypes minerule::stringToAlgorithmType (
            const std::string & s )
```

Definition at line 77 of file AlgorithmTypes.cpp.

```
00077                                                        {
00078     AlgorithmTypes algoType;
00079     for( algoType=ATNone; algoType<ATEnd; algoType=static_cast<AlgorithmTypes>(algoType+1) ) {
00080       if(algoNames[algoType]==s)
00081                 return algoType;
00082       }
00083
00084     throw MineruleException( MR_ERROR_INTERNAL, "Cannot convert "+s+" to a valid AlgorithmType" );
00085   }
```

5.1.4.61 stringToMiningTask()

```
MiningTasks minerule::stringToMiningTask (
            const std::string & s )
```

Definition at line 40 of file AlgorithmTypes.cpp.

```
00040                                                       {
00041     MiningTasks task;
00042     for(task=MTMineRules; task<MTEnd; task=static_cast<MiningTasks>(task+1) ) {
00043       if( taskNames[task]==s )
00044                   return task;
00045     }
00046
00047
00048     throw MineruleException( MR_ERROR_INTERNAL, "Cannot convert "+s+" to a valid MiningTask" );
00049   }
```

5.1.4.62 stringWithListOfAlgorithmTypes()

```
std::string minerule::stringWithListOfAlgorithmTypes ( )
```

Definition at line 101 of file AlgorithmTypes.cpp.

```
00101                                                {
00102     return buildStringWithListOfAlgorithmTypes();
00103   }
```

5.1.4.63 stringWithListOfMiningTasks()

```
const std::string & minerule::stringWithListOfMiningTasks ( )
```

Definition at line 64 of file AlgorithmTypes.cpp.

```
00064                                                    {
00065     static std::string l = buildStringWithListOfMiningTasks();
00066
00067     return l;
00068   }
```

5.1.4.64 tickNumberUpdateHandler()

```
void minerule::tickNumberUpdateHandler (
            int n,
            int count,
            int num )
```

Definition at line 89 of file Progress.cpp.

```
00089                                                          {
00090         if(n<0) {
00091                 return;
00092         }
00093
00094         if((count+n) % num == 0) {
00095                 MRLog() « count+n « std::endl;
00096         }
00097 }
```

5.1.4.65 trim()

```
std::string minerule::trim (
            std::string text )
```

Definition at line 213 of file ParsedMinerule.cpp.

```
00213                                   {
00214                 size_t begSpaces = text.find_first_not_of(" ");
00215                 if( begSpaces == text.npos ) // all elements are spaces
00216                         return "";
00217
00218                 text.erase( 0, begSpaces );
00219
00220                 size_t endSpaces = text.find_last_not_of(" ");
00221                 if( endSpaces == text.npos || endSpaces+1 > text.length() )
00222                         return text;
00223
00224                 text.erase( endSpaces+1, text.length()-endSpaces );
00225                 return text;
00226         }
```

5.1.5 Variable Documentation

5.1.5.1 context

```
std::vector<MineruleOptions::OptionBase*> minerule::context
```

Definition at line 29 of file OptionParserLib.cpp.

5.1.5.2 DB_LEVEL_ALL

```
const int minerule::DB_LEVEL_ALL = 1
```

Definition at line 38 of file Bitstring.hpp.

5.1.5.3 DEBUG_LEVEL

```
const int minerule::DEBUG_LEVEL = 0
```

Definition at line 37 of file Bitstring.hpp.

5.1.5.4 DISJ_LOSS

```
const int minerule::DISJ_LOSS = 1
```

Definition at line 29 of file GAQueryCombinator.cpp.

5.1.5.5 HAM_LOSS

```
const int minerule::HAM_LOSS = 10
```

Definition at line 28 of file GAQueryCombinator.cpp.

5.1.5.6 MineruleErrorsNames

```
const char* minerule::MineruleErrorsNames[]
```

Initial value:

```
= {
  "MR_ERROR_NO_ERROR",
  "MR_ERROR_UNKNOWN",
  "MR_ERROR_INTERNAL",
  "MR_ERROR_INPUT_FILE_NOT_FOUND",
  "MR_ERROR_OUTPUT_FILE_PROBLEM",
  "MR_ERROR_NO_MINERULE_SPECIFIED",
  "MR_ERROR_NO_OPTIONFILE_SPECIFIED",
  "MR_ERROR_OPTION_CONFIGURATION",
  "MR_ERROR_MINERULE_ALREADY_EXISTS",
  "MR_ERROR_DATABASE_ERROR",
  "MR_ERROR_OPTION_PARSING",
  "MR_ERROR_MINERULETEXT_PARSING",
  "MR_ERROR_CATALOGUE_ERROR",
  "MR_ERROR_OPTIMIZER_ERROR",
  "MR_ERROR_INSTALLATION_PROBLEM",
  "MR_ERROR_SAFETY_PROBLEM"
}
```

Definition at line 21 of file MineruleErrors.cpp.

5.1.5.7 outputObj

```
ParsedMinerule* minerule::outputObj
```

Definition at line 16 of file ParserLibrary.cpp.

5.1.5.8 QUERY_LOSS

```
const int minerule::QUERY_LOSS = 1
```

Definition at line 30 of file GAQueryCombinator.cpp.

5.2 mrc Namespace Reference

Data Structures

- class Exception
- class Options
- class Printer

Enumerations

- enum Results {
  SUCCESS =0 , NOTHING_TO_DO , QUERY_NAME_FOUND , QUERY_NAME_NOT_FOUND , CATALOGUE_NOT_INSTALLED , ERROR_OPTION_PARSING , ERROR_CANNOT_OPEN_OPTION_FILE , ERROR_EXCEPTION_THROWN }

Functions

- void doLoadOptions (const char *fname, minerule::MineruleOptions &opt)
- void parseOptions (int argc, char **argv, minerule::MineruleOptions &mopt, Options &popt)
- void printVersion ()
- void printHelp (int argc, char **argv)
- int checkQueryName (const Options &options)
- void printMRQueryList (const Options &options)
- void deleteQuery (const Options &options)
- Results checkCatalogue ()
- Results installCatalogue ()
- Results uninstallCatalogue ()
- Results addDerivedQuery (const Options &options)
- int performCommands (Options &options)

5.2.1 Enumeration Type Documentation

5.2.1.1 Results

enum mrc::Results

Enumerator

SUCCESS	
NOTHING_TO_DO	
QUERY_NAME_FOUND	
QUERY_NAME_NOT_FOUND	
CATALOGUE_NOT_INSTALLED	
ERROR_OPTION_PARSING	
ERROR_CANNOT_OPEN_OPTION_FILE	
ERROR_EXCEPTION_THROWN	

Definition at line 21 of file ErrorCodes.hpp.

```
00021                     {
00022           SUCCESS=0,
00023           NOTHING_TO_DO,
00024           QUERY_NAME_FOUND,
00025           QUERY_NAME_NOT_FOUND,
00026           CATALOGUE_NOT_INSTALLED,
00027           ERROR_OPTION_PARSING,
00028           ERROR_CANNOT_OPEN_OPTION_FILE,
00029           ERROR_EXCEPTION_THROWN,
00030         } Results;
```

5.2.2 Function Documentation

5.2.2.1 addDerivedQuery()

```
Results mrc::addDerivedQuery (
            const Options & options )
```

Definition at line 264 of file MRCUtils.cpp.

```
00264                                                   {
00265                 minerule::OptimizerCatalogue::addDerivedResult(options.getOriginalQuery(),
      options.getDerivedQuery());
00266                 return SUCCESS;
00267         }
```

5.2.2.2 checkCatalogue()

```
Results mrc::checkCatalogue ( )
```

Definition at line 210 of file MRCUtils.cpp.

```
00210                                   {
00211                 if(minerule::OptimizerCatalogue::checkInstallation()) {
00212                         minerule::MRLog() « "The catalogue appears to be installed properly." «
      std::endl;
00213                         return SUCCESS;
00214                 } else {
00215                         minerule::MRLog() « "The catalogue is " « minerule::StringUtils::toRed("NOT")
      «" installed" « std::endl;
00216                         exit(CATALOGUE_NOT_INSTALLED);
00217                 }
00218         }
```

5.2.2.3 checkQueryName()

```
int mrc::checkQueryName (
            const Options & options )
```

Definition at line 180 of file MRCUtils.cpp.

```
00180                                                   {
00181                 const std::string& tablename =
00182                         options.getSearchParam(Options::QryName);
00183
00184                 try {
00185                         Printer printer(std::cout, options);
00186                         minerule::CatalogueInfo info;
00187                         minerule::OptimizerCatalogue::getMRQueryInfo(tablename,info,
      options.getListFormat().size);
00188                         printer.print(info);
00189                 } catch (minerule::MineruleException& mr) {
00190                         std::cout « "The query you specified is NOT present in the catalogue." «
      std::endl;
00191                         return mrc::QUERY_NAME_NOT_FOUND;
00192                 }
00193
00194                 return mrc::QUERY_NAME_FOUND;
00195         }
```

5.2.2.4 deleteQuery()

```
void mrc::deleteQuery (
            const Options & options )
```

Definition at line 205 of file MRCUtils.cpp.

```
00205                                                   {
00206                 if( checkQueryName(options)==mrc::QUERY_NAME_FOUND )
00207                         minerule::OptimizerCatalogue::deleteMinerule(
      options.getSearchParam(Options::QryName) );
00208         }
```

5.2.2.5 doLoadOptions()

```
void mrc::doLoadOptions (
            const char * fname,
            minerule::MineruleOptions & opt )
```

Definition at line 38 of file MRCUtils.cpp.

```
00038                                                                                   {
00039
00040                 if( minerule::FileUtils::fileExists(fname) ) {
00041                         opt.readFromFile(fname);
00042                 } else {
00043                         std::string errstr;
00044                         if(errno==0) {
00045                                 errstr = "Not a regular file!";
00046                         } else {
00047                                 errstr = strerror(errno);
00048                         }
00049
00050                         throw Exception(mrc::ERROR_CANNOT_OPEN_OPTION_FILE,
00051                                 std::string("Could not open file:") + fname + ". The reason is:" +
      errstr);
00052                 }
00053         }
```

5.2.2.6 installCatalogue()

```
Results mrc::installCatalogue ( )
```

Definition at line 220 of file MRCUtils.cpp.

```
00220                                           {
00221                 minerule::MRLog() << "Checking current installation status." << std::endl;
00222                 if(minerule::OptimizerCatalogue::checkInstallation()) {
00223                         minerule::MRLog() << minerule::StringUtils::toBold("Minerule catalogue seems
      already installed.") << std::endl;
00224                         minerule::MRLog() << minerule::StringUtils::toRed("Cowardly refusing to install
      over a working catalogue...") << std::endl;
00225                         return NOTHING_TO_DO;
00226                 } else {
00227                         minerule::MRLog() << "Checking if everything is ok." << std::endl;
00228                         minerule::MRLog() << minerule::StringUtils::toBold("The catalogue is not
      installed (properly).") << std::endl;
00229                         minerule::MRLog() << "Proceeding with the installation..." << std::endl;
00230                         minerule::OptimizerCatalogue::install();
00231                         minerule::MRLog() << "Done!" << std::endl;
00232                         minerule::MRLog() << "Checking the installation..." << std::endl;

00233                         if(!minerule::OptimizerCatalogue::checkInstallation()) {
00234                                 minerule::MRLog() << "Something went wrong. This is likely a bug.
      Please report it!" << std::endl;
00235                                 return CATALOGUE_NOT_INSTALLED;
00236                         } else {
00237                                 minerule::MRLog() << minerule::StringUtils::toBold("The catalogue is
      now properly installed.") << std::endl;
00238                                 return SUCCESS;
00239                         }
00240                 }
00241         }
```

5.2.2.7 parseOptions()

```
void mrc::parseOptions (
        int argc,
        char ** argv,
        minerule::MineruleOptions & mopt,
        Options & popt )
```

Definition at line 55 of file MRCUtils.cpp.

```
00055                                                                                            {
00056         const char* optstr = "cCIUhf:lF:n:d:va:";
00057 
00058         int opt;
00059         bool didLoadOptions=false;
00060         while((opt=getopt(argc, argv, optstr))!=-1) {
00061                 switch( opt ) {
00062                 case 'c':
00063                         minerule::StringUtils::setColorsEnabled(false);
00064                         break;
00065                 case 'C':
00066                         popt.setCheckCatalogue();
00067                         break;
00068                 case 'U':
00069                         popt.setUninstallCatalogue();
00070                         break;
00071                 case 'I':
00072                         popt.setInstallCatalogue();
00073                         break;
00074                 case 'f':
00075                         doLoadOptions(optarg, mopt);
00076                         didLoadOptions=true;
00077                         break;
00078                 case 'h':
00079                         printHelp(argc, argv);
00080                         break;
00081                 case 'F':
00082                         popt.setListFormat(optarg);
00083                         break;
00084                 case 'l':
00085                         popt.setShowList();
00086                         break;
00087                 case 'n':
00088                         popt.setSearchQry();
00089                         popt.setSearchParam(Options::QryName, optarg);
00090                         break;
00091                 case 'd':
00092                         popt.setDeleteQry();
00093                         popt.setSearchParam(Options::QryName, optarg);
00094                         break;
00095                 case 'a': {
00096                                 std::vector<std::string> queries =
       minerule::StringUtils::split(optarg, ",");
00097                                 if( queries.size() != 2 ) {
00098                                         throw Exception( mrc::ERROR_OPTION_PARSING, "Wrong
       format for -a options, you should pass the original query name and the derived query name separated
       by a ',' (no spaces)");
00099                                 }
00100                                 popt.setAddDerivedQuery(queries[0], queries[1]);
00101                         }
00102                         break;
00103                 case 'v':
00104                         printVersion();
00105                         exit(0);
00106                 case '?':
00107                 default:
00108                         throw Exception( mrc::ERROR_OPTION_PARSING, "Error while parsing
       options, use -h to show" " the help message" );
00109                 }
00110         }
00111         if( !didLoadOptions  ) {
00112                 if(
       !minerule::FileUtils::fileExists(minerule::MineruleOptions::DEFAULT_FILE_NAME) )
00113                         throw Exception( mrc::ERROR_OPTION_PARSING,"You should specify at
       least one -f option or provide ./"+minerule::MineruleOptions::DEFAULT_FILE_NAME);
00114                 else
00115                         doLoadOptions(minerule::MineruleOptions::DEFAULT_FILE_NAME.c_str(),
       mopt);
00116         }
00117 }
```

5.2.2.8 performCommands()

```
int mrc::performCommands (
            Options & options )
```

Definition at line 273 of file MRCUtils.cpp.

```
00273                                                {
00274                 if( options.getCommand()==Options::ShowList ) {
00275                         printMRQueryList(options);
00276                         return mrc::SUCCESS;
00277                 }
00278
00279                 if( options.getCommand()==Options::SearchQry) {
00280                         return checkQueryName(options);
00281                 }
00282
00283                 if( options.getCommand()==Options::DeleteQry ) {
00284                         deleteQuery(options);
00285                         return mrc::SUCCESS;
00286                 }
00287
00288                 if( options.getCommand()==Options::CheckCatalogue ) {
00289                         return checkCatalogue();
00290                 }
00291
00292                 if( options.getCommand()==Options::InstallCatalogue ) {
00293                         return installCatalogue();
00294                 }
00295
00296                 if( options.getCommand()==Options::UninstallCatalogue ) {
00297                         return uninstallCatalogue();
00298                 }
00299
00300                 if( options.getCommand()==Options::AddDerivedQuery ) {
00301                         return addDerivedQuery(options);
00302                 }
00303
00304
00305                 return mrc::NOTHING_TO_DO;
00306         }
```

5.2.2.9 printHelp()

```
void mrc::printHelp (
            int argc,
            char ** argv )
```

Definition at line 127 of file MRCUtils.cpp.

```
00127                                                {
00128                 using namespace minerule;
00129                 std::cout « StringUtils::toBold("Usage:") « std::endl
00130                         « "   " « StringUtils::toBold(argv[0]) « " [-v] [-h] [-f  <optionfile>] [-C]
      [-I] [-U] [-n <queryname>] [-l] [-d] [-a <ori>,<der>][-c] [-F <formatSpecs>]" « std::endl
00131                         « std::endl « std::endl
00132                         « "Handles the minerule catalogue." « std::endl
00133                         « std::endl
00134                         « StringUtils::toBold("Program Information") « std::endl
00135                         « StringUtils::toBold("-v") « " - Output the version message and exits." «
      std::endl
00136                         « StringUtils::toBold("-h") « " - Output this message and returns
      NOTHING_TO_DO." « std::endl
00137                         « std::endl
00138                         « StringUtils::toBold("Option Handling") « std::endl
00139                         « StringUtils::toBold("-f") « " - file name of the option file (if omitted the
      program will)" « std::endl
00140                         « "     look for a file named 'option.txt' in the current directory" «
      std::endl
00141                         « std::endl
00142                         « StringUtils::toBold("Installation") « std::endl
00143                         « StringUtils::toBold("-C") « " - Checks if the optimizer catalogue is
      installed correctly and exits." «std::endl
00144                         « StringUtils::toBold("-I") « " - Installs the optimizer catalogue and exits."
      «std::endl
```

```
00145                         « StringUtils::toBold("-U") « " - Uninstall the catalogue (be careful, this
      cannot be undone!)." « std::endl
00146                         « "     dbms must be either 'mysql' or 'postgres'." « std::endl
00147                         « std::endl
00148                         « StringUtils::toBold("Dealing with Catalogue Entries") « std::endl
00149                         « StringUtils::toBold("-n") « " - Look in the catalogue for the specified
      query." « std::endl
00150                         « "     it returns QUERY_NAME_FOUND, or QUERY_NAME_NOT_FOUND" « std::endl
00151                         « "     accordingly to whether the query could be found."«std::endl
00152                         « StringUtils::toBold("-l") « " - Print the list of already executed queries,
      returns" « std::endl
00153                         « "     SUCCESS upon completion." « std::endl
00154                         « StringUtils::toBold("-d") « " - Delete the given minerule from the system
      (notice that" « std::endl
00155                         « "     the safety options in the option file must be setted" « std::endl
00156                         « "     in such a way to allow the deletion. " « std::endl
00157                         « StringUtils::toBold("-a") « " - Adds a derived result set. You must provide
      the name <ori> of the original" « std::endl
00158                         « "     minerule and the name <der> of the derived result. The database must
      then" « std::endl
00159                         « "     contain a table named <der> and two additional tables <der>_body_elems
      and " « std::endl
00160                         « "     <der>_head_elems" « std::endl
00161                         « std::endl
00162                         « StringUtils::toBold("Output Handling") « std::endl
00163                         « StringUtils::toBold("-c") « " - Disables color output." « std::endl
00164                         « StringUtils::toBold("-F") « " - Format specifiers for printing the list of
      queries (-l)" «std::endl
00165                         « "     Valid specifiers: s - Print the size of the result set" « std::endl
00166                         « "                       t - Print the text of the original mr" «std::endl
00167                         « "                       r - Print the result set table names" « std::endl
00168                         « "     the default is '', i.e.: print onlty the qry name." « std::endl
00169                         « std::endl
00170                         « StringUtils::toBold("Return Values:") « std::endl
00171                         « "      SUCCESS = " « mrc::SUCCESS « std::endl
00172                         « "      NOTHING_TO_DO = " « mrc::NOTHING_TO_DO « std::endl
00173                         « "      QUERY_NAME_FOUND = " « mrc::QUERY_NAME_FOUND « std::endl
00174                         « "      CATALOGUE_NOT_INSTALLED = " « mrc::CATALOGUE_NOT_INSTALLED «
      std::endl
00175                         « "      QUERY_NAME_NOT_FOUND = " « mrc::QUERY_NAME_NOT_FOUND « std::endl
00176                         « "      ERROR_OPTION_PARSING = " « mrc::ERROR_OPTION_PARSING « std::endl
00177
                                                              « "      ERROR_CANNOT_OPEN_OPTION_FILE = " «
      mrc::ERROR_CANNOT_OPEN_OPTION_FILE «std::endl;
00178         }
```

5.2.2.10 printMRQueryList()

```
void mrc::printMRQueryList (
              const Options & options )
```

Definition at line 197 of file MRCUtils.cpp.

```
00197                                                 {
00198                 Printer printer(std::cout, options);
00199                 std::vector<minerule::CatalogueInfo> mrqlist;
00200                 minerule::OptimizerCatalogue::getMRQueryInfos(mrqlist, options.getListFormat().size );
00201
00202                 printer.print(mrqlist);
00203         }
```

5.2.2.11 printVersion()

```
void mrc::printVersion ( )
```

Definition at line 119 of file MRCUtils.cpp.

```
00119                           {
00120                 std::cout « "MRCatalogue v:" « MR_VERSION « std::endl;
00121         }
```

5.2.2.12 uninstallCatalogue()

```
Results mrc::uninstallCatalogue ( )
```

Definition at line 243 of file MRCUtils.cpp.

```
00243                                   {
00244                 minerule::MRLog() « "You choose to " « minerule::StringUtils::toBold(" uninstall ") «
      "the minerule catalogue" « std::endl;
00245                 minerule::MRLog() « "This implies " « minerule::StringUtils::toBold(" destroying ") «
      "all its tables" « std::endl;
00246
00247                 std::cout « "Are you " « minerule::StringUtils::toBold("sure ") « "you want to
      proceed? (y/N): ";
00248                 std::string confirm;
00249                 std::cin » confirm;
00250                 transform(confirm.begin(), confirm.end(), confirm.begin(), ::toupper);
00251                 if( confirm != "Y" && confirm != "YES") {
00252                         minerule::MRLog() « "Ok. Bailing out." « std::endl;
00253                         return NOTHING_TO_DO;
00254                 }
00255
00256                 minerule::MRLog() « "Uninstalling the catalogue..." « std::endl;
00257                 minerule::OptimizerCatalogue::uninstall();
00258                 minerule::MRLog() « "Done" « std::endl;
00259
00260                 return SUCCESS;
00261         }
```

5.3 mrdb Namespace Reference

Data Structures

- class Connection
- class DatabaseMetaData
- class PreparedStatement
- class ResultSet
- class ResultSetMetaData
- class SQLException
- class Statement
- struct Types

5.4 mrmatch Namespace Reference

Data Structures

- class GidRulesMatcher
- class Matcher
- class Options
- class RuleGidsDBMatcher
- class RuleGidsMatcher

Typedefs

- typedef unsigned int MatcherSpecs

Enumerations

- enum ExitCodes { SUCCESS =0 , MRMATCH_OPTION_PARSING_ERROR }
- enum MatchKind { None = 0 , RuleGids = 1 , GidRules = 2 , MatchKindMask = 3 }
- enum MatchOutput { OutOnConsole = 4 , OutOnDB = 8 , MatchOutputMask = 12 }

Functions

- SourceTableRequirements sourceTableRequirements (const ParsedMinerule &minerule)
- void execute (const Options &options)

5.4.1 Typedef Documentation

5.4.1.1 MatcherSpecs

```
typedef unsigned int mrmatch::MatcherSpecs
```

Definition at line 42 of file mrmatch.hpp.

5.4.2 Enumeration Type Documentation

5.4.2.1 ExitCodes

```
enum mrmatch::ExitCodes
```

Enumerator

SUCCESS	
MRMATCH_OPTION_PARSING_ERROR	

Definition at line 23 of file mrmatch.hpp.

```
00023 { SUCCESS=0, MRMATCH_OPTION_PARSING_ERROR  } ExitCodes;
```

5.4.2.2 MatchKind

```
enum mrmatch::MatchKind
```

Enumerator

None	

Enumerator

RuleGids	
GidRules	
MatchKindMask	

Definition at line 40 of file mrmatch.hpp.

```
00040 { None = 0, RuleGids = 1, GidRules = 2, MatchKindMask = 3 } MatchKind;
```

5.4.2.3 MatchOutput

```
enum mrmatch::MatchOutput
```

Enumerator

OutOnConsole	
OutOnDB	
MatchOutputMask	

Definition at line 41 of file mrmatch.hpp.

```
00041 { OutOnConsole = 4, OutOnDB = 8, MatchOutputMask = 12 } MatchOutput;
```

5.4.3 Function Documentation

5.4.3.1 execute()

```
void mrmatch::execute (
            const Options & options )
```

Definition at line 39 of file mrmatch.cpp.

```
00039                                           {
00040                 MRLogPush("Building source table...");
00041                 // rebuild source table
00042                 CatalogueInfo info;
00043                 OptimizerCatalogue::getMRQueryInfo(options.queryName(), info, false);
00044                 ParsedMinerule minerule(info.qryText);
00045                 if( options.tableName() != "" ) {
00046                         minerule.tab_source = options.tableName();
00047                 }
00048
00049                 SourceTableRequirements requirements = sourceTableRequirements(minerule);
00050                 SourceTable st( minerule, requirements);
00051                 MRLogPop();
00052
00053                 MRLogPusher("Reading rules...");
00054                 // load past minerule result
00055                 QueryResult::Iterator it;
00056                 OptimizerCatalogue::getMRQueryResultIterator(options.queryName(), it, minerule.sup,
      minerule.conf);
00057
00058                 std::auto_ptr<Matcher> matcher(Matcher::newMatcher(options, minerule));
00059
00060                 while(it.next()) {
00061                         Rule& rule = matcher->addRule();
```

```
00062                         it.getRule(rule);
00063                 }
00064                 MRLogPop();
00065
00066                 MRLogPush("Matching...");
00067                 matcher->match(st);
00068                 MRLogPop();
00069
00070                 MRLogPush("Printing results:");
00071                 matcher->printMatches( );
00072
00073                 MRLogPop();
00074         }
```

5.4.3.2 sourceTableRequirements()

```
SourceTableRequirements mrmatch::sourceTableRequirements (
             const ParsedMinerule & minerule )
```

Definition at line 32 of file mrmatch.cpp.

```
00032                                                                                                {
00033                 if(minerule.hasCrossConditions() || !OptimizerCatalogue::hasIDConstraints(minerule))
00034                         return SourceTableRequirements( SourceTableRequirements::CrossProduct |
      SourceTableRequirements::SortedGids );
00035                 else
00036                         return SourceTableRequirements(SourceTableRequirements::SortedGids);

00037         }
```

5.5 mrprint Namespace Reference

Data Structures

- class Options

Enumerations

- enum MRPRINT_ERRORS { MRPRINT_OPTION_PARSING_ERROR =1 }

Functions

- void printError (const std::string &error)
- void printRules (std::string queryname, RuleFormatter &formatter, double conf)
- std::string getQueryName (const Options &options)
- RuleFormatter * newFormatter (const Options &options)

5.5.1 Enumeration Type Documentation

5.5.1.1 MRPRINT_ERRORS

```
enum mrprint::MRPRINT_ERRORS
```

Enumerator

MRPRINT_OPTION_PARSING_ERROR	

Definition at line 23 of file Options.hpp.

```
00023 { MRPRINT_OPTION_PARSING_ERROR=1 };
```

5.5.2 Function Documentation

5.5.2.1 getQueryName()

```
std::string mrprint::getQueryName (
            const Options & options )
```

Definition at line 54 of file PrintRules.cpp.

```
00054                                                   {
00055                 if( !options.queryName().empty() )
00056                         return options.queryName();
00057
00058                 return minerule::OptimizerCatalogue::getMRQueryName(options.queryNumber());
00059         }
```

5.5.2.2 newFormatter()

```
RuleFormatter * mrprint::newFormatter (
            const Options & options )
```

Definition at line 62 of file PrintRules.cpp.

```
00062                                                       {
00063                 RuleFormatter* rf = NULL;
00064                 std::string sortOrder = options.sortOrder();
00065
00066                 if(sortOrder=="no") {
00067                                 rf=new SimpleRuleFormatter();
00068                 } else if(sortOrder=="scbh") {
00069                                 rf=new SortedRuleFormatter<QueryResult::SortSuppConfBodyHead>();
00070                 } else if(sortOrder=="bhsc") {
00071                                 rf=new SortedRuleFormatter<QueryResult::SortBodyHeadSuppConf>();
00072                 } else if(sortOrder=="hbsc") {
00073                                 rf=new SortedRuleFormatter<QueryResult::SortHeadBodySuppConf>();
00074                 } else if(sortOrder=="csbh") {
00075                                 rf=new SortedRuleFormatter<QueryResult::SortConfSuppBodyHead>();
00076                 } else if(sortOrder=="cbhs") {
00077                                 rf=new SortedRuleFormatter<QueryResult::SortConfBodyHeadSupp>();
00078                 } else if(sortOrder=="cbsh") {
00079                                 rf=new SortedRuleFormatter<QueryResult::SortConfBodySuppHead>();
00080                 } else {
00081                                 printError("Unsupported sort order:" + sortOrder);
00082                                 exit(MRPRINT_OPTION_PARSING_ERROR);
00083                 }
00084
00085                 if(rf==NULL) {
00086                         rf = new SimpleRuleFormatter();
00087                 }
00088
00089                 if( options.noLogArtifacts() )
00090                         rf->setSuppressLog(true);
00091
00092                 return rf;
00093         }
```

5.5.2.3 printError()

```
void mrprint::printError (
            const std::string & error )
```

Definition at line 26 of file Options.cpp.

```
00026                                                {
00027                 std::cout « minerule::StringUtils::toRed("Error:") « error « std::endl;
00028         }
```

5.5.2.4 printRules()

```
void mrprint::printRules (
            std::string queryname,
            RuleFormatter & formatter,
            double conf )
```

Definition at line 39 of file PrintRules.cpp.

```
00039                                                                                {
00040                 QueryResult::Iterator qit;
00041                 OptimizerCatalogue::getMRQueryResultIterator( queryname, qit, -1, conf );
00042
00043                 while( qit.next() ) {
00044                         Rule r;
00045                         qit.getRule(r);
00046
00047                         formatter.printRule(r);
00048                 }
00049
00050                 formatter.postExec();
00051         }
```

Chapter 6

Data Structure Documentation

6.1 __taglist_AND_node Struct Reference

`#include <ParsedMinerule.hpp>`

Data Fields

- simple_pred * sp
- list_AND_node * next

6.1.1 Detailed Description

Definition at line 31 of file ParsedMinerule.hpp.

6.1.2 Field Documentation

6.1.2.1 next

```
list_AND_node* __taglist_AND_node::next
```

Definition at line 34 of file ParsedMinerule.hpp.

6.1.2.2 sp

```
simple_pred* __taglist_AND_node::sp
```

Definition at line 33 of file ParsedMinerule.hpp.

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.2 __taglist_OR_node Struct Reference

`#include <ParsedMinerule.hpp>`

Data Fields

- list_AND_node * l_and
- list_OR_node * next

6.2.1 Detailed Description

Definition at line 37 of file ParsedMinerule.hpp.

6.2.2 Field Documentation

6.2.2.1 l_and

`list_AND_node* __taglist_OR_node::l_and`

Definition at line 39 of file ParsedMinerule.hpp.

6.2.2.2 next

`list_OR_node* __taglist_OR_node::next`

Definition at line 40 of file ParsedMinerule.hpp.

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.3 __tagminerule Struct Reference

`#include <ParsedMinerule.hpp>`

Data Fields

- struct att_list * ga
- struct att_list * ca
- struct att_list * ra
- list_OR_node * mc
- list_OR_node * gc
- list_OR_node * cc

6.3.1 Detailed Description

Definition at line 43 of file ParsedMinerule.hpp.

6.3.2 Field Documentation

6.3.2.1 ca

```
struct att_list* __tagminerule::ca
```

Definition at line 46 of file ParsedMinerule.hpp.

6.3.2.2 cc

```
list_OR_node* __tagminerule::cc
```

Definition at line 50 of file ParsedMinerule.hpp.

6.3.2.3 ga

```
struct att_list* __tagminerule::ga
```

Definition at line 45 of file ParsedMinerule.hpp.

6.3.2.4 gc

```
list_OR_node* __tagminerule::gc
```

Definition at line 49 of file ParsedMinerule.hpp.

6.3.2.5 mc

```
list_OR_node* __tagminerule::mc
```

Definition at line 48 of file ParsedMinerule.hpp.

6.3.2.6 ra

```
struct att_list* __tagminerule::ra
```

Definition at line 47 of file ParsedMinerule.hpp.

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.4 __tagsimple_pred Struct Reference

```
#include <ParsedMinerule.hpp>
```

Data Fields

- char * val1
- char * op
- char * val2

6.4.1 Detailed Description

Definition at line 24 of file ParsedMinerule.hpp.

6.4.2 Field Documentation

6.4.2.1 op

```
char* __tagsimple_pred::op
```

Definition at line 27 of file ParsedMinerule.hpp.

6.4.2.2 val1

```
char* __tagsimple_pred::val1
```

Definition at line 26 of file ParsedMinerule.hpp.

6.4.2.3 val2

```
char* __tagsimple_pred::val2
```

Definition at line 28 of file ParsedMinerule.hpp.

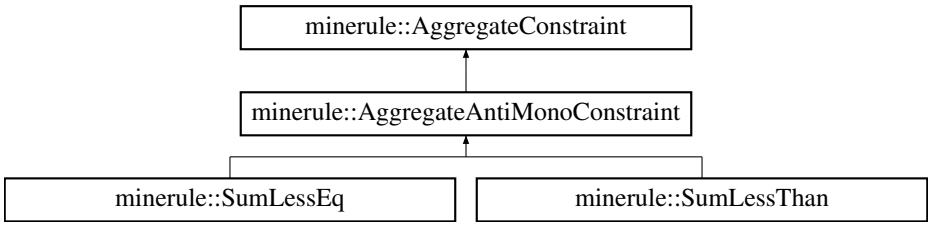
The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.5 minerule::AggregateAntiMonoConstraint Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::AggregateAntiMonoConstraint:



Public Member Functions

- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)=0
- virtual bool check (Transaction &items)=0

Data Fields

- int attributeIndex

6.5.1 Detailed Description

Definition at line 52 of file Constraints.hpp.

6.5.2 Member Function Documentation

6.5.2.1 check() [1/2]

```
virtual bool minerule::AggregateAntiMonoConstraint::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [pure virtual]
```

Implements minerule::AggregateConstraint.

Implemented in minerule::SumLessThan, and minerule::SumLessEq.

6.5.2.2 check() [2/2]

```
virtual bool minerule::AggregateAntiMonoConstraint::check (
            Transaction & items )  [pure virtual]
```

Implements minerule::AggregateConstraint.

Implemented in minerule::SumLessThan, and minerule::SumLessEq.

6.5.3 Field Documentation

6.5.3.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

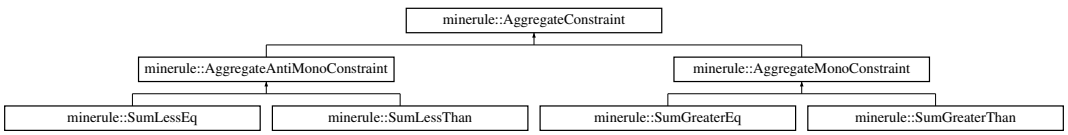
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp

6.6 minerule::AggregateConstraint Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::AggregateConstraint:



Public Member Functions

- AggregateConstraint (int ai)
- AggregateConstraint ()
- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)=0
- virtual bool check (Transaction &items)=0

Data Fields

- int attributeIndex

Friends

- std::istream & operator>> (std::istream &in, AggregateConstraint &i1)
- std::ostream & operator<< (std::ostream &out, const AggregateConstraint &i1)

6.6.1 Detailed Description

Definition at line 26 of file Constraints.hpp.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AggregateConstraint() [1/2]

```
minerule::AggregateConstraint::AggregateConstraint (
            int ai )  [inline]
```

Definition at line 29 of file Constraints.hpp.

```
00029 : attributeIndex(ai) {}
```

6.6.2.2 AggregateConstraint() [2/2]

```
minerule::AggregateConstraint::AggregateConstraint ( )  [inline]
```

Definition at line 30 of file Constraints.hpp.

```
00030 : attributeIndex(0) {}
```

6.6.3 Member Function Documentation

6.6.3.1 check() [1/2]

```
virtual bool minerule::AggregateConstraint::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [pure virtual]
```

Implemented in minerule::SumLessThan, minerule::SumLessEq, minerule::SumGreaterThan, minerule::SumGreaterEq, minerule::AggregateMonoConstraint, and minerule::AggregateAntiMonoConstraint.

6.6.3.2 check() [2/2]

```
virtual bool minerule::AggregateConstraint::check (
            Transaction & items )  [pure virtual]
```

Implemented in minerule::SumLessThan, minerule::SumLessEq, minerule::SumGreaterThan, minerule::SumGreaterEq, minerule::AggregateMonoConstraint, and minerule::AggregateAntiMonoConstraint.

6.6.4 Friends And Related Function Documentation

6.6.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const AggregateConstraint & i1 )  [friend]
```

Definition at line 37 of file Constraints.hpp.

```
00037                                                                    {
00038                 out << i1.attributeIndex;
00039                 return out;
00040         }
```

6.6.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            AggregateConstraint & i1 )  [friend]
```

Definition at line 33 of file Constraints.hpp.

```
00033                                                                   {
00034                 in >> i1.attributeIndex;
00035                 return in;
00036         }
```

6.6.5 Field Documentation

6.6.5.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex
```

Definition at line 28 of file Constraints.hpp.

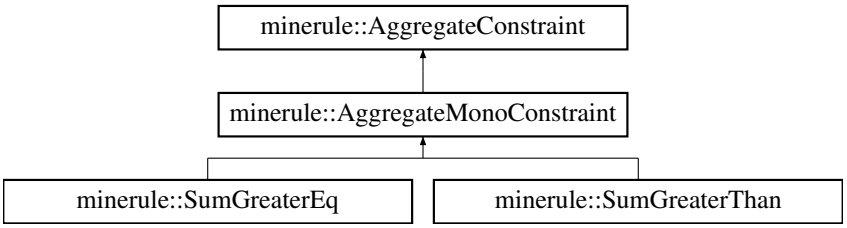
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp

6.7 minerule::AggregateMonoConstraint Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::AggregateMonoConstraint:



Public Member Functions

- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)=0
- virtual bool check (Transaction &items)=0

Data Fields

- int attributeIndex

6.7.1 Detailed Description

Definition at line 44 of file Constraints.hpp.

6.7.2 Member Function Documentation

6.7.2.1 check() [1/2]

```
virtual bool minerule::AggregateMonoConstraint::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [pure virtual]
```

Implements minerule::AggregateConstraint.

Implemented in minerule::SumGreaterThan, and minerule::SumGreaterEq.

6.7.2.2 check() [2/2]

```
virtual bool minerule::AggregateMonoConstraint::check (
            Transaction & items )  [pure virtual]
```

Implements minerule::AggregateConstraint.

Implemented in minerule::SumGreaterThan, and minerule::SumGreaterEq.

6.7.3 Field Documentation

6.7.3.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp

6.8 minerule::Algorithms Class Reference

```
#include <Algorithms.hpp>
```

Static Public Member Functions

- static MiningAlgorithmBase * newAlgorithm (const OptimizedMinerule &mr)
- static void executeMinerule (OptimizedMinerule &mr)
- static void executeExtractionAlgorithm (OptimizedMinerule &mr)

  It executes a rule extraction algorithm.
- static bool executeIncrementalAlgorithm (OptimizedMinerule &mr)
- static MiningAlgorithmBase * getBestRulesMiningAlgorithm (const OptimizedMinerule &mr)
- static MiningAlgorithmBase * getBestItemsetsMiningAlgorithm (const OptimizedMinerule &mr)
- static MiningAlgorithmBase * getBestSequencesMiningAlgorithm (const OptimizedMinerule &mr)

6.8.1 Detailed Description

Definition at line 29 of file Algorithms.hpp.

6.8.2 Member Function Documentation

6.8.2.1 executeExtractionAlgorithm()

```
void minerule::Algorithms::executeExtractionAlgorithm (
            OptimizedMinerule & mr )  [static]
```

It executes a rule extraction algorithm.

Definition at line 159 of file Algorithms.cpp.

```
00159                                                                   {
00160                 MiningAlgorithmBase* algo =  Algorithms::newAlgorithm(mr);
00161                 algo->execute();
00162                 delete algo;
00163         }
```

6.8.2.2 executeIncrementalAlgorithm()

```
bool minerule::Algorithms::executeIncrementalAlgorithm (
            OptimizedMinerule & mr )  [static]
```

It execute an incremental algorithm. It returns false if an algorithm with the required properties has not yet been implemented.

Definition at line 141 of file Algorithms.cpp.

```
00141                                                                   {
00142
00143                 IncrementalAlgorithm* incrAlgo = IncrementalAlgorithm::newIncrementalAlgorithm(mr);
00144
00145                 if( incrAlgo!=NULL ) {
00146                         incrAlgo->execute();
00147
00148                         delete incrAlgo;
00149                         return true;
00150                 } else {
00151                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00152                                         « " to the system yet." « std::endl;
00153                         return false;
00154                 }
00155         }
```

6.8.2.3 executeMinerule()

```
void minerule::Algorithms::executeMinerule (
            OptimizedMinerule & mr )  [static]
```

It executes the provided minerule and stores the results in the database. It exploits the optimization info to select the best option.

Definition at line 196 of file Algorithms.cpp.

```
00196                                                                   {
00197                 checkAndHandleHomonymMinerules(mr);
00198
00199                 showDebugInfo("Unoptimized Minerule info", mr);
00200                 mr.optimize(); // internally it will check if the optimization option is set.
00201                 showDebugInfo("Optimized Minerule info", mr);
00202
00203
00204                 std::string unsupportedRelation = "";
00205                 OptimizerCatalogue::MineruleResultInfo result(mr.getParsedMinerule());
00206
00207                 switch(mr.getOptimizationInfo().relationship) {
00208
00209                         // EQUIVALENCE
00210                         case OptimizedMinerule::Equivalence:
00211                                 {
00212                                         MRLog("Using equivalence relationship.");
00213
00214                                         CatalogueInfo catInfo;
00215                                         OptimizerCatalogue::getMRQueryInfo(
    mr.getOptimizationInfo().minerule.tab_result, catInfo );
00216                                         result.resultset = catInfo.resName;
00217
00218                                         MRDebug() << "Inclusion found with respect minerule:" <<
    mr.getOptimizationInfo().minerule.getText() << std::endl;
00219                                         MRDebug() << "Current Minerule:" << result.getText() <<
    std::endl;
00220                                         OptimizerCatalogue::addMineruleResult(result);
00221                                 }
00222                                 break;
00223
00224                         // DOMINANCE
00225                         case OptimizedMinerule::Dominance:
00226                                 unsupportedRelation = "Dominance";
00227                                 MRLog("Using dominance relationship.");
00228
00229                         // COMBINATION
00230                         case OptimizedMinerule::Combination:
00231                                 if(unsupportedRelation=="") {
00232                                         MRLog("Using combination relationship.");
00233
00234                                         unsupportedRelation="Combination";
00235                         }
00236
00237                                 if(executeIncrementalAlgorithm(mr)) {
00238                                         OptimizerCatalogue::addMineruleResult(result);
00239                                         break;
00240                                 } else {
00241                                         MRLog() << "The support for the found dominance relationship is
    not yet implemented"
00242                                                 << " switching back to the non-incremental mining
    algorithm;" << std::endl;
00243                                 }
00244
00245                         // INCLUSION
00246                         case OptimizedMinerule::Inclusion:
00247                                 if( unsupportedRelation=="" ) {
00248                                         unsupportedRelation = "Inclusion";
00249                                         MRLog("Using inclusion relationship.");
00250                                 }
00251
00252                         // NO RELATION FOUND
00253                         case OptimizedMinerule::None:
00254                                 if( unsupportedRelation!="" ) {
00255                                         MRWarn() << "The optimizer found that there exists a minerule
    in the catalogue" << std::endl
00256                                                 << "which is in `" << unsupportedRelation <<"'
    relationship with the" << std::endl
00257                                                 << "current one. Unfortunately such kind of
    relationship is still" << std::endl
00258                                                 << "not supported and hence I will switch to the
    default algorithm" << std::endl;
00259                                 }
```

```
00260
00261                                   executeExtractionAlgorithm(mr);
00262
00263                                   if( mr.getParsedMinerule().miningTask==MTMineRules ||
      mr.getParsedMinerule().miningTask==MTMineItemsets )
00264                                                   OptimizerCatalogue::addMineruleResult(result);
00265                                   break;
00266
00267                           // ERROR
00268                           default:
00269                                   throw MineruleException(MR_ERROR_INTERNAL, "Unexpected Relationship!
      This is a BUG! Please report it!");
00270                   }
00271           }
```

6.8.2.4 getBestItemsetsMiningAlgorithm()

```
MiningAlgorithmBase * minerule::Algorithms::getBestItemsetsMiningAlgorithm (
            const OptimizedMinerule & mr )  [static]
```

Returns

an instance of the best itemset mining algorithm available for the given mine rule.

Definition at line 74 of file Algorithms.cpp.

```
00074                                                                                   {
00075                   MRDebugPusher pusher("Choosing the best algorithm for the given MR");
00076
00077                   AlgorithmTypes userChoiceOfAT =
00078
      MineruleOptions::getSharedOptions().getMiningAlgorithms().getRulesMiningAlgorithms().getPreferredAlgorithm();
00079
00080                   MiningAlgorithmBase* userChoice= MiningAlgorithm::algorithmForType(userChoiceOfAT,
      mr);
00081                   if( userChoice->canHandleMinerule() ) {
00082                           MRDebug() « "Selected the algorithm given by the user preference" « std::endl;
00083                           return userChoice;
00084                   }
00085                   else {
00086                           MRDebug() « "User preference cannot be fulfilled" « std::endl;
00087                           delete userChoice;
00088                           userChoice=NULL;
00089                   }
00090
00091    // we failed to satisfy user preference, it is up to us
00092    // to find the best algorithm.
00093
00094                   if( ConstrItemSetsExtraction(mr).canHandleMinerule() ) {
00095                           MRDebug() « "Selected ConstrItemSetsExtraction" « std::endl;
00096                           return new ConstrItemSetsExtraction(mr);
00097                   }
00098
00099                   MRDebug() « "Panic! No known algorithm can handle it." « std::endl;
00100                   throw MineruleException( MR_ERROR_INTERNAL, "No known algorithm can handle the given
      minerule!" );
00101           }
```

6.8.2.5 getBestRulesMiningAlgorithm()

```
MiningAlgorithmBase * minerule::Algorithms::getBestRulesMiningAlgorithm (
            const OptimizedMinerule & mr )  [static]
```

Returns

an instance of the best rule mining algorithm available for the given mine rule.

Definition at line 33 of file Algorithms.cpp.

```
00033                                                                                                   {
00034                 MRDebugPusher pusher("Choosing the best algorithm for the given MR");
00035
00036                 AlgorithmTypes userChoiceOfAT =
      MineruleOptions::getSharedOptions().getMiningAlgorithms().getRulesMiningAlgorithms().getPreferredAlgorithm();
00037
00038                 MiningAlgorithmBase* userChoice= MiningAlgorithm::algorithmForType(userChoiceOfAT,
      mr);
00039                 if( userChoice->canHandleMinerule() ) {
00040                         MRDebug() << "Selected the algorithm given by the user preference" << std::endl;
00041                         return userChoice;
00042                 }
00043                 else {
00044                         MRDebug() << "User preference cannot be fulfilled" << std::endl;
00045                         delete userChoice;
00046                         userChoice=NULL;
00047                 }
00048
00049  // we failed to satisfy user preference, it is up to us
00050  // to find the best algorithm.
00051
00052                 if( BFSWithGidsNoCross(mr).canHandleMinerule() ) {
00053                         MRDebug() << "Selected BFSWithGidsNoCross" << std::endl;
00054                         return new BFSWithGidsNoCross(mr);
00055                 }
00056
00057                 MRDebug() << "BFSWithGidsNoCross cannot handle it." << std::endl;
00058
00059                 if( BFSWithGidsAndCross(mr).canHandleMinerule() ) {
00060                         MRDebug() << "Selected BFSWithGidsAndCross" << std::endl;
00061                         return new BFSWithGidsAndCross(mr);
00062                 }
00063
00064                 MRDebug() << "BFSWithGidsAndCross cannot handle it." << std::endl;
00065
00066
00067                 MRDebug() << "Panic! No known algorithm can handle it." << std::endl;
00068                 throw MineruleException( MR_ERROR_INTERNAL, "No known algorithm can handle the given
      minerule!" );
00069         }
```

6.8.2.6 getBestSequencesMiningAlgorithm()

```
MiningAlgorithmBase * minerule::Algorithms::getBestSequencesMiningAlgorithm (
            const OptimizedMinerule & mr )  [static]
```

Returns

an instance of the best sequence mining algorithm available for the given mine rule.

Definition at line 106 of file Algorithms.cpp.

```
00106                                                                                                   {
00107                 AlgorithmsOptions opts;
00108
00109                 opts.setSupport( mr.getParsedMinerule().sup );
00110                 opts.setBodyCardinalities( mr.getParsedMinerule().bodyCardinalities );
00111
00112                 //CCSMiner* miner = new CCSMiner(mr,opts);
00113                 STSMiner* miner= new STSMiner(mr,opts);
00114                 if(miner->canHandleMinerule())
00115                         return miner;
00116                 else {
00117                         delete miner;
00118                         throw MineruleException( MR_ERROR_INTERNAL, "Cannot handle specified mine
      request." );
00119                 }
00120         }
```

6.8.2.7 newAlgorithm()

```
MiningAlgorithmBase * minerule::Algorithms::newAlgorithm (
            const OptimizedMinerule & mr )  [static]
```

Returns

a new MiningAlgorithm choosen among the available ones using some criteria. One constraint it respects is that it returns an algorithm which is able to handle clustering whenever the minerule requires it

Definition at line 132 of file Algorithms.cpp.

```
00132                                                                                {
00133                 switch( mr.getParsedMinerule().miningTask ) {
00134                         case MTMineRules:            return getBestRulesMiningAlgorithm(mr);
00135                         case MTMineItemsets:      return getBestItemsetsMiningAlgorithm(mr);
00136                         case MTMineSequences:    return getBestSequencesMiningAlgorithm(mr);
00137                         default: throw MineruleException( MR_ERROR_INTERNAL, "Cannot handle
      "+miningTaskToString(mr.getParsedMinerule().miningTask)+" mining task");
00138                 }
00139         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Algorithms.hpp
- /Users/esposito/Software/minerule/src/Algorithms/Algorithms.cpp

6.9 minerule::AlgorithmsOptions Class Reference

`#include <AlgorithmsOptions.hpp>`

Public Member Functions

- AlgorithmsOptions ()
- void setSupport (double sup)
- void setConfidence (double conf)
- void setHeadCardinalities (const MinMaxPair &n)
- void setBodyCardinalities (const MinMaxPair &n)
- void setConnection (mrdb::Connection *connection)
- void setStatement (mrdb::PreparedStatement *statement)
- void setSourceRowDescription (const SourceRowColumnIds &srdes)
- void setOutTableName (const std::string &fname)
- void setMiningAlgorithmsOptions (const MineruleOptions::MiningAlgorithms &opt)
- double getSupport () const
- double getConfidence () const
- const std::string & getOutTableName () const
- const MineruleOptions::MiningAlgorithms & getMiningAlgorithmsOptions () const
- const MinMaxPair & getHeadCardinalities () const
- const MinMaxPair & getBodyCardinalities () const
- const SourceRowColumnIds & getSourceRowDescription () const

6.9.1 Detailed Description

Definition at line 30 of file AlgorithmsOptions.hpp.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 AlgorithmsOptions()

```
minerule::AlgorithmsOptions::AlgorithmsOptions ( )  [inline]
```

Definition at line 52 of file AlgorithmsOptions.hpp.

```
00053       : headCardinalities(MinMaxPair(1, 1000)),
00054         bodyCardinalities(MinMaxPair(1, 1000)){};
```

6.9.3 Member Function Documentation

6.9.3.1 getBodyCardinalities()

```
const MinMaxPair & minerule::AlgorithmsOptions::getBodyCardinalities ( ) const  [inline]
```

Returns

the MinMaxPair specifying the minimum/maximum number of elements that can be present in the body of a mined rule.

Definition at line 126 of file AlgorithmsOptions.hpp.

```
00126 { return bodyCardinalities; }
```

6.9.3.2 getConfidence()

```
double AlgorithmsOptions::getConfidence ( ) const
```

Returns

the current confidence threshold

Definition at line 31 of file AlgorithmsOptions.cpp.

```
00031 { return confidence; }
```

6.9.3.3 getHeadCardinalities()

```
const MinMaxPair & minerule::AlgorithmsOptions::getHeadCardinalities ( ) const  [inline]
```

Returns

the MinMaxPair specifying the minimum/maximum number of elements that can be present in the head of a mined rule.

Definition at line 122 of file AlgorithmsOptions.hpp.

```
00122 { return headCardinalities; }
```

6.9.3.4 getMiningAlgorithmsOptions()

```
const MineruleOptions::MiningAlgorithms & minerule::AlgorithmsOptions::getMiningAlgorithms↩
Options ( ) const  [inline]
```

Returns

the mining options

Definition at line 116 of file AlgorithmsOptions.hpp.

```
00116                                                                    {
00117     return miningAlgorithmsOptions;
00118   }
```

6.9.3.5 getOutTableName()

```
const std::string & minerule::AlgorithmsOptions::getOutTableName ( ) const  [inline]
```

Returns

the name of the output table

Definition at line 113 of file AlgorithmsOptions.hpp.

```
00113 { return outTableName; }
```

6.9.3.6 getSourceRowDescription()

```
const SourceRowColumnIds & AlgorithmsOptions::getSourceRowDescription ( ) const
```

Returns

the source row description the algorithm should use to figure out which columns of the source table are to interpreted as head/body/group elements.

Definition at line 40 of file AlgorithmsOptions.cpp.

```
00040                                                                    {
00041   return sourceRowDescription;
00042 }
```

6.9.3.7 getSupport()

```
double AlgorithmsOptions::getSupport ( ) const
```

Returns

the current support threshold

Definition at line 29 of file AlgorithmsOptions.cpp.

```
00029 { return support; }
```

6.9.3.8 setBodyCardinalities()

```
void minerule::AlgorithmsOptions::setBodyCardinalities (
            const MinMaxPair & n )  [inline]
```

Sets the body cardinalities: i.e., the minimum/maximum number of elements that can be present in the body a mined rule.

Parameters

n	a min/max pair

Definition at line 72 of file AlgorithmsOptions.hpp.

```
00072 { bodyCardinalities = n; }
```

6.9.3.9 setConfidence()

```
void AlgorithmsOptions::setConfidence (
            double conf )
```

Sets the confidence threshold

Parameters

conf	the confidence threshold

Definition at line 22 of file AlgorithmsOptions.cpp.

```
00022 { confidence = conf; }
```

6.9.3.10 setConnection()

```
void minerule::AlgorithmsOptions::setConnection (
            mrdb::Connection * connection )
```

Sets the mrdb connection to be used

Parameters

connection	the connection to be set. The class will store a weak reference to the connection. Deallocation remains therefore under the caller responsibility

6.9.3.11 setHeadCardinalities()

```
void minerule::AlgorithmsOptions::setHeadCardinalities (
            const MinMaxPair & n )  [inline]
```

Sets head cardinalities: i.e., the minimum/maximum number of elements that can be present in the head a mined rule.

Parameters

n	a min/max pair

Definition at line 67 of file AlgorithmsOptions.hpp.

```
00067 { headCardinalities = n; }
```

6.9.3.12 setMiningAlgorithmsOptions()

```
void minerule::AlgorithmsOptions::setMiningAlgorithmsOptions (
            const MineruleOptions::MiningAlgorithms & opt )  [inline]
```

Sets the options pertaining to the specific mining algorithm being executed (e.g., number of partitions for the Partition algorithm.)

Definition at line 102 of file AlgorithmsOptions.hpp.

```
00102                                                                    {
00103     miningAlgorithmsOptions = opt;
00104   }
```

6.9.3.13 setOutTableName()

```
void minerule::AlgorithmsOptions::setOutTableName (
            const std::string & fname )  [inline]
```

Sets the name of the output table. The output table will be created by the mining algorithm and should not exists on the database. If it exists an error will be generated.

Parameters

fname	the name of the output table

Definition at line 96 of file AlgorithmsOptions.hpp.
```
00096 { outTableName = fname; }
```

6.9.3.14 setSourceRowDescription()

```
void AlgorithmsOptions::setSourceRowDescription (
            const SourceRowColumnIds & srdes )
```

Sets the source row description needed by algorithms to distinguish body/head/group attributes in the mining set.

Parameters

srdes	a source row description

Definition at line 24 of file AlgorithmsOptions.cpp.
```
00025                                       {
00026   sourceRowDescription = srdes;
00027 }
```

6.9.3.15 setStatement()

```
void minerule::AlgorithmsOptions::setStatement (
            mrdb::PreparedStatement * statement )
```

Sets the prepared statement needed to build the result set that has to be mined.

Parameters

statement	the statement to be set. The class will store a weak reference to the statement. Deallocation remains therefore under the caller responsibility

6.9.3.16 setSupport()

```
void AlgorithmsOptions::setSupport (
            double sup )
```

Sets the support threshold

Parameters

sup	the support threshold

Definition at line 20 of file AlgorithmsOptions.cpp.
```
00020 { support = sup; }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/AlgorithmsOptions.hpp
- /Users/esposito/Software/minerule/src/Algorithms/AlgorithmsOptions.cpp

6.10 minerule::Attr_def Struct Reference

`#include <STSMiner.hpp>`

Public Member Functions

- Attr_def (std::string n, std::string t)

Data Fields

- std::string attr_name
- std::string attr_type

6.10.1 Detailed Description

Definition at line 22 of file STSMiner.hpp.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Attr_def()

```
minerule::Attr_def::Attr_def (
            std::string n,
            std::string t )  [inline]
```

Definition at line 26 of file STSMiner.hpp.

```
00026 : attr_name(n), attr_type(t) { }
```

6.10.3 Field Documentation

6.10.3.1 attr_name

```
std::string minerule::Attr_def::attr_name
```

Definition at line 23 of file STSMiner.hpp.

6.10.3.2 attr_type

```
std::string minerule::Attr_def::attr_type
```

Definition at line 24 of file STSMiner.hpp.

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp

6.11 Attribute Class Reference

```
#include <ParsedMinerule.hpp>
```

Public Member Functions

- Attribute (const char *s)
- bool operator< (const Attribute &a) const

Data Fields

- std::string name

6.11.1 Detailed Description

Definition at line 56 of file ParsedMinerule.hpp.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Attribute()

```
Attribute::Attribute (
            const char * s )
```

6.11.3 Member Function Documentation

6.11.3.1 operator<()

```
bool Attribute::operator< (
            const Attribute & a ) const
```

6.11.4 Field Documentation

6.11.4.1 name

```
std::string Attribute::name
```

Definition at line 59 of file ParsedMinerule.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.12 AttributesUtil Class Reference

Public Member Functions

- AttributesUtil ()
- std::vector< int > generatePositions (const ParsedMinerule::AttrVector &attrs)

Static Public Member Functions

- static std::string names_to_string (const ParsedMinerule::AttrVector &attrs)

6.12.1 Detailed Description

Definition at line 124 of file SourceRowMetaInfo.cpp.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 AttributesUtil()

```
AttributesUtil::AttributesUtil ( )  [inline]
```

Definition at line 127 of file SourceRowMetaInfo.cpp.

```
00127 : curr_attr_pos(1) {}
```

6.12.3 Member Function Documentation

6.12.3.1 generatePositions()

```
std::vector< int > AttributesUtil::generatePositions (
            const ParsedMinerule::AttrVector & attrs )  [inline]
```

Definition at line 129 of file SourceRowMetaInfo.cpp.

```
00129                                                                                {
00130                 std::vector<int> result;
00131                 for( ParsedMinerule::AttrVector::const_iterator it = attrs.begin(); it!=attrs.end();
      ++it ) {
00132                         result.push_back(curr_attr_pos++);
00133                 }
00134                 return result;
00135         }
```

6.12.3.2 names_to_string()

```
static std::string AttributesUtil::names_to_string (
            const ParsedMinerule::AttrVector & attrs )  [inline], [static]
```

Definition at line 137 of file SourceRowMetaInfo.cpp.

```
00137                                                                                {
00138                 std::string result;
00139                 for( ParsedMinerule::AttrVector::const_iterator it = attrs.begin(); it!=attrs.end();
      ++it ) {
00140                         if( it != attrs.begin() ) result += ",";
00141                         result += *it;
00142                 }
00143
00144                 return result;
00145         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/src/Database/SourceRowMetaInfo.cpp

6.13 minerule::Bem_cond Class Reference

```
#include <ParsedMinerule.hpp>
```

Public Member Functions

- Bem_cond ()
- Bem_cond (const minerule::Bem_cond &copy_me)

Static Public Member Functions

- static std::vector< Bem_cond * > copyBemCond (std::vector< Bem_cond * > copy_me)

Data Fields

- std::string type
- std::string attr
- std::string op
- std::string val
- int count_min
- int count_max
- Bem_cond * and_c

6.13.1 Detailed Description

Definition at line 67 of file ParsedMinerule.hpp.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Bem_cond() [1/2]

```
minerule::Bem_cond::Bem_cond ( )  [inline]
```

Definition at line 77 of file ParsedMinerule.hpp.

```
00077             {
00078         type="";
00079         attr="";
00080         op="";
00081         val="";
00082         count_min=1;
00083         count_max=std::numeric_limits<int>::max();
00084         and_c=NULL;
00085     }
```

6.13.2.2 Bem_cond() [2/2]

```
minerule::Bem_cond::Bem_cond (
            const minerule::Bem_cond & copy_me )  [inline]
```

Definition at line 87 of file ParsedMinerule.hpp.

```
00087                                                {
00088         type=copy_me.type;
00089         attr=copy_me.attr;
00090         op=copy_me.op;
00091         val=copy_me.val;
00092         count_min=copy_me.count_min;
00093         count_max=copy_me.count_max;
00094         if(copy_me.and_c!=NULL)
00095             and_c= new Bem_cond(*(copy_me.and_c));
00096         else
00097             and_c=NULL;
00098     }
```

6.13.3 Member Function Documentation

6.13.3.1 copyBemCond()

```
static std::vector< Bem_cond * > minerule::Bem_cond::copyBemCond (
            std::vector< Bem_cond * > copy_me )  [inline], [static]
```

Definition at line 100 of file ParsedMinerule.hpp.

```
00100                                                                    {
00101         std::vector<Bem_cond*> out;
00102         for(int i=0; i<copy_me.size(); ++i)
00103             out.push_back(new Bem_cond(*copy_me[i]));
00104         return out;
00105     }
```

6.13.4 Field Documentation

6.13.4.1 and_c

```
Bem_cond* minerule::Bem_cond::and_c
```

Definition at line 75 of file ParsedMinerule.hpp.

6.13.4.2 attr

```
std::string minerule::Bem_cond::attr
```

Definition at line 70 of file ParsedMinerule.hpp.

6.13.4.3 count_max

```
int minerule::Bem_cond::count_max
```

Definition at line 74 of file ParsedMinerule.hpp.

6.13.4.4 count_min

```
int minerule::Bem_cond::count_min
```

Definition at line 73 of file ParsedMinerule.hpp.

6.13.4.5 op

```
std::string minerule::Bem_cond::op
```

Definition at line 71 of file ParsedMinerule.hpp.

6.13.4.6 type

```
std::string minerule::Bem_cond::type
```

Definition at line 69 of file ParsedMinerule.hpp.

6.13.4.7 val

```
std::string minerule::Bem_cond::val
```

Definition at line 72 of file ParsedMinerule.hpp.

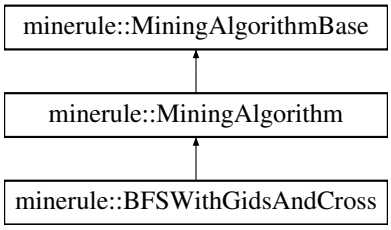
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.14 minerule::BFSWithGidsAndCross Class Reference

```
#include <BFSWithGidsAndCross.hpp>
```

Inheritance diagram for minerule::BFSWithGidsAndCross:



Public Member Functions

- BFSWithGidsAndCross (const OptimizedMinerule &mr)
- virtual ~BFSWithGidsAndCross ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual void initialize ()
- virtual void execute ()
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static bool getMineruleHasSameBodyHead ()
- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- AlgorithmsOptions options
- Connection connection
- const OptimizedMinerule & minerule

6.14.1 Detailed Description

Definition at line 25 of file BFSWithGidsAndCross.hpp.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 BFSWithGidsAndCross()

```
minerule::BFSWithGidsAndCross::BFSWithGidsAndCross (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 141 of file BFSWithGidsAndCross.hpp.

```
00141                                                   :
00142       MiningAlgorithm(mr), sourceTable(NULL) {}
```

6.14.2.2 ~BFSWithGidsAndCross()

```
virtual minerule::BFSWithGidsAndCross::~BFSWithGidsAndCross ( )  [inline], [virtual]
```

Definition at line 144 of file BFSWithGidsAndCross.hpp.

```
00144                                       {
00145         if(sourceTable!=NULL) delete sourceTable;
00146     }
```

6.14.3 Member Function Documentation

6.14.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.14.3.2 canHandleMinerule()

```
virtual bool minerule::BFSWithGidsAndCross::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 158 of file BFSWithGidsAndCross.hpp.

```
00158                                          {
00159       return !minerule.getParsedMinerule().requiresClusters() &&
       !minerule.getParsedMinerule().hasDisjuctionsInMC();
00160     }
```

6.14.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                    {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.14.3.4 getMineruleHasSameBodyHead()

```
static bool minerule::BFSWithGidsAndCross::getMineruleHasSameBodyHead ( )  [inline], [static]
```

Definition at line 152 of file BFSWithGidsAndCross.hpp.

```
00152                                                {
00153                 return mineruleHasSameBodyHead;
00154         }
```

6.14.3.5 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                   {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.14.3.6 mineRules()

```
void minerule::BFSWithGidsAndCross::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 297 of file BFSWithGidsAndCross.cpp.

```
00297                                   {
00298                 MRLogPush("Starting BFSWithGidsAndCross mining algorithm...");
00299
00300                 MRLog() « "Preparing data sources..." « std::endl;
00301                 prepareData();
00302                 Progress progress(200);
00303
00304                 float support = options.getSupport();
00305                 int maxBody = options.getBodyCardinalities().getMax();
00306                 int maxHead = options.getHeadCardinalities().getMax();
00307
00308                 ItemType gid1;
00309                 BodyMap bodyMap(connection, progress);
00310
00311                 int totalGroups = sourceTable->getTotGroups();
00312                 int howManyRows = 0;
00313                 int howManyGroups = 0;
00314
00315
00316                 MRLogPush("Reading data...");
00317
00318                 while (!ruleIterator.isAfterLast()) {
00319                         gid1 = ruleIterator->getGroup();
00320                         howManyGroups++;
00321
00322                         Transaction t1;
00323
00324                         t1.load(gid1,ruleIterator);
00325                         howManyRows += bodyMap.add(howManyGroups,t1);
00326                 }
00327
```

```
00328                 MRLog() « "Total groups: " « totalGroups « std::endl;
00329                 MRLogPop();
00330
00331                 MRLogPush("Starting rule extraction...");
00332
00333                 bodyMap.updateCount();
00334                 MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00335                 bodyMap.pruneMap(support*totalGroups);
00336                 MRLog() « "Total bodies after pruning: " « bodyMap.size() « std::endl;
00337                 NewRuleSet rs;
00338
00339                 progress.start();
00340                 int nrules = bodyMap.generateRules(support,totalGroups,maxBody,maxHead);
00341                 progress.end();
00342                 MRLog() « "After extracting rules, rules: " « nrules « std::endl;
00343
00344                 MRLogPop();
00345                 connection.finalize();
00346
00347                 MRLogPop();
00348         }
```

6.14.3.7 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.14.3.8 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::BFSWithGidsAndCross::sourceTableRequirements ( )
const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 148 of file BFSWithGidsAndCross.hpp.

```
00148                                                                                 {
00149                         return SourceTableRequirements(SourceTableRequirements::CrossProduct |
      SourceTableRequirements::SortedGids);
00150                 };
```

6.14.4 Field Documentation

6.14.4.1 connection

```
Connection minerule::MiningAlgorithm::connection  [protected], [inherited]
```

Definition at line 62 of file MiningAlgorithmBase.hpp.

6.14.4.2 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

6.14.4.3 options

```
AlgorithmsOptions minerule::MiningAlgorithm::options  [protected], [inherited]
```

Definition at line 61 of file MiningAlgorithmBase.hpp.

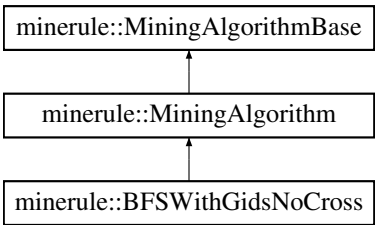
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsAndCross.hpp
- /Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsAndCross.cpp

6.15 minerule::BFSWithGidsNoCross Class Reference

```
#include <BFSWithGidsNoCross.hpp>
```

Inheritance diagram for minerule::BFSWithGidsNoCross:



Public Member Functions

- BFSWithGidsNoCross (const OptimizedMinerule &mr)
- virtual ~BFSWithGidsNoCross ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual void initialize ()
- virtual void execute ()
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static bool getMineruleHasSameBodyHead ()
- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- AlgorithmsOptions options
- Connection connection
- const OptimizedMinerule & minerule

6.15.1 Detailed Description

Definition at line 26 of file BFSWithGidsNoCross.hpp.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 BFSWithGidsNoCross()

```
minerule::BFSWithGidsNoCross::BFSWithGidsNoCross (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 125 of file BFSWithGidsNoCross.hpp.

```
00125                                                       : MiningAlgorithm(mr),
      sourceTable(NULL) {
00126                       mineruleHasSameBodyHead = mr.getParsedMinerule().hasSameBodyHead();
00127                 }
```

6.15.2.2 ~BFSWithGidsNoCross()

```
virtual minerule::BFSWithGidsNoCross::~BFSWithGidsNoCross ( )  [inline], [virtual]
```

Definition at line 129 of file BFSWithGidsNoCross.hpp.

```
00129                                   {
00130                       if(sourceTable!=NULL) delete sourceTable;
00131                 }
```

6.15.3 Member Function Documentation

6.15.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                   {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.15.3.2 canHandleMinerule()

```
virtual bool minerule::BFSWithGidsNoCross::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 144 of file BFSWithGidsNoCross.hpp.

```
00144                                                   {
00145                         return
00146                                 !minerule.getParsedMinerule().hasCrossConditions() &&
00147                                 !minerule.getParsedMinerule().requiresClusters() &&
00148                                 !minerule.getParsedMinerule().hasDisjuctionsInMC();
00149                 }
```

6.15.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                   {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.15.3.4 getMineruleHasSameBodyHead()

```
static bool minerule::BFSWithGidsNoCross::getMineruleHasSameBodyHead ( )  [inline], [static]
```

Definition at line 140 of file BFSWithGidsNoCross.hpp.

```
00140                                                    {
00141                         return mineruleHasSameBodyHead;
00142                 }
```

6.15.3.5 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                   {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
```

```
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.15.3.6 mineRules()

```
void minerule::BFSWithGidsNoCross::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 243 of file BFSWithGidsNoCross.cpp.

```
00243                                       {
00244     MRLogPush("Starting BFSWithGidsNoCross mining algorithm...");
00245
00246     MRLog() « "Preparing data sources..." « std::endl;
00247     prepareData();
00248                 Progress progress(200);
00249
00250     float support = options.getSupport();
00251     int maxBody = options.getBodyCardinalities().getMax();
00252     int maxHead = options.getHeadCardinalities().getMax();
00253
00254     ItemType gid1;
00255
00256     if(bodyIterator.isAfterLast())
00257                 throw new MineruleException(MR_ERROR_INTERNAL,"Cannot find initial GID for body
      elements");
00258     if(headIterator.isAfterLast())
00259                 throw new MineruleException(MR_ERROR_INTERNAL,"Cannot find initial GID for head
      elements");
00260
00261         MRLogPush("Reading data");
00262     BodyMap bodyMap(connection, progress);
00263
00264     int totalGroups = sourceTable->getTotGroups();
00265     int howManyRows = 0;
00266     int howManyGroups = 0;
00267
00268                 Progress readProgress(10000);
00269                 readProgress.start();
00270     while (!bodyIterator.isAfterLast()) {
00271       ItemType gid = bodyIterator->getGroup();
00272
00273           Transaction t1, t2;
00274       t1.loadBody(gid,bodyIterator);
00275
00276       bool found2 = t2.findGid(gid,headIterator);
00277       if (found2) {
00278                   t2.loadHead(gid,headIterator);
00279       }
00280
00281       howManyRows += bodyMap.add(howManyGroups,t1,t2);
00282       howManyGroups++;
00283
00284                         readProgress.tick();
00285     }
00286                 readProgress.end();
00287
00288     MRLog() « "Total groups: " « totalGroups « std::endl;
00289         MRLogPop();
00290
00291     MRLogPush("Starting rule extraction...");
00292
00293     bodyMap.updateCount();
00294
00295     MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
```

```
00296     bodyMap.pruneMap(support*totalGroups);
00297     MRLog() << "Total bodies after pruning: " << bodyMap.size() << std::endl;
00298     NewRuleSet rs;
00299 
00300                progress.start();
00301     int nrules = bodyMap.generateRules(support,totalGroups,maxBody,maxHead);
00302                progress.end();
00303 
00304     MRLog() << "After extracting rules, rules: " << nrules << std::endl;
00305 
00306     MRLogPop();
00307 
00308     MRLogPop();
00309 
00310     connection.finalize();
00311   }
```

6.15.3.7 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.15.3.8 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::BFSWithGidsNoCross::sourceTableRequirements ( )
const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 133 of file BFSWithGidsNoCross.hpp.

```
00133                                                                    {
00134                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00135                 };
```

6.15.4 Field Documentation

6.15.4.1 connection

```
Connection minerule::MiningAlgorithm::connection  [protected], [inherited]
```

Definition at line 62 of file MiningAlgorithmBase.hpp.

6.15.4.2 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

6.15.4.3 options

```
AlgorithmsOptions minerule::MiningAlgorithm::options  [protected], [inherited]
```

Definition at line 61 of file MiningAlgorithmBase.hpp.

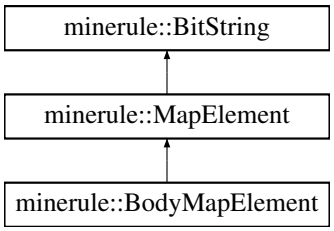
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsNoCross.hpp
- /Users/esposito/Software/minerule/src/Algorithms/BFSWithGidsNoCross.cpp

6.16 minerule::BitString Class Reference

```
#include <Bitstring.hpp>
```

Inheritance diagram for minerule::BitString:



Public Member Functions

- boolean test (int i) const
- BitString ()
- BitString (int nbits)
- BitString (const BitString &bs)
- BitString & set ()
- BitString & set (int i, boolean value=true)
- BitString & reset ()
- BitString & reset (int i)
- BitString & clear ()
- BitString & clear (int i)
- BitString & invert ()
- BitString & invert (int i)
- BitString & operator&= (const BitString &bs)
- BitString & operator|= (const BitString &bs)

- BitString & operator^= (const BitString &bs)
- BitString & operator= (const BitString &bs)
- boolean operator== (const BitString &bs)
- boolean operator!= (const BitString &bs)
- BitString operator& (const BitString &bs1)
- int count (boolean what=true) const
- bool moreThan (double threshold) const
- int length () const
- int ssize () const
- int size () const
- void serialize (char *serialized, int *start)
- void unserialize (char *serialized, int *start)
- void print ()
- void print (std::ostream &out)
- void setNBit (int num)
- void * getElmA (int which)
- Bits getElm (int which)
- void insert (Bits element)
- std::vector< Bits > & gbits ()
- char operator[] (int which) const

Static Public Attributes

- static int intersections = 0

Friends

- std::ostream & operator<< (std::ostream &out, const BitString &bs)
- std::istream & operator>> (std::istream &in, BitString &bs)

6.16.1 Detailed Description

Definition at line 47 of file Bitstring.hpp.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 BitString() [1/3]

```
minerule::BitString::BitString ( )
```

Definition at line 63 of file Bitstring.cpp.

```
00063 { init(0); }
```

6.16.2.2 BitString() [2/3]

```
minerule::BitString::BitString (
            int nbits )
```

Definition at line 65 of file Bitstring.cpp.

```
00066 {
00067 init(nbits);
00068 }
```

6.16.2.3 BitString() [3/3]

```
minerule::BitString::BitString (
            const BitString & bs )
```

Definition at line 70 of file Bitstring.cpp.

```
00070                                          {
00071 n = bs.n;
00072 Nw = bs.Nw;
00073 for (int i=0; i<=Nw; i++) bits.insert(bits.end(),bs.bits[i]);
00074 }
```

6.16.3 Member Function Documentation

6.16.3.1 clear() [1/2]

```
BitString & minerule::BitString::clear ( )
```

Definition at line 121 of file Bitstring.cpp.

```
00122 {
00123 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00124 return *this;
00125 }
```

6.16.3.2 clear() [2/2]

```
BitString & minerule::BitString::clear (
            int i )
```

Definition at line 131 of file Bitstring.cpp.

```
00132 {
00133 return set(i, false);
00134 }
```

6.16.3.3 count()

```
int minerule::BitString::count (
            boolean what = true ) const
```

Definition at line 284 of file Bitstring.cpp.

```
00285 {
00286 int i,j, k=0;
00287 for (i=0; i<=Nw; i++) {
00288 //      Bits b = bits[i];
00289 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b>>=1;}
00290         for (j=0; j<sizeof(Bits); j++)
00291         k += NBITS[*(((unsigned char*)&bits[i])+j)];
00292 }
00293 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00294 return k;
00295 }
```

6.16.3.4 gbits()

```
std::vector< Bits > & minerule::BitString::gbits ( )  [inline]
```

Definition at line 101 of file Bitstring.hpp.

```
00101 {  return bits; }
```

6.16.3.5 getElm()

```
Bits minerule::BitString::getElm (
            int which )  [inline]
```

Definition at line 99 of file Bitstring.hpp.

```
00099 {  return bits[which]; }
```

6.16.3.6 getElmA()

```
void * minerule::BitString::getElmA (
            int which )  [inline]
```

Definition at line 98 of file Bitstring.hpp.

```
00098 {  return &(bits[which]); }
```

6.16.3.7 insert()

```
void minerule::BitString::insert (
            Bits element )  [inline]
```

Definition at line 100 of file Bitstring.hpp.

```
00100 {  bits.push_back(element); }
```

6.16.3.8 invert() [1/2]

```
BitString & minerule::BitString::invert ( )
```

Definition at line 140 of file Bitstring.cpp.

```
00141 {
00142 for (int i=0; i<=Nw; i++) bits[i] = ~bits[i];
00143 return *this;
00144 }
```

6.16.3.9 invert() [2/2]

```
BitString & minerule::BitString::invert (
            int i )
```

Definition at line 150 of file Bitstring.cpp.

```
00151 {
00152 if (n <= i) _Xran(i);
00153 bits[i/Nb] ^= (Bits)1 « i%Nb;
00154 return *this;
00155 }
```

6.16.3.10 length()

```
int minerule::BitString::length ( ) const  [inline]
```

Definition at line 87 of file Bitstring.hpp.

```
00087 { return n; }
```

6.16.3.11 moreThan()

```
bool minerule::BitString::moreThan (
            double threshold ) const
```

Definition at line 301 of file Bitstring.cpp.

```
00302 {
00303 int i,j, k=(int)(threshold);
00304 for (i=0; i<=Nw && k >= 0; i++) {
00305 //      Bits b = bits[i];
00306 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b»=1;}
00307         for (j=0; j<sizeof(Bits); j++)
00308         k -= NBITS[*(((unsigned char*)&bits[i])+j)];
00309 }
00310 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00311 return k<0;
00312 }
```

6.16.3.12 operator"!=()

```
boolean minerule::BitString::operator!= (
            const BitString & bs )
```

Definition at line 263 of file Bitstring.cpp.

```
00264 {
00265 return !operator==(bs);
00266 }
```

6.16.3.13 operator&()

```
BitString minerule::BitString::operator& (
            const BitString & bs1 )
```

Definition at line 223 of file Bitstring.cpp.

```
00224 {
00225         BitString bs;
00226         int min = Nw < bs1.Nw ? Nw : bs1.Nw;
00227         for (int i=0; i<=min; i++) bs.bits.insert(bs.bits.end(),bits[i] & bs1.bits[i]);
00228         bs.Nw = min;
00229         bs.n = n < bs1.n ? n : bs1.n;
00230         return bs;
00231 }
```

6.16.3.14 operator&=()

```
BitString & minerule::BitString::operator&= (
            const BitString & bs )
```

Definition at line 195 of file Bitstring.cpp.

```
00196 {
00197 /*
00198 if (n < bs.n) {
00199         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n);
00200         n = bs.n;
00201 }
00202 if (Nw < bs.Nw) {
00203         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), ~(char)0); }
00204         n = bs.n;
00205 }
00206 for (int i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00207 */
00208 if (Nw < bs.Nw) {
00209         for (int i=0; i<=Nw; i++) bits[i] &= bs.bits[i];
00210 } else {
00211         int i;
00212         for (i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00213         for (; i<=Nw; i++) bits[i] = (Bits)0;
00214 }
00215 intersections++;
00216 return *this;
00217 }
```

6.16.3.15 operator=()

```
BitString & minerule::BitString::operator= (
            const BitString & bs )
```

Definition at line 237 of file Bitstring.cpp.

```
00238 {
00239 bits.erase(bits.begin(),bits.end());
00240 n = bs.n;
00241 Nw = bs.Nw;
00242 for (int i=0; i<=Nw; i++) bits.insert(bits.end(),bs.bits[i]);
00243 //for (int i=0; i<=Nw && i<=bs.Nw; i++) bits[i] = bs.bits[i];
00244 return *this;
00245 }
```

6.16.3.16 operator==()

```
boolean minerule::BitString::operator== (
            const BitString & bs )
```

Definition at line 251 of file Bitstring.cpp.

```
00252 {
00253 boolean eq = n == bs.n;
00254 for (int i=0; i<Nw && eq; i++) eq = bits[i] == bs.bits[i];
00255 for (int i=(Nw-1)*Nb; i<n && eq; i++) eq = test(i) == bs.test(i);
00256 return eq;
00257 }
```

6.16.3.17 operator[]()

```
char minerule::BitString::operator[] (
            int which ) const  [inline]
```

Definition at line 103 of file Bitstring.hpp.

```
00103 { return (test(which) ? '1' : '0'); }
```

6.16.3.18 operator^=()

```
BitString & minerule::BitString::operator^= (
            const BitString & bs )
```

Definition at line 178 of file Bitstring.cpp.

```
00179 {
00180 if (n < bs.n) {
00181         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n,bs.test(n));
00182         n = bs.n;
00183 }
00184 if (Nw < bs.Nw) {
00185         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), bs.bits[Nw]); }
00186 }
00187 for (int i=0; i<=bs.Nw; i++) bits[i] ^= bs.bits[i];
00188 return *this;
00189 }
```

6.16.3.19 operator"|=()

```
BitString & minerule::BitString::operator|= (
            const BitString & bs )
```

Definition at line 161 of file Bitstring.cpp.

```
00162 {
00163 if (n < bs.n) {
00164         for (; n < (Nw+1)*Nb && n < bs.n; n++) reset(n);
00165         n = bs.n;
00166 }
00167 if (Nw < bs.Nw) {
00168         for (; Nw <= bs.Nw; ) { Nw++; bits.insert(bits.end(), (Bits)0); }
00169 }
00170 for (int i=0; i<=bs.Nw; i++) bits[i] |= bs.bits[i];
00171 return *this;
00172 }
```

6.16.3.20 print() [1/2]

```
void minerule::BitString::print ( )
```

Definition at line 404 of file Bitstring.cpp.

```
00405 {int i,dim;
00406
00407  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00408  {
00409  dim=(*this).length();
00410  for (i=0;i<dim;i++)
00411  {
00412   if ((i%8==0)&&(i!=0)) std::cout«"-";
00413   std::cout«test(i);
00414  }
00415  }
00416 }
```

6.16.3.21 print() [2/2]

```
void minerule::BitString::print (
            std::ostream & out )
```

Definition at line 390 of file Bitstring.cpp.

```
00391 {int i,dim;
00392
00393  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00394  {
00395  dim=(*this).length();
00396  for (i=0;i<dim;i++)
00397  {
00398   if ((i%8==0)&&(i!=0)) out«"-";
00399   out«test(i);
00400  }
00401  }
00402 }
```

6.16.3.22 reset() [1/2]

```
BitString & minerule::BitString::reset ( )
```

Definition at line 102 of file Bitstring.cpp.

```
00103 {
00104 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00105 return *this;
00106 }
```

6.16.3.23 reset() [2/2]

```
BitString & minerule::BitString::reset (
            int i )
```

Definition at line 112 of file Bitstring.cpp.

```
00113 {
00114 return set(i, false);
00115 }
```

6.16.3.24 serialize()

```
void minerule::BitString::serialize (
            char * serialized,
            int * start )
```

Definition at line 352 of file Bitstring.cpp.

```
00353 {int i;
00354
00355  memcpy(&(serialized[(*start)]),&Nw,sizeof(int));
00356         //cout<<int(serialized[(*start)])<<" "<<std::endl;
00357  //cout<<"\"";
00358  (*start) += sizeof(int);
00359  memcpy(&(serialized[(*start)]),&n,sizeof(int));
00360  (*start) += sizeof(int);
00361         for (i=0;i<=Nw;i++)
00362         {
00363          memcpy(&(serialized[(*start)]),&(bits[i]),sizeof(Bits));
00364          //cout<<(int)(serialized[(*start)])<<" ";
00365                 (*start) += sizeof(Bits);
00366         }
00367 }
```

6.16.3.25 set() [1/2]

```
BitString & minerule::BitString::set ( )
```

Definition at line 80 of file Bitstring.cpp.

```
00081 {
00082 for (int i=0; i<=Nw; i++) bits[i] = ~(Bits)0;
00083 return *this;
00084 }
```

6.16.3.26 set() [2/2]

```
BitString & minerule::BitString::set (
            int i,
            boolean value = true )
```

Definition at line 90 of file Bitstring.cpp.

```
00091 {
00092 if (n <= i) _Xran(i);
00093 if (value) bits[i/Nb] |= (Bits)1 « i%Nb;
00094 else bits[i/Nb] &= ~((Bits)1 « i%Nb);
00095 return *this;
00096 }
```

6.16.3.27 setNBit()

```
void minerule::BitString::setNBit (
            int num )  [inline]
```

Definition at line 97 of file Bitstring.hpp.

```
00097 {  Nw = (num == 0) ? 0 : (num-1) / Nb;n=num; }
```

6.16.3.28 size()

```
int minerule::BitString::size ( ) const  [inline]
```

Definition at line 89 of file Bitstring.hpp.

```
00089 { return n; }
```

6.16.3.29 ssize()

```
int minerule::BitString::ssize ( ) const  [inline]
```

Definition at line 88 of file Bitstring.hpp.

```
00088 { return bits.size(); }
```

6.16.3.30 test()

```
boolean minerule::BitString::test (
            int i ) const  [inline]
```

Definition at line 58 of file Bitstring.hpp.

```
00058                                   {
00059                         if (i<0 || n <= i) return false;
00060                         return ((bits[i/Nb] & ((Bits)1 « i%Nb)) != 0);
00061                 }
```

6.16.3.31 unserialize()

```
void minerule::BitString::unserialize (
            char * serialized,
            int * start )
```

Definition at line 369 of file Bitstring.cpp.

```
00370 {int i;
00371  char tmpchar;
00372
00373  memcpy(&Nw,&(serialized[(*start)]),sizeof(int));
00374         //cout«int(serialized[(*start)])«" "«std::endl;
00375  //cout«"\"";
00376  (*start) += sizeof(int);
00377  memcpy(&n,&(serialized[(*start)]),sizeof(int));
00378  (*start) += sizeof(int);
00379  for (i=bits.size();i<=Nw;i++) bits.insert(bits.end(),(Bits)0);
00380         for (i=0;i<=Nw;i++)
00381         {
00382          memcpy(&tmpchar,&serialized[(*start)],sizeof(Bits));
00383          //cout«(int)(serialized[(*start)])«" ";
00384          bits[i] = tmpchar;
00385          (*start) += sizeof(Bits);
00386         }
00387 // setNBit(numbit);
00388 }
```

6.16.4 Friends And Related Function Documentation

6.16.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const BitString & bs )  [friend]
```

Definition at line 318 of file Bitstring.cpp.

```
00319 {
00320 for (int i=0; i<bs.n; i++) out « (bs.test(i) ? '1' : '0');
00321 return out;
00322 }
```

6.16.4.2 operator>>

```
std::istream & operator>> (
            std::istream & in,
            BitString & bs )  [friend]
```

Definition at line 328 of file Bitstring.cpp.

```
00329 {
00330 std::ios::iostate St = std::ios::goodbit;
00331 boolean Chg = false;
00332 int C = in.rdbuf()->sgetc();
00333 bs.reset();
00334 while (C != EOF && C == ' ') C = in.rdbuf()->snextc();
00335 for (size_t M = 0; true ; C = in.rdbuf()->snextc(), ++M)
00336 {
00337         if (C == EOF) {St |= std::ios::eofbit; break; }
00338         else if (C != '0' && C != '1') break;
00339         //else if (_X.allocation() <= _X.length())
00340         //       {_St |= ios::failbit;
00341         //       break; }
00342         else { bs.set(M,C=='1'); Chg = true; }
00343 }
00344 //if (!_Chg)  _St |= ios::failbit;
00345 in.setstate(St);
00346 return (in);
00347 }
```

6.16.5 Field Documentation

6.16.5.1 intersections

```
int minerule::BitString::intersections = 0  [static]
```

Definition at line 55 of file Bitstring.hpp.

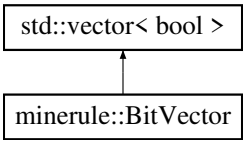
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Bitstring.hpp
- /Users/esposito/Software/minerule/src/Utils/Bitstring.cpp

6.17 minerule::BitVector Class Reference

```
#include <BitVector.hpp>
```

Inheritance diagram for minerule::BitVector:



Public Member Functions

- BitVector operator& (BitVector input)
- BitVector operator| (BitVector input)

Data Fields

- T elements

  STL member.

6.17.1 Detailed Description

Definition at line 22 of file BitVector.hpp.

6.17.2 Member Function Documentation

6.17.2.1 operator&()

```
BitVector minerule::BitVector::operator& (
            BitVector input )  [inline]
```

Definition at line 27 of file BitVector.hpp.

```
00027                                                   {
00028                         BitVector ris;
00029                         for (size_t i=0;i<this->size();++i)
00030                                 ris.push_back((*this)[i]&input[i]);
00031                         return ris;
00032                 }
```

6.17.2.2 operator"|()

```
BitVector minerule::BitVector::operator| (
            BitVector input )  [inline]
```

Definition at line 34 of file BitVector.hpp.

```
00034                                                   {
00035                         BitVector ris;
00036                         for (size_t i=0;i<this->size();++i)
00037                                 ris.push_back((*this)[i]|input[i]);
00038                         return ris;
00039                 }
```

6.17.3 Field Documentation

6.17.3.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/BitVector.hpp

6.18 minerule::Body Class Reference

```
#include <IncrAlgoClasses.hpp>
```

Public Types

- typedef std::map< ItemType, NodeRowB * > RowBContainer

Public Member Functions

- Body ()
- ~Body ()
- void setAncestor (ItemType b)
- ItemType getAncestor ()
- Head * insertItemSetB (ItemSet &SREV, double supp)
- void findBodiesInTree (ItemSet *body)
- void findChildInTree (ItemSet *b1, ItemSet *h1)
- void findRulesInTree (ItemSet *itemset_body, ItemSet *itemset_head)
- void findRulesInTree (ItemSet *b1, ItemSet *b1nb2, ItemSet *h1, ItemSet *h1nh2)
- void extractRules (std::vector< ItemType > &body, double thrR, double thrB, int ngroups, Connection *pconnection)
- void provaStampaLiv1 ()

Data Fields

- RowBContainer * NRB

Static Public Attributes

- static size_t countb =0

6.18.1 Detailed Description

Definition at line 34 of file IncrAlgoClasses.hpp.

6.18.2 Member Typedef Documentation

6.18.2.1 RowBContainer

```
typedef std::map<ItemType,NodeRowB*> minerule::Body::RowBContainer
```

Definition at line 36 of file IncrAlgoClasses.hpp.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 Body()

```
minerule::Body::Body ( )
```

Definition at line 30 of file IncrAlgoClasses.cpp.

```
00030           {
00031   NRB=new RowBContainer;
00032   ItemType* tmp=new ItemType();
00033   ancestor=*tmp;
00034   countb++;
00035   delete tmp;
00036  }
```

6.18.3.2 ~Body()

```
minerule::Body::~Body ( )
```

Definition at line 38 of file IncrAlgoClasses.cpp.

```
00038           {
00039     countb--;
00040     RowBContainer::iterator it;
00041     it=NRB->begin();
00042     while(it!=NRB->end()){
00043       delete (it->second);
00044       it++;
00045     }
00046     delete NRB;
00047   }
```

6.18.4 Member Function Documentation

6.18.4.1 extractRules()

```
void minerule::Body::extractRules (
            std::vector< ItemType > & body,
            double thrR,
            double thrB,
            int ngroups,
            Connection * pconnection )
```

Definition at line 459 of file IncrAlgoClasses.cpp.

```
00461                                                                   {
00462  typedef std::map<ItemType,NodeRowB*> RowBContainer;
00463
00464  RowBContainer::iterator itB;
00465  Body* btmp;
00466  itB=this->NRB->begin();
00467  std::vector<ItemType> head;
00468  while (itB!=this->NRB->end()){
00469    double suppb=itB->second->getSupp();
00470    body.push_back(itB->first);
00471    Head* h=itB->second->getHead();
00472    if (h!=NULL){
00473         itB->second->getHead()->extractRules(body,head,thrR,thrB,suppb,ngroups,pconnection);
00474    }
00475    btmp=itB->second->getChild();
00476    if (btmp!=NULL){
00477      btmp->extractRules(body,thrR,thrB,ngroups,pconnection);
00478    }
00479    body.pop_back();
00480    itB++;
00481  }
00482 }
```

6.18.4.2 findBodiesInTree()

```
void minerule::Body::findBodiesInTree (
            ItemSet * body )
```

Definition at line 178 of file IncrAlgoClasses.cpp.

```
00178                                        {
00179
00180   RowBContainer::iterator ite=NRB->begin();
00181   ItemSet::iterator itfind;
00182   while(ite!=NRB->end()){
00183     itfind=find(body->begin(),body->end(),ite->first);
00184     //se l'ho trovato
00185     if (itfind!=body->end()) {
00186       ite->second->incremSupp();
00187       //cout«"ho increm supp di body"«std::endl;
00188       Body* c=ite->second->getChild();
00189       //se ha figli
00190       if (c!=NULL) c->findBodiesInTree(body);
00191     }
00192   ite++;
00193   }
00194  }
```

6.18.4.3 findChildInTree()

```
void minerule::Body::findChildInTree (
            ItemSet * b1,
            ItemSet * h1 )
```

Definition at line 281 of file IncrAlgoClasses.cpp.

```
00281                                                {
00282
00283   RowBContainer::iterator ite=NRB->begin();
00284   ItemSet::iterator itfind;
00285   //cout«"esamino body "«ite->first«std::endl;
00286   while(ite!=NRB->end()){
00287     itfind=find(b1->begin(),b1->end(),ite->first);
00288     //se l'ho trovato
00289     if (itfind!=b1->end()) {
00290       ite->second->decremSupp();
00291       //cout«"                          decremento body "«ite->first«std::endl;
00292       //cout«"ho decrem supp di body"«std::endl;
00293       Head* h=ite->second->getHead();
00294       if (h!=NULL) {
00295         h->findHead2(h1);
00296       }
00297       Body* c=ite->second->getChild();
00298       //se ha figli
00299       if (c!=NULL) c->findChildInTree(b1,h1);
00300     }
00301   ite++;
00302   }
00303  }
```

6.18.4.4 findRulesInTree() [1/2]

```
void minerule::Body::findRulesInTree (
            ItemSet * b1,
            ItemSet * b1nb2,
            ItemSet * h1,
            ItemSet * h1nh2 )
```

Definition at line 347 of file IncrAlgoClasses.cpp.

```
00348                                                    {
00349   RowBContainer::iterator ite=NRB->begin();
00350   ItemSet::iterator itfind;
00351   while(ite!=NRB->end()){
00352     //cout«"esamino body "«ite->first«std::endl;
00353     itfind=find(b1nb2->begin(),b1nb2->end(),ite->first);
00354
00355     if(itfind!=b1nb2->end()) {//se l'ho trovato
00356       ite->second->decremSupp();
00357       //cout«"                        decremento body "«ite->first«std::endl;
00358       Head* h=ite->second->getHead();
00359       if (h!=NULL) {
00360         h->findHead2(h1);
00361       }
00362       Body* c=ite->second->getChild();
00363       //se ha figli
00364       if (c!=NULL) {
00365         c->findChildInTree(b1,h1);
00366       }
00367
00368     }else{//non l'ho trovato
00369       itfind=find(b1->begin(),b1->end(),ite->first);
00370       if(itfind!=b1->end()) {//se non soddisfa B1andnotB2 e soddisfa B1 allora soddisfa B2
00371         Head* h=ite->second->getHead();
00372         if (h!=NULL) {
00373           h->findHead2bis(h1nh2,h1);
00374         }
00375         Body* c=ite->second->getChild();
00376         //se ha figli
00377         if (c!=NULL) {
00378
00379           c->findRulesInTree(b1,b1nb2,h1,h1nh2);
00380         }
00381       }
00382
00383     }
00384   ite++;
00385   }
00386 }
```

6.18.4.5 findRulesInTree() [2/2]

```
void minerule::Body::findRulesInTree (
            ItemSet * itemset_body,
            ItemSet * itemset_head )
```

Definition at line 309 of file IncrAlgoClasses.cpp.

```
00309                                                    {
00310
00311
00312    //ItemSet:: iterator itb=body->begin();
00313    //ItemSet:: iterator ity=body->end();
00314
00315   RowBContainer::iterator ite=NRB->begin();
00316   ItemSet::iterator itfind;
00317   while(ite!=NRB->end()){
00318     itfind=find(body->begin(),body->end(),ite->first);
00319     if(itfind!=body->end()) {//se l'ho trovato
00320       ite->second->incremSupp();
00321       //cout«"ho decrem suppb da findRules.."«std::endl;
00322       Head* h=ite->second->getHead();
00323
00324       /*      std::cout « "heads:";
00325       copy( head->begin(),
00326             head->end(),
00327             std::ostream_iterator<ItemType>(cout, " "));
00328             std::cout « "end";*/
00329
00330
00331       //cout«h«std::endl;
00332       if (h!=NULL) {
00333         h->findHead(head);
00334       }
00335       Body* c=ite->second->getChild();
00336       //se ha figli
00337       if (c!=NULL) {
00338         c->findRulesInTree(body,head);
00339       }
```

```
00340     }
00341   ite++;
00342   }
00343  }
```

6.18.4.6 getAncestor()

```
ItemType minerule::Body::getAncestor ( )  [inline]
```

Definition at line 55 of file IncrAlgoClasses.hpp.

```
00055 {return ancestor;}
```

6.18.4.7 insertItemSetB()

```
Head * minerule::Body::insertItemSetB (
            ItemSet & SREV,
            double supp )
```

Definition at line 117 of file IncrAlgoClasses.cpp.

```
00117                                                    {
00118   ItemSet:: iterator ite=SREV.begin();
00119   ItemSet:: iterator itend=SREV.end();
00120   Head* h;
00121   h=insertItemSetB(ite, itend, supp);
00122   return h;
00123  }
```

6.18.4.8 provaStampaLiv1()

```
void minerule::Body::provaStampaLiv1 ( )
```

Definition at line 484 of file IncrAlgoClasses.cpp.

```
00484                              {
00485  RowBContainer::iterator it=NRB->begin();
00486  while(it!=NRB->end()){
00487    std::cout«"elemento "«it->first«" "«std::endl;
00488    std::cout«"child: "«it->second->getChild()«std::endl;
00489    std::cout«"head: "«it->second->getHead()«std::endl;
00490    it++;
00491  }
00492
00493 }
```

6.18.4.9 setAncestor()

```
void minerule::Body::setAncestor (
            ItemType b )  [inline]
```

Definition at line 54 of file IncrAlgoClasses.hpp.

```
00054 {ancestor=b;}
```

6.18.5 Field Documentation

6.18.5.1 countb

```
size_t minerule::Body::countb =0  [static]
```

Definition at line 48 of file IncrAlgoClasses.hpp.

6.18.5.2 NRB

```
RowBContainer* minerule::Body::NRB
```

Definition at line 52 of file IncrAlgoClasses.hpp.

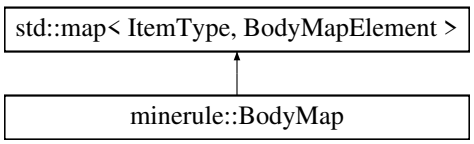
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp
- /Users/esposito/Software/minerule/src/Algorithms/IncrAlgoClasses.cpp

6.19 minerule::BodyMap Class Reference

```
#include <Bodymap.hpp>
```

Inheritance diagram for minerule::BodyMap:



Public Member Functions

- int nextID ()
- BodyMap (std::string f, double totG)
- BodyMap (Connection &cc, double totG)
- void openOutputFiles ()
- void closeOutputFiles ()
- void setTotalGroups (double totG)
- bool checkAntiMono (NewRule &rc)
- bool checkMono (NewRule &rc)
- int add (const int gid, Transaction &t1, Transaction &t2, bool secondPass=false)
- int add (Transaction &t1, const int gid)
- void pruneMap (double threshold, bool onlyBody=false)
- void updateCount ()
- int buildItemset (NewRuleSet &rs, NewRule &rc, std::vector< BodyMap::iterator > &v, int maxCard, float threshold, int j)
- int generateStartItemSets (NewRuleSet &rs, NewRule &rc, std::vector< BodyMap::iterator > &v, int max↩ Card, float threshold, int j)
- int buildRules (NewRuleSet &rs2, NewRule rc, BodyMap::iterator p, float threshold, int maxBody, int max↩ Head)
- int buildHead (NewRuleSet &rs, float threshold, int maxHead, int suppBody, NewRule &rc, std::map< ItemType, MapElement >::iterator j)
- int buildRules (NewRuleSet &rs, NewRule &rc, std::vector< BodyMap::iterator > &vb, std::vector< Body↩ Map::iterator > &vh, int maxBodyCard, int maxHeadCard, float threshold, int j)
- std::vector< BodyMap::iterator > buildHead (NewRuleSet &rs1, NewRule rc, std::vector< BodyMap::iterator > &v, int maxCard, float threshold, int j)
- void howManyItemsets ()
- void saveRules (const NewRuleSet &rs, double bodySupp)
- void saveItemsets (const NewRuleSet &rs, double bodySupp)
- void saveItemset (const NewRule &r, double bodySupp)

Data Fields

- std::vector< AggregateMonoConstraint * > monoConstr
- std::vector< AggregateAntiMonoConstraint * > antiMonoConstr
- K keys

  STL member.
- T elements

  STL member.

6.19.1 Detailed Description

Definition at line 138 of file Bodymap.hpp.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 BodyMap() [1/2]

```
minerule::BodyMap::BodyMap (
            std::string f,
            double totG )  [inline]
```

Definition at line 150 of file Bodymap.hpp.

```
00150 : outfile(f), totGroups(totG) {};
```

6.19.2.2 BodyMap() [2/2]

```
minerule::BodyMap::BodyMap (
            Connection & cc,
            double totG )  [inline]
```

Definition at line 151 of file Bodymap.hpp.

```
00151 : connection(&cc), totGroups(totG) {};
```

6.19.3 Member Function Documentation

6.19.3.1 add() [1/2]

```
int minerule::BodyMap::add (
            const int gid,
            Transaction & t1,
            Transaction & t2,
            bool secondPass = false )
```

Definition at line 117 of file Bodymap.cpp.

```
00117                                                                                   {
00118         int howMany = 0;
00119         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00120                 map<ItemType, BodyMapElement>::iterator found = find(*i);
00121                 if (found == end()) {
00122                         if (secondPass) continue;
00123                         BodyMapElement bme;
00124 //                      bme.insert(gid);
00125                         (*this)[*i] = bme;
00126                         found = find(*i);
00127                 }
00128                 found->second.insert(gid);
00129                 for (Transaction::iterator j = t2.begin(); j != t2.end(); j++)
00130                         if (*i != *j /*&& i->price < j->price*/) {
00131                                 howMany++;
00132                                 found->second.insert(*j,gid,false);
00133 //                              found->second.insert(*j,me,secondPass);
00134                         }
00135         }
00136         return howMany;
00137 }
```

6.19.3.2 add() [2/2]

```
int minerule::BodyMap::add (
            Transaction & t1,
            const int gid )
```

Definition at line 97 of file Bodymap.cpp.

```
00097                                                   {
00098         int howMany = 0;
00099         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00100                 map<ItemType, BodyMapElement>::iterator found = find(*i);
00101                 if (found == end()) {
00102                         BodyMapElement bme;
00103 //                      bme.insert(gid);
00104                         (*this)[*i] = bme;
00105                         found = find(*i);
00106                 }
00107                 found->second.insert(gid);
00108                 int n = t1.values.size()/t1.size();
00109                 for (int k=0; k < n; k++) {
00110                         found->second.setMinMax(k,t1.values[k+howMany*n]);
00111                 }
00112                 howMany++;
00113         }
00114         return howMany;
00115 }
```

6.19.3.3 buildHead() [1/2]

```
int minerule::BodyMap::buildHead (
            NewRuleSet & rs,
            float threshold,
            int maxHead,
            int suppBody,
            NewRule & rc,
            std::map< ItemType, MapElement >::iterator j )
```

Definition at line 396 of file Bodymap.cpp.

```
00396
                                        {
00397         int nrules = 0;
00398         if (rc.head.size() < (size_t)maxHead) {
00399                 map<ItemType, MapElement>::iterator eh = rc.lastBody->second.heads.end();
00400                 while (j != eh) {
00401 //cout « "Extending head of " « *i « " with " « j->first  « " size " « j->second.size() « std::endl;
00402                 if (std::find(rc.body.begin(), rc.body.end(),j->first) == rc.body.end()) {
00403                         GidList newGidList(rc.gids);
00404                         newGidList &= j->second;
00405                                 if (newGidList.moreThan(threshold)) {
00406                                         NewRule r(rc,j,newGidList);
00407                                         rs.insert(rs.end(),r);
00408                                         nrules += 1;
00409                                         nrules += buildHead(rs,threshold,maxHead,suppBody,r,++j);
00410                         } else j++;
00411                 } else j++;
00412         }
00413         }
00414         return nrules;
00415 }
```

6.19.3.4 buildHead() [2/2]

```
std::vector< BodyMap::iterator > minerule::BodyMap::buildHead (
            NewRuleSet & rs1,
            NewRule rc,
            std::vector< BodyMap::iterator > & v,
            int maxCard,
            float threshold,
            int j )
```

Definition at line 262 of file Bodymap.cpp.

```
00262                                                {
00263  std::vector<BodyMap::iterator> newV;
00264 //      NewRuleSet rs;
00265         int start = rs.size();
00266         for (size_t k=j+1; k<v.size(); k++)// {
00267                 //GidList newGidList;
00268         if (std::find(rc.body.begin(), rc.body.end(),v[k]->first) == rc.body.end()) {
00269                 GidList newGidList(v[k]->second);
00270         //      if (rc.gids.count(true) > 0 )
00271                         newGidList &= rc.gids;
00272 //              else newGidList = v[k]->second;
00273                 if (newGidList.moreThan(threshold)) {
00274                         NewRule r(rc,v[k],newV.size(),newGidList);
00275                         rs.insert(rs.end(),r);
00276                         newV.insert(newV.end(),v[k]);
00277                 }
00278         }
00279         int nrules = rs.size();
00280         if (newV.size() > 0 && 1 < maxCard) {
00281                 for (int i=start; i<nrules; i++)
00282                         buildHead(rs,rs[i],newV,maxCard-1,threshold,rs[i].bodySupp);
00283         }
00284         return newV;
00285 }
```

6.19.3.5 buildItemset()

```
int minerule::BodyMap::buildItemset (
            NewRuleSet & rs,
            NewRule & rc,
            std::vector< BodyMap::iterator > & v,
            int maxCard,
            float threshold,
            int j )
```

Definition at line 301 of file Bodymap.cpp.

```
00301                                                {
00302  std::vector<BodyMap::iterator> newV;
00303         NewRuleSet rs;
00304         int start = rs.size();
00305         bool intersect = rc.gids.size() > 0;
00306         int howManyRules = 0;
00307         size_t l = rc.body.size()+1;
00308         for (size_t k=j+1; k<v.size(); k++) {
00309                 //GidList newGidList;
00310                 //GidList newGidList(v[k]->second);
00311                 NewRule r(rc,v[k],v[k]->second,newV.size());
00312                 if (!checkAntiMono(r)) continue;
00313                 r.satisfy = r.satisfy || checkMono(r);
00314                 if (intersect)
00315                         r.gids &= rc.gids;
00316 //              else newGidList = v[k]->second;
00317                 if (r.gids.moreThan(threshold)) {
00318                         //NewRule r(rc,v[k],newGidList,newV.size());
00319                         if(r.satisfy) howManyRules ++;
00320                         rs.insert(rs.end(),r);
00321                         newV.insert(newV.end(),v[k]);
```

```
00322                 }
00323         }
00324         int nrules = rs.size();
00325         while (l >= itemsets.size()) itemsets.insert(itemsets.end(),0);
00326         itemsets[l] += nrules;
00327         saveItemsets(rs, -1);
00328         if (newV.size() > 0 && 1 < maxCard) {
00329                 for (int i=start; i<nrules; i++) {
00330                         //cout « rs[i];
00331                         howManyRules +=
       buildItemset(rs,rs[i],newV,maxCard-1,threshold,rs[i].bodySupp);
00332                 }
00333         }
00334         return howManyRules;
00335 }
```

6.19.3.6 buildRules() [1/2]

```
int minerule::BodyMap::buildRules (
            NewRuleSet & rs,
            NewRule & rc,
            std::vector< BodyMap::iterator > & vb,
            std::vector< BodyMap::iterator > & vh,
            int maxBodyCard,
            int maxHeadCard,
            float threshold,
            int j )
```

Definition at line 229 of file Bodymap.cpp.

```
00229                                                                                {
00230  std::vector<BodyMap::iterator> newVb, newVh;
00231         NewRuleSet rs;
00232         int start = rs.size();
00233         bool intersect = (rc.gids.size() > 0 );
00234         for (size_t k=j+1; k<vb.size(); k++) {
00235                 //GidList newGidList;
00236                 GidList newGidList(vb[k]->second);
00237                 if (intersect)
00238                         newGidList &= rc.gids;
00239 //              else newGidList = v[k]->second;
00240                 if (newGidList.moreThan(threshold)) {
00241                         NewRule r(rc,vb[k],newGidList,newVb.size());
00242                         rs.insert(rs.end(),r);
00243                         newVb.insert(newVb.end(),vb[k]);
00244                 }
00245         }
00246         int nrules = rs.size();
00247         int howManyRules = 0;
00248         for (int i=start; i<nrules; i++) {
00249             if (i<nrules) {
00250                 NewRuleSet rs2;
00251 //              curBodySupp = rs[i].gids.count(true);
00252                 newVh = buildHead(rs2,rs[i],vh,maxHeadCard,threshold,-1);
00253                 howManyRules += rs2.size();
00254                 saveRules(rs2, rs[i].gids.count());
00255              }
00256                 if (newVb.size() > 0 && maxBodyCard > 1) {
00257                         howManyRules += buildRules(rs1,
       rs[i],newVb,newVh,maxBodyCard-1,maxHeadCard,threshold,rs[i].bodySupp);
00258                 }
00259         }
00260         return howManyRules;
00261 }
```

6.19.3.7 buildRules() [2/2]

```
int minerule::BodyMap::buildRules (
            NewRuleSet & rs2,
            NewRule rc,
            BodyMap::iterator p,
            float threshold,
            int maxBody,
            int maxHead )
```

Definition at line 375 of file Bodymap.cpp.

```
00375
                {
00376         int howManyRules = 0;
00377
00378         bool intersect = rc.gids.size() > 0;
00379         while (p != end()) {
00380                 GidList newGidList(p->second);
00381                 if (intersect)
00382                         newGidList &= rc.gids;
00383                 //else newGidList = p->second;
00384                 if (newGidList.moreThan(threshold)) {
00385                         NewRule r(rc,p,newGidList);
00386                         howManyRules += buildHead(rs2, threshold, maxHead, r.gids.count(true), r,
      p->second.heads.begin());
00387                         p++;
00388                 if (r.body.size() < (size_t)maxBody) {
00389                         howManyRules += buildRules(rs2, r, p, threshold, maxBody, maxHead);
00390                 }
00391                 } else p++;
00392         }
00393         return howManyRules;
00394 }
```

6.19.3.8 checkAntiMono()

```
bool minerule::BodyMap::checkAntiMono (
            NewRule & rc )
```

Definition at line 287 of file Bodymap.cpp.

```
00287                                       {
00288         bool flag = true;
00289         for (size_t i=0; i<antiMonoConstr.size() && flag; i++)
00290                 flag = flag && antiMonoConstr[i]->check(*this,rc.body);
00291         return flag;
00292 }
```

6.19.3.9 checkMono()

```
bool minerule::BodyMap::checkMono (
            NewRule & rc )
```

Definition at line 294 of file Bodymap.cpp.

```
00294                                     {
00295         bool flag = true;
00296         for (size_t i=0; i<monoConstr.size() && flag; i++)
00297                 flag = flag && monoConstr[i]->check(*this,rc.body);
00298         return flag;
00299 }
```

6.19.3.10 closeOutputFiles()

```
void minerule::BodyMap::closeOutputFiles ( )  [inline]
```

Definition at line 158 of file Bodymap.hpp.

```
00158 { outR.close(); outHB.close(); }
```

6.19.3.11 generateStartItemSets()

```
int minerule::BodyMap::generateStartItemSets (
            NewRuleSet & rs,
            NewRule & rc,
            std::vector< BodyMap::iterator > & v,
            int maxCard,
            float threshold,
            int j )
```

Definition at line 337 of file Bodymap.cpp.

```
00337
                                {
00338  std::vector<BodyMap::iterator> newV;
00339         NewRuleSet rs;
00340         //      int start = rs.size();
00341         bool intersect = rc.gids.size() > 0;
00342         int howManyRules = 0;
00343         size_t l = rc.body.size()+1;
00344         while (l >= itemsets.size()) itemsets.insert(itemsets.end(),0);
00345         size_t k=j+1;
00346         if (k<v.size()) {
00347                 //GidList newGidList;
00348                 //GidList newGidList(v[k]->second);
00349                 NewRule r(rc,v[k],v[k]->second,k);
00350                 if (checkAntiMono(r)) {
00351                 r.satisfy = r.satisfy || checkMono(r);
00352                 if (intersect)
00353                         r.gids &= rc.gids;
00354 //              else newGidList = v[k]->second;
00355                 if (r.gids.moreThan(threshold)) {
00356                         //NewRule r(rc,v[k],newGidList,newV.size());
00357                         if(r.satisfy) {
00358                         //      rs.insert(rs.end(),r);
00359                                 //cout << r;
00360                                 saveItemset(r, -1);
00361                                 howManyRules += buildItemset(rs,r,v,maxCard-1,threshold,r.bodySupp) +
      1;
00362                                 itemsets[l] += 1;
00363                                 //howManyRules += generateStartItemSets(rs,rc,v,maxCard,threshold,k);
00364                         } else {
00365                                 howManyRules += generateStartItemSets(rs,r,v,maxCard-1,threshold,k);
00366                                 if (howManyRules == 0) return howManyRules;
00367                         }
00368                 }
00369                 }
00370                 howManyRules += generateStartItemSets(rs,rc,v,maxCard,threshold,k);
00371         }
00372         return howManyRules;
00373 }
```

6.19.3.12 howManyItemsets()

```
void minerule::BodyMap::howManyItemsets ( )
```

Definition at line 417 of file Bodymap.cpp.

```
00417                           {
00418         for (size_t i = 1; i < itemsets.size(); i++) {
00419                 std::cout << "Itemset Length: " << i << " -> " << itemsets[i] << std::endl;
00420         }
00421 }
```

6.19.3.13 nextID()

```
int minerule::BodyMap::nextID ( )  [inline]
```

Definition at line 149 of file Bodymap.hpp.

```
00149 { return nextid++; }
```

6.19.3.14 openOutputFiles()

```
void minerule::BodyMap::openOutputFiles ( )  [inline]
```

Definition at line 152 of file Bodymap.hpp.

```
00152                                   {
00153                         std::string filename = outfile + ".r";
00154                         outR.open(filename.c_str());
00155                         std::string filename1 = outfile + ".hb";
00156                         outHB.open(filename1.c_str());
00157                 }
```

6.19.3.15 pruneMap()

```
void minerule::BodyMap::pruneMap (
            double threshold,
            bool onlyBody = false )
```

Definition at line 214 of file Bodymap.cpp.

```
00214                                                   {
00215         //BodyMap newMap(outfile, totGroups);
00216         for (iterator i = begin(); i != end(); i++)
00217                 if (!i->second.pruneMap(threshold,onlyBody)) {
00218                 //       newMap[i->first] = i->second;
00219                         i->second.done = true;
00220                 } else i->second.done = false;
00221         //(*this) = newMap;
00222 }
```

6.19.3.16 saveItemset()

```
void minerule::BodyMap::saveItemset (
            const NewRule & r,
            double bodySupp )
```

Definition at line 423 of file Bodymap.cpp.

```
00424                                           {
00425     int c = r.gids.count();
00426     connection->insert( r.body,
00427                         r.head,
00428                         c/totGroups,
00429                         1, true );
00430 }
```

6.19.3.17 saveItemsets()

```
void minerule::BodyMap::saveItemsets (
            const NewRuleSet & rs,
            double bodySupp )
```

Definition at line 448 of file Bodymap.cpp.

```
00449                                                {
00450 //    static ofstream out(outfile.c_str());
00451     NewRuleSet::const_iterator it = rs.begin();
00452     int c;
00453     for(it=rs.begin(); it!=rs.end(); it++) {
00454         c = it->gids.count();
00455         connection->insert( it->body,
00456                             it->head,
00457                             c/totGroups,
00458                             1, true );
00459     }
00460   }
```

6.19.3.18 saveRules()

```
void minerule::BodyMap::saveRules (
            const NewRuleSet & rs,
            double bodySupp )
```

Definition at line 432 of file Bodymap.cpp.

```
00433                                                {
00434     NewRuleSet::const_iterator it = rs.begin();
00435     bool bodyID = true;
00436     int c;
00437     for(it=rs.begin(); it!=rs.end(); it++) {
00438       c = it->gids.count();
00439       connection->insert( it->body,
00440                           it->head,
00441                           c/totGroups,
00442                           c/bodySupp, bodyID );
00443 //      bodyID = false;
00444     }
00445   }
```

6.19.3.19 setTotalGroups()

```
void minerule::BodyMap::setTotalGroups (
            double totG )  [inline]
```

Definition at line 159 of file Bodymap.hpp.

```
00159 { totGroups = totG; }
```

6.19.3.20 updateCount()

```
void minerule::BodyMap::updateCount ( )
```

Definition at line 224 of file Bodymap.cpp.

```
00224                          {
00225         for (iterator i = begin(); i != end(); i++)
00226                 i->second.updateCount();
00227 }
```

6.19.4 Field Documentation

6.19.4.1 antiMonoConstr

```
std::vector<AggregateAntiMonoConstraint *> minerule::BodyMap::antiMonoConstr
```

Definition at line 161 of file Bodymap.hpp.

6.19.4.2 elements

```
T std::map< K, T >::elements  [inherited]
```

STL member.

6.19.4.3 keys

```
K std::map< K, T >::keys  [inherited]
```

STL member.

6.19.4.4 monoConstr

```
std::vector<AggregateMonoConstraint *> minerule::BodyMap::monoConstr
```

Definition at line 160 of file Bodymap.hpp.

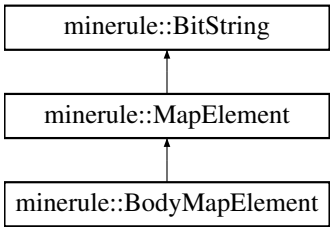
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp
- /Users/esposito/Software/minerule/src/Algorithms/Bodymap.cpp

6.20 minerule::BodyMapElement Class Reference

```
#include <Bodymap.hpp>
```

Inheritance diagram for minerule::BodyMapElement:



Public Member Functions

- BodyMapElement ()
- BodyMapElement (const BodyMapElement &bm)
- void insert (const int x)
- void insert (const ItemType &item, const int gid, bool secondPass=false)
- bool pruneMap (double threshold, bool onlyBody=false)
- bool updateCount ()
- void setMinMax (int which, int v)
- void insert (Bits element)
- boolean test (int i) const
- BitString & set ()
- BitString & set (int i, boolean value=true)
- BitString & reset ()
- BitString & reset (int i)
- BitString & clear ()
- BitString & clear (int i)
- BitString & invert ()
- BitString & invert (int i)
- BitString & operator&= (const BitString &bs)
- BitString & operator|= (const BitString &bs)
- BitString & operator^= (const BitString &bs)
- boolean operator== (const BitString &bs)
- boolean operator!= (const BitString &bs)
- BitString operator& (const BitString &bs1)
- int count (boolean what=true) const
- bool moreThan (double threshold) const
- int length () const
- int ssize () const
- int size () const
- void serialize (char *serialized, int *start)
- void unserialize (char *serialized, int *start)
- void print ()
- void print (std::ostream &out)
- void setNBit (int num)
- void * getElmA (int which)
- Bits getElm (int which)
- std::vector< Bits > & gbits ()
- char operator[ ] (int which) const

Data Fields

- std::vector< MinMax > attribute
- bool done
- std::map< ItemType, MapElement > heads
- int counter

Static Public Attributes

- static int intersections = 0

6.20.1 Detailed Description

Definition at line 69 of file Bodymap.hpp.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 BodyMapElement() [1/2]

```
minerule::BodyMapElement::BodyMapElement ( )  [inline]
```

Definition at line 73 of file Bodymap.hpp.

```
00073 { done = false; }
```

6.20.2.2 BodyMapElement() [2/2]

```
minerule::BodyMapElement::BodyMapElement (
            const BodyMapElement & bm )  [inline]
```

Definition at line 74 of file Bodymap.hpp.

```
00074                                                          {
00075                         done = bm.done;
00076                         for (size_t i=0; i<bm.attribute.size(); i++)
      attribute.insert(attribute.end(),bm.attribute[i]);
00077                         BitString::operator=((BitString&)bm);
00078                 }
```

6.20.3 Member Function Documentation

6.20.3.1 clear() [1/2]

```
BitString & minerule::BitString::clear ( )  [inherited]
```

Definition at line 121 of file Bitstring.cpp.

```
00122 {
00123 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00124 return *this;
00125 }
```

6.20.3.2 clear() [2/2]

```
BitString & minerule::BitString::clear (
            int i )  [inherited]
```

Definition at line 131 of file Bitstring.cpp.

```
00132 {
00133 return set(i, false);
00134 }
```

6.20.3.3 count()

```
int minerule::BitString::count (
            boolean what = true ) const  [inherited]
```

Definition at line 284 of file Bitstring.cpp.

```
00285 {
00286 int i,j, k=0;
00287 for (i=0; i<=Nw; i++) {
00288 //      Bits b = bits[i];
00289 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b>>=1;}
00290         for (j=0; j<sizeof(Bits); j++)
00291         k += NBITS[*(((unsigned char*)&bits[i])+j)];
00292 }
00293 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00294 return k;
00295 }
```

6.20.3.4 gbits()

```
std::vector< Bits > & minerule::BitString::gbits ( )  [inline], [inherited]
```

Definition at line 101 of file Bitstring.hpp.

```
00101 {  return bits; }
```

6.20.3.5 getElm()

```
Bits minerule::BitString::getElm (
            int which )  [inline], [inherited]
```

Definition at line 99 of file Bitstring.hpp.

```
00099 {  return bits[which]; }
```

6.20.3.6 getElmA()

```
void * minerule::BitString::getElmA (
            int which )  [inline], [inherited]
```

Definition at line 98 of file Bitstring.hpp.

```
00098 {  return &(bits[which]); }
```

6.20.3.7 insert() [1/3]

```
void minerule::BitString::insert (
            Bits element )  [inline], [inherited]
```

Definition at line 100 of file Bitstring.hpp.

```
00100 {  bits.push_back(element); }
```

6.20.3.8 insert() [2/3]

```
void minerule::BodyMapElement::insert (
            const int x )  [inline]
```

Definition at line 80 of file Bodymap.hpp.

```
00080 { set(x,true); }
```

6.20.3.9 insert() [3/3]

```
void minerule::BodyMapElement::insert (
            const ItemType & item,
            const int gid,
            bool secondPass = false )
```

Definition at line 26 of file Bodymap.cpp.

```
00026                                                                                {
00027         std::map<ItemType, MapElement>::iterator found = heads.find(item);
00028         if (found == heads.end()) {
00029                 if (!secondPass) heads[item].insert(gid);
00030         } else found->second.insert(gid);
00031 }
```

6.20.3.10 invert() [1/2]

```
BitString & minerule::BitString::invert ( )  [inherited]
```

Definition at line 140 of file Bitstring.cpp.

```
00141 {
00142 for (int i=0; i<=Nw; i++) bits[i] = ~bits[i];
00143 return *this;
00144 }
```

6.20.3.11 invert() [2/2]

```
BitString & minerule::BitString::invert (
            int i )  [inherited]
```

Definition at line 150 of file Bitstring.cpp.

```
00151 {
00152 if (n <= i) _Xran(i);
00153 bits[i/Nb] ^= (Bits)1 « i%Nb;
00154 return *this;
00155 }
```

6.20.3.12 length()

```
int minerule::BitString::length ( ) const  [inline], [inherited]
```

Definition at line 87 of file Bitstring.hpp.

```
00087 { return n; }
```

6.20.3.13 moreThan()

```
bool minerule::BitString::moreThan (
            double threshold ) const  [inherited]
```

Definition at line 301 of file Bitstring.cpp.

```
00302 {
00303 int i,j, k=(int)(threshold);
00304 for (i=0; i<=Nw && k >= 0; i++) {
00305 //      Bits b = bits[i];
00306 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b>>=1;}
00307         for (j=0; j<sizeof(Bits); j++)
00308         k -= NBITS[*(((unsigned char*)&bits[i])+j)];
00309 }
00310 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00311 return k<0;
00312 }
```

6.20.3.14 operator"!=()

```
boolean minerule::BitString::operator!= (
            const BitString & bs )  [inherited]
```

Definition at line 263 of file Bitstring.cpp.

```
00264 {
00265 return !operator==(bs);
00266 }
```

6.20.3.15 operator&()

```
BitString minerule::BitString::operator& (
            const BitString & bs1 )  [inherited]
```

Definition at line 223 of file Bitstring.cpp.

```
00224 {
00225         BitString bs;
00226         int min = Nw < bs1.Nw ? Nw : bs1.Nw;
00227         for (int i=0; i<=min; i++) bs.bits.insert(bs.bits.end(),bits[i] & bs1.bits[i]);
00228         bs.Nw = min;
00229         bs.n = n < bs1.n ? n : bs1.n;
00230         return bs;
00231 }
```

6.20.3.16 operator&=()

```
BitString & minerule::BitString::operator&= (
            const BitString & bs )  [inherited]
```

Definition at line 195 of file Bitstring.cpp.

```
00196 {
00197 /*
00198 if (n < bs.n) {
00199         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n);
00200         n = bs.n;
00201 }
00202 if (Nw < bs.Nw) {
00203         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), ~(char)0); }
00204         n = bs.n;
00205 }
00206 for (int i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00207 */
00208 if (Nw < bs.Nw) {
00209         for (int i=0; i<=Nw; i++) bits[i] &= bs.bits[i];
00210 } else {
00211         int i;
00212         for (i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00213         for (; i<=Nw; i++) bits[i] = (Bits)0;
00214 }
00215 intersections++;
00216 return *this;
00217 }
```

6.20.3.17 operator==()

```
boolean minerule::BitString::operator== (
            const BitString & bs )  [inherited]
```

Definition at line 251 of file Bitstring.cpp.

```
00252 {
00253 boolean eq = n == bs.n;
00254 for (int i=0; i<Nw && eq; i++) eq = bits[i] == bs.bits[i];
00255 for (int i=(Nw-1)*Nb; i<n && eq; i++) eq = test(i) == bs.test(i);
00256 return eq;
00257 }
```

6.20.3.18 operator[]()

```
char minerule::BitString::operator[] (
            int which ) const  [inline], [inherited]
```

Definition at line 103 of file Bitstring.hpp.

```
00103 { return (test(which) ? '1' : '0'); }
```

6.20.3.19 operator^=()

```
BitString & minerule::BitString::operator^= (
            const BitString & bs )  [inherited]
```

Definition at line 178 of file Bitstring.cpp.

```
00179 {
00180 if (n < bs.n) {
00181         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n,bs.test(n));
00182         n = bs.n;
00183 }
00184 if (Nw < bs.Nw) {
00185         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), bs.bits[Nw]); }
00186 }
00187 for (int i=0; i<=bs.Nw; i++) bits[i] ^= bs.bits[i];
00188 return *this;
00189 }
```

6.20.3.20 operator"|=()

```
BitString & minerule::BitString::operator|= (
            const BitString & bs )  [inherited]
```

Definition at line 161 of file Bitstring.cpp.

```
00162 {
00163 if (n < bs.n) {
00164         for (; n < (Nw+1)*Nb && n < bs.n; n++) reset(n);
00165         n = bs.n;
00166 }
00167 if (Nw < bs.Nw) {
00168         for (; Nw <= bs.Nw; ) { Nw++; bits.insert(bits.end(), (Bits)0); }
00169 }
00170 for (int i=0; i<=bs.Nw; i++) bits[i] |= bs.bits[i];
00171 return *this;
00172 }
```

6.20.3.21 print() [1/2]

```
void minerule::BitString::print ( )  [inherited]
```

Definition at line 404 of file Bitstring.cpp.

```
00405 {int i,dim;
00406
00407  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00408  {
00409  dim=(*this).length();
00410  for (i=0;i<dim;i++)
00411  {
00412   if ((i%8==0)&&(i!=0)) std::cout<<"-";
00413   std::cout<<test(i);
00414  }
00415  }
00416 }
```

6.20.3.22 print() [2/2]

```
void minerule::BitString::print (
            std::ostream & out )  [inherited]
```

Definition at line 390 of file Bitstring.cpp.

```
00391 {int i,dim;
00392
00393  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00394  {
00395  dim=(*this).length();
00396  for (i=0;i<dim;i++)
00397  {
00398   if ((i%8==0)&&(i!=0)) out«"-";
00399   out«test(i);
00400  }
00401  }
00402 }
```

6.20.3.23 pruneMap()

```
bool minerule::BodyMapElement::pruneMap (
            double threshold,
            bool onlyBody = false )
```

Definition at line 33 of file Bodymap.cpp.

```
00033                                                                    {
00034         if (!moreThan(threshold)) return false;
00035         else if (onlyBody) return true;
00036         std::map<ItemType, MapElement> newMap;
00037         for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++)
00038                 if (i->second.counter >= threshold) {i->second.counter = 0; newMap[i->first] =
      i->second;}
00039         heads = newMap;
00040         return heads.size() > 0;
00041 }
```

6.20.3.24 reset() [1/2]

```
BitString & minerule::BitString::reset ( )  [inherited]
```

Definition at line 102 of file Bitstring.cpp.

```
00103 {
00104 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00105 return *this;
00106 }
```

6.20.3.25 reset() [2/2]

```
BitString & minerule::BitString::reset (
            int i )  [inherited]
```

Definition at line 112 of file Bitstring.cpp.

```
00113 {
00114 return set(i, false);
00115 }
```

6.20.3.26 serialize()

```
void minerule::BitString::serialize (
            char * serialized,
            int * start )  [inherited]
```

Definition at line 352 of file Bitstring.cpp.

```
00353 {int i;
00354
00355  memcpy(&(serialized[(*start)]),&Nw,sizeof(int));
00356         //cout«int(serialized[(*start)])«" "«std::endl;
00357  //cout«"\"";
00358  (*start) += sizeof(int);
00359  memcpy(&(serialized[(*start)]),&n,sizeof(int));
00360  (*start) += sizeof(int);
00361         for (i=0;i<=Nw;i++)
00362         {
00363          memcpy(&(serialized[(*start)]),&(bits[i]),sizeof(Bits));
00364          //cout«(int)(serialized[(*start)])«" ";
00365                 (*start) += sizeof(Bits);
00366         }
00367 }
```

6.20.3.27 set() [1/2]

```
BitString & minerule::BitString::set ( )  [inherited]
```

Definition at line 80 of file Bitstring.cpp.

```
00081 {
00082 for (int i=0; i<=Nw; i++) bits[i] = ~(Bits)0;
00083 return *this;
00084 }
```

6.20.3.28 set() [2/2]

```
BitString & minerule::BitString::set (
            int i,
            boolean value = true )  [inherited]
```

Definition at line 90 of file Bitstring.cpp.

```
00091 {
00092 if (n <= i) _Xran(i);
00093 if (value) bits[i/Nb] |= (Bits)1 « i%Nb;
00094 else bits[i/Nb] &= ~((Bits)1 « i%Nb);
00095 return *this;
00096 }
```

6.20.3.29 setMinMax()

```
void minerule::BodyMapElement::setMinMax (
            int which,
            int v )
```

Definition at line 88 of file Bodymap.cpp.

```
00088                                                 {
00089         MinMax mm;
00090         for (int i=attribute.size(); i<=which; i++) {
00091                 attribute.insert(attribute.end(),mm);
00092         }
00093         if (attribute[which].minValue() > v) attribute[which].min = v;
00094         if (attribute[which].maxValue() < v) attribute[which].max = v;
00095 }
```

6.20.3.30 setNBit()

```
void minerule::BitString::setNBit (
            int num )  [inline], [inherited]
```

Definition at line 97 of file Bitstring.hpp.

```
00097 {  Nw = (num == 0) ? 0 : (num-1) / Nb;n=num; }
```

6.20.3.31 size()

```
int minerule::BitString::size ( ) const  [inline], [inherited]
```

Definition at line 89 of file Bitstring.hpp.

```
00089 { return n; }
```

6.20.3.32 ssize()

```
int minerule::BitString::ssize ( ) const  [inline], [inherited]
```

Definition at line 88 of file Bitstring.hpp.

```
00088 { return bits.size(); }
```

6.20.3.33 test()

```
boolean minerule::BitString::test (
            int i ) const  [inline], [inherited]
```

Definition at line 58 of file Bitstring.hpp.

```
00058                                   {
00059                         if (i<0 || n <= i) return false;
00060                         return ((bits[i/Nb] & ((Bits)1 << i%Nb)) != 0);
00061                 }
```

6.20.3.34 unserialize()

```
void minerule::BitString::unserialize (
            char * serialized,
            int * start )  [inherited]
```

Definition at line 369 of file Bitstring.cpp.

```
00370 {int i;
00371  char tmpchar;
00372
00373  memcpy(&Nw,&(serialized[(*start)]),sizeof(int));
00374         //cout<<int(serialized[(*start)])<<" "<<std::endl;
00375  //cout<<"\"";
00376  (*start) += sizeof(int);
00377  memcpy(&n,&(serialized[(*start)]),sizeof(int));
00378  (*start) += sizeof(int);
00379  for (i=bits.size();i<=Nw;i++) bits.insert(bits.end(),(Bits)0);
00380         for (i=0;i<=Nw;i++)
00381         {
00382          memcpy(&tmpchar,&serialized[(*start)],sizeof(Bits));
00383          //cout<<(int)(serialized[(*start)])<<" ";
00384          bits[i] = tmpchar;
00385          (*start) += sizeof(Bits);
00386         }
00387 // setNBit(numbit);
00388 }
```

6.20.3.35 updateCount()

```
bool minerule::BodyMapElement::updateCount ( )
```

Definition at line 43 of file Bodymap.cpp.

```
00043                                   {
00044 //      if (size() < threshold) return false;
00045 //      map<ItemType, MapElement> newMap;
00046 //      for (map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++)
00047 //{
00048 //              i->second.counter += i->second.count(true);
00049 //}
00050 //              if (i->second.size() >= threshold) newMap[i->first] = i->second;
00051 //      heads = newMap;
00052         counter += count(true);
00053         return heads.size() > 0;
00054 }
```

6.20.4 Field Documentation

6.20.4.1 attribute

```
std::vector<MinMax> minerule::BodyMapElement::attribute
```

Definition at line 71 of file Bodymap.hpp.

6.20.4.2 counter

```
int minerule::MapElement::counter  [inherited]
```

Definition at line 51 of file Bodymap.hpp.

6.20.4.3 done

```
bool minerule::BodyMapElement::done
```

Definition at line 72 of file Bodymap.hpp.

6.20.4.4 heads

```
std::map<ItemType, MapElement > minerule::BodyMapElement::heads
```

Definition at line 81 of file Bodymap.hpp.

6.20.4.5 intersections

```
int minerule::BitString::intersections = 0  [static], [inherited]
```

Definition at line 55 of file Bitstring.hpp.

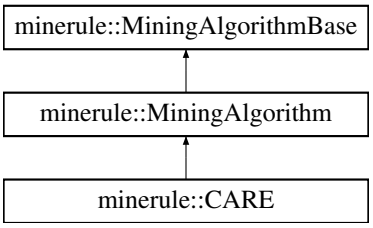
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp
- /Users/esposito/Software/minerule/src/Algorithms/Bodymap.cpp

6.21 minerule::CARE Class Reference

```
#include <Care.hpp>
```

Inheritance diagram for minerule::CARE:



Public Member Functions

- CARE (const OptimizedMinerule &mr)
- virtual ~CARE ()
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual void initialize ()
- virtual void execute ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- AlgorithmsOptions options
- Connection connection
- const OptimizedMinerule & minerule

6.21.1 Detailed Description

Definition at line 25 of file Care.hpp.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 CARE()

```
minerule::CARE::CARE (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 44 of file Care.hpp.

```
00044 : MiningAlgorithm(mr) {}
```

6.21.2.2 ~CARE()

```
virtual minerule::CARE::~CARE ( )  [inline], [virtual]
```

Definition at line 46 of file Care.hpp.

```
00046 {}
```

6.21.3 Member Function Documentation

6.21.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.21.3.2 canHandleMinerule()

```
virtual bool minerule::CARE::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 50 of file Care.hpp.

```
00050                                          {
00051       return
00052         !minerule.getParsedMinerule().hasCrossConditions() &&
00053         !minerule.getParsedMinerule().requiresClusters() &&
00054         !minerule.getParsedMinerule().hasDisjuctionsInMC();
00055     }
```

6.21.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                       {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.21.3.4 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                          {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.21.3.5 mineRules()

```
void minerule::CARE::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 67 of file Care.cpp.

```
00067                                   {
00068                 MRLogPush("Starting CARE mining algorithm...");
00069
00070                 MRLog() « "Preparing data sources..." « std::endl;
00071                 prepareData();
00072
00073                 double support = options.getSupport();
00074                 int maxBody = options.getBodyCardinalities().getMax();
00075                 int maxHead = options.getHeadCardinalities().getMax();
00076
00077                 ItemType gid1;
00078
00079                 BodyMap bodyMap(connection,1);
00080                 BodyMap headMap(connection,1);
00081
00082                 int howManyGroups = 0;
00083                 int totalGroups = sourceTable->getTotGroups();
00084                 int howManyRows = 0;
00085
00086                 while (!bodyIterator.isAfterLast()) {
00087                         ItemType gid = bodyIterator->getGroup();
00088                         Transaction t1, t2;
00089
00090                         t1.loadBody(gid,bodyIterator);
00091                         bool found2 = t2.findGid(gid,headIterator);
00092                         if (found2) {
00093                                 t2.loadHead(gid,headIterator);
00094                         }
00095                         howManyRows += bodyMap.add(t1,howManyGroups);
00096                         howManyRows += headMap.add(t2,howManyGroups);
00097                         howManyGroups++;
00098                 }
00099
00100                MRLog() « "Total rows: " « howManyRows « std::endl;
00101                MRLog() « "Total groups: " « totalGroups « std::endl;
00102                MRLogPop();
00103
00104                MRLogPush("Starting rule extraction...");
00105
00106                bodyMap.updateCount();
00107                headMap.updateCount();
00108                MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00109   //  MRLogPop();
00110 //    MRLogPush("Starting pruning ...");
00111                bodyMap.pruneMap(support*totalGroups,true);
00112                headMap.pruneMap(support*totalGroups,true);
00113 //    MRLogPop();
00114                bodyMap.setTotalGroups(totalGroups);
00115 //  MRLogPush("Starting rule extraction ...");
00116                std::vector<BodyMap::iterator> v;
00117                std::vector<BodyMap::iterator> v1;
00118                std::multimap<int, BodyMap::iterator> temp;
00119                for (BodyMap::iterator i = bodyMap.begin(); i!=bodyMap.end(); i++)
00120                        if (!i->second.done) temp.insert(std::pair<int,
       BodyMap::iterator>(i->second.count(),i));
00121                for (std::multimap<int, BodyMap::iterator>::iterator i = temp.begin(); i!=temp.end();
       i++)
00122                        v.insert(v.end(),i->second);
00123                for (BodyMap::iterator i = headMap.begin(); i!=headMap.end(); i++)
00124                        v1.insert(v1.end(),i);
00125                MRLog() « "Total bodies after pruning: " « v.size() « std::endl;
00126
00127                NewRule r;
00128                NewRuleSet rs1;
00129    //bodyMap.openOutputFiles();
00130                int nrules = bodyMap.buildRules(rs1,r,v,v1,maxBody,maxHead,support*totalGroups,-1);
00131    //bodyMap.closeOutputFiles();
00132                MRLog() « "After extracting rules: " « nrules « std::endl;
00133                MRLogPop();
00134                connection.finalize();
00135        }
```

6.21.3.6 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.21.3.7 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::MiningAlgorithmBase::sourceTableRequirements ( )
const  [inline], [virtual], [inherited]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CCSMiner, minerule::FSMiner, and minerule::STSMiner.

Definition at line 43 of file MiningAlgorithmBase.hpp.

```
00043                                                                    {
00044                         return SourceTableRequirements();
00045                 };
```

6.21.4 Field Documentation

6.21.4.1 connection

```
Connection minerule::MiningAlgorithm::connection  [protected], [inherited]
```

Definition at line 62 of file MiningAlgorithmBase.hpp.

6.21.4.2 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

6.21.4.3 options

```
AlgorithmsOptions minerule::MiningAlgorithm::options  [protected], [inherited]
```

Definition at line 61 of file MiningAlgorithmBase.hpp.

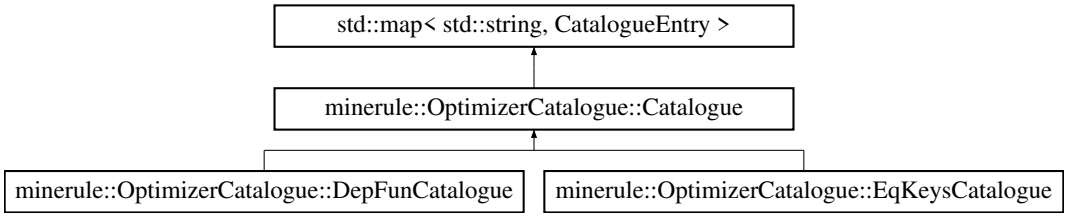
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Care.hpp
- /Users/esposito/Software/minerule/src/Algorithms/Care.cpp

6.22 minerule::OptimizerCatalogue::Catalogue Class Reference

`#include <OptimizerCatalogue.hpp>`

Inheritance diagram for minerule::OptimizerCatalogue::Catalogue:



Public Member Functions

- Catalogue ()
- Catalogue (const Catalogue &catalogue)
- virtual ~Catalogue ()
- void insertMapping (const std::string &table, const KeyCols &origKeyCols, int refKeyId, OrderType orderType)
- void initialize ()
- virtual const std::string & getSchemaInfo (const std::string &schemaKey) const =0

Static Public Member Functions

- static OrderType stringToOrder (const std::string &)

Data Fields

- K keys
  STL member.
- T elements
  STL member.

6.22.1 Detailed Description

Definition at line 78 of file OptimizerCatalogue.hpp.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Catalogue() [1/2]

`minerule::OptimizerCatalogue::Catalogue::Catalogue ( )  [inline]`

Definition at line 80 of file OptimizerCatalogue.hpp.

```
00080 : std::map<std::string, CatalogueEntry>() {}
```

6.22.2.2 Catalogue() [2/2]

```
minerule::OptimizerCatalogue::Catalogue::Catalogue (
            const Catalogue & catalogue )  [inline]
```

Definition at line 81 of file OptimizerCatalogue.hpp.

```
00081                                           :
00082               map<std::string, CatalogueEntry>(catalogue) {}
```

6.22.2.3 ~Catalogue()

```
virtual minerule::OptimizerCatalogue::Catalogue::~Catalogue ( )  [inline], [virtual]
```

Definition at line 83 of file OptimizerCatalogue.hpp.

```
00083 {}
```

6.22.3 Member Function Documentation

6.22.3.1 getSchemaInfo()

```
virtual const std::string & minerule::OptimizerCatalogue::Catalogue::getSchemaInfo (
            const std::string & schemaKey ) const  [pure virtual]
```

Implemented in minerule::OptimizerCatalogue::EqKeysCatalogue, and minerule::OptimizerCatalogue::DepFunCatalogue.

6.22.3.2 initialize()

```
void minerule::OptimizerCatalogue::Catalogue::initialize ( )
```

Definition at line 153 of file OptimizerCatalogue.cpp.

```
00153                                           {
00154   mrdb::Connection *connection =
00155       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00156
00157   try {
00158     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00159     std::string getCatalogueQuery =
00160         (std::string) "SELECT A." + getSchemaInfo("tab_main_tab_name") + ",B." +
00161         getSchemaInfo("tab_cols_col_name") + ",A." +
00162         getSchemaInfo("tab_main_rhs_key_id") + ",A." +
00163         getSchemaInfo("tab_main_order_type") + " " + "FROM " +
00164         getSchemaInfo("tab_main") + " AS A," + getSchemaInfo("tab_cols") +
00165         " AS B " + "WHERE A." + getSchemaInfo("tab_main_lhs_key_id") + "=B." +
00166         getSchemaInfo("tab_cols_key_id");
00167
00168     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getCatalogueQuery));
00169     KeyCols origKey;
00170     unsigned long refKeyId = 0;
00171     std::string curTable;
00172     OrderType orderType;
00173
00174     // We start building the first of the two col l lists.
00175     // we accumulate the result in the set origKey.
00176
```

```
00177     if (rs->next()) {
00178       curTable = rs->getString(1);
00179       origKey.insert(rs->getString(2));
00180       refKeyId = rs->getInt(3);
00181       orderType = stringToOrder(rs->getString(4));
00182     } else {
00183       return;
00184     }
00185 
00186     while (rs->next()) {
00187       if (curTable != rs->getString(1) ||
00188           refKeyId != (unsigned long)rs->getInt(3)) {
00189         // here the current rule changed, in fact either the table
00190         // or the refKeyId are different
00191         insertMapping(curTable, origKey, refKeyId, orderType);
00192         origKey.clear();
00193 
00194         curTable = rs->getString(1);
00195         origKey.insert(rs->getString(2));
00196         refKeyId = rs->getInt(3);
00197         orderType = stringToOrder(rs->getString(4));
00198       } else {
00199         // here we continue to accumulate the columns in origKey
00200         origKey.insert(rs->getString(2));
00201       }
00202     } // end while
00203 
00204     assert(!origKey.empty());
00205     insertMapping(curTable, origKey, refKeyId, orderType);
00206   } catch (mrdb::SQLException &e) {
00207     throw MineruleException(
00208         MR_ERROR_DATABASE_ERROR,
00209         std::string(
00210             "Cannot access the db while "
00211             "initializing a member of the OptimizerCatalogue, the reason is:") +
00212             e.what());
00213   }
00214 }
```

6.22.3.3 insertMapping()

```
void minerule::OptimizerCatalogue::Catalogue::insertMapping (
            const std::string & table,
            const KeyCols & origKeyCols,
            int refKeyId,
            OrderType orderType )
```

Definition at line 104 of file OptimizerCatalogue.cpp.

```
00106                                 {
00107   mrdb::Connection *connection =
00108       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00109 
00110   try {
00111     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00112 
00113     std::string getKeyQuery =
00114         std::string("SELECT " + getSchemaInfo("tab_cols_col_name") + " " +
00115                     "FROM " + getSchemaInfo("tab_cols") + " " + "WHERE " +
00116                     getSchemaInfo("tab_cols_key_id") + "=") +
00117         Converter((long)refKey).toString();
00118 
00119     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getKeyQuery));
00120     KeyCols refKeyCols;
00121 
00122     while (rs->next()) {
00123       refKeyCols.insert(rs->getString(1));
00124     }
00125 
00126     //      OrderType orderType = getOrderType(origKey,refKey);
00127     // (*this)[table][cols] = pair<KeyCols,OrderType>(refKeyCols,orderType);
00128     std::pair< KeyCols, std::pair<KeyCols, OrderType> > insElem(
00129         cols, std::pair<KeyCols, OrderType>(refKeyCols, orderType));
00130     (*this)[table].insert(insElem);
00131 
00132   } catch (mrdb::SQLException &e) {
00133     throw MineruleException(
00134         MR_ERROR_DATABASE_ERROR,
```

```
00135         std::string("Cannot access the db while "
00136                     "initializing a member of the OptimizerCatalogue,"
00137                     "the reason is:") +
00138             e.what());
00139   }
00140 }
```

6.22.3.4 stringToOrder()

```
OptimizerCatalogue::OrderType minerule::OptimizerCatalogue::Catalogue::stringToOrder (
            const std::string & str )  [static]
```

Definition at line 143 of file OptimizerCatalogue.cpp.

```
00143                                                                    {
00144   if (str == "r")
00145     return Reversed;
00146
00147   if (str == "e")
00148     return Equal;
00149
00150   return None;
00151 }
```

6.22.4 Field Documentation

6.22.4.1 elements

```
T std::map< K, T >::elements  [inherited]
```

STL member.

6.22.4.2 keys

```
K std::map< K, T >::keys  [inherited]
```

STL member.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp
- /Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp

6.23 minerule::CatalogueInfo Class Reference

```
#include <OptimizerCatalogue.hpp>
```

Public Member Functions

- void updateQrySize ()

Data Fields

- std::string qryName
- std::string qryText
- std::string resName
- std::vector< std::string > resTables
- size_t resSize

6.23.1 Detailed Description

Definition at line 50 of file OptimizerCatalogue.hpp.

6.23.2 Member Function Documentation

6.23.2.1 updateQrySize()

```
void minerule::CatalogueInfo::updateQrySize ( )
```

Definition at line 542 of file OptimizerCatalogue.cpp.

```
00542                                    {
00543   ParsedMinerule p;
00544   MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00545   p.init(qryText);
00546   MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00547
00548   std::string qry = "SELECT count(*) "
00549                     "FROM " +
00550                     resName + " " + "WHERE supp>=" +
00551                     Converter(p.sup).toString() + " AND con>=" +
00552                     Converter(p.conf).toString();
00553
00554   mrdb::Connection *connection =
00555       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00556   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00557   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(qry));
00558
00559   MRDebug() << "updateQrySize, the query is:" << qry << std::endl;
00560
00561   if (rs->next()) {
00562     resSize = rs->getInt(1);
00563   } else {
00564     throw MineruleException(MR_ERROR_CATALOGUE_ERROR,
00565                             "CatalogueInfo::updateQrySize - cannot"
00566                             " retrieve the size of the result set");
00567   }
00568 }
```

6.23.3 Field Documentation

6.23.3.1 qryName

```
std::string minerule::CatalogueInfo::qryName
```

Definition at line 52 of file OptimizerCatalogue.hpp.

6.23.3.2 qryText

```
std::string minerule::CatalogueInfo::qryText
```

Definition at line 53 of file OptimizerCatalogue.hpp.

6.23.3.3 resName

```
std::string minerule::CatalogueInfo::resName
```

Definition at line 54 of file OptimizerCatalogue.hpp.

6.23.3.4 resSize

```
size_t minerule::CatalogueInfo::resSize
```

Definition at line 56 of file OptimizerCatalogue.hpp.

6.23.3.5 resTables

```
std::vector<std::string> minerule::CatalogueInfo::resTables
```

Definition at line 55 of file OptimizerCatalogue.hpp.

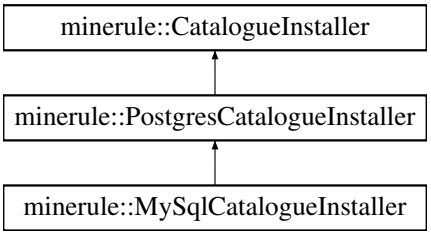
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp
- /Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp

6.24 minerule::CatalogueInstaller Class Reference

`#include <CatalogueInstaller.hpp>`

Inheritance diagram for minerule::CatalogueInstaller:



Public Types

- enum SupportedDbms { MySql , Postgres }

Public Member Functions

- CatalogueInstaller ()
- virtual ~CatalogueInstaller ()
- virtual void installMRQuery ()=0
- virtual void installMRAttList ()=0
- virtual void installMREqKeys ()=0
- virtual void installMREqKeysCol ()=0
- virtual void installMRDepFun ()=0
- virtual void installMRDepFunCol ()=0
- virtual void installMRAutoincrement ()=0
- virtual void initializeAutoincrement ()=0
- virtual void dropMRQuery ()=0
- virtual void dropMRAttList ()=0
- virtual void dropMREqKeys ()=0
- virtual void dropMREqKeysCol ()=0
- virtual void dropMRDepFun ()=0
- virtual void dropMRDepFunCol ()=0
- virtual void dropMRAutoincrement ()=0
- virtual void install ()
- virtual void uninstall ()

Static Public Member Functions

- static CatalogueInstaller * newInstaller (SupportedDbms dbms)
- static CatalogueInstaller * newInstaller ()

Protected Attributes

- mrdb::Connection * _connection
- mrdb::Statement * _statement

6.24.1 Detailed Description

Definition at line 24 of file CatalogueInstaller.hpp.

6.24.2 Member Enumeration Documentation

6.24.2.1 SupportedDbms

```
enum minerule::CatalogueInstaller::SupportedDbms
```

Enumerator

MySql	
Postgres	

Definition at line 29 of file CatalogueInstaller.hpp.

```
00029 { MySql, Postgres } SupportedDbms;
```

6.24.3 Constructor & Destructor Documentation

6.24.3.1 CatalogueInstaller()

```
minerule::CatalogueInstaller::CatalogueInstaller ( )
```

Definition at line 23 of file CatalogueInstaller.cpp.

```
00023 : _connection( MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection() ), _statement(
      _connection->createStatement() ) {}
```

6.24.3.2 ~CatalogueInstaller()

```
virtual minerule::CatalogueInstaller::~CatalogueInstaller ( )  [inline], [virtual]
```

Definition at line 32 of file CatalogueInstaller.hpp.

```
00032                                       {
00033                         delete _statement;
00034                 };
```

6.24.4 Member Function Documentation

6.24.4.1 dropMRAttList()

```
virtual void minerule::CatalogueInstaller::dropMRAttList ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.2 dropMRAutoincrement()

```
virtual void minerule::CatalogueInstaller::dropMRAutoincrement ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.3 dropMRDepFun()

```
virtual void minerule::CatalogueInstaller::dropMRDepFun ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.4 dropMRDepFunCol()

```
virtual void minerule::CatalogueInstaller::dropMRDepFunCol ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.5 dropMREqKeys()

```
virtual void minerule::CatalogueInstaller::dropMREqKeys ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.6 dropMREqKeysCol()

```
virtual void minerule::CatalogueInstaller::dropMREqKeysCol ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.7 dropMRQuery()

```
virtual void minerule::CatalogueInstaller::dropMRQuery ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.8 initializeAutoincrement()

```
virtual void minerule::CatalogueInstaller::initializeAutoincrement ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.9 install()

```
virtual void minerule::CatalogueInstaller::install ( )  [inline], [virtual]
```

Definition at line 57 of file CatalogueInstaller.hpp.

```
00057                                   {
00058                         uninstall();
00059
00060                         installMRQuery();
00061                         installMRAttList();
00062                         installMREqKeys();
00063                         installMREqKeysCol();
00064                         installMRDepFun();
00065                         installMRDepFunCol();
00066                         installMRAutoincrement();
00067                         initializeAutoincrement();
00068                 }
```

6.24.4.10 installMRAttList()

```
virtual void minerule::CatalogueInstaller::installMRAttList ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.11 installMRAutoincrement()

```
virtual void minerule::CatalogueInstaller::installMRAutoincrement ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.12 installMRDepFun()

```
virtual void minerule::CatalogueInstaller::installMRDepFun ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.13 installMRDepFunCol()

```
virtual void minerule::CatalogueInstaller::installMRDepFunCol ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.14 installMREqKeys()

```
virtual void minerule::CatalogueInstaller::installMREqKeys ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.15 installMREqKeysCol()

```
virtual void minerule::CatalogueInstaller::installMREqKeysCol ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.16 installMRQuery()

```
virtual void minerule::CatalogueInstaller::installMRQuery ( )  [pure virtual]
```

Implemented in minerule::PostgresCatalogueInstaller.

6.24.4.17 newInstaller() [1/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller ( )  [static]
```

Definition at line 34 of file CatalogueInstaller.cpp.

```
00034                                                    {
00035                 std::string dbmsStr = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00036 
00037                 if(dbmsStr == "mysql") return newInstaller(MySql);
00038                 if(dbmsStr == "postgres") return newInstaller(Postgres);
00039 
00040                 throw MineruleException(MR_ERROR_OPTION_CONFIGURATION,std::string("Option mrdb::dbms
    is set to an unsupported value.") +
00041                         " Current value is:"+dbmsStr+ ". Supported values are 'postgres' and
    'mysql'.");
00042         }
```

6.24.4.18 newInstaller() [2/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller (
            SupportedDbms dbms )  [static]
```

Definition at line 25 of file CatalogueInstaller.cpp.

```
00025                                                                    {
00026                 switch(dbms) {
00027                 case MySql:
00028                         return new MySqlCatalogueInstaller();
00029                 case Postgres:
00030                         return new PostgresCatalogueInstaller();
00031                 }
00032         }
```

6.24.4.19 uninstall()

```
virtual void minerule::CatalogueInstaller::uninstall ( )  [inline], [virtual]
```

Definition at line 70 of file CatalogueInstaller.hpp.

```
00070                                 {
00071                         dropMRQuery();
00072                         dropMRAttList();
00073                         dropMREqKeys();
00074                         dropMREqKeysCol();
00075                         dropMRDepFun();
00076                         dropMRDepFunCol();
00077                         dropMRAutoincrement();
00078                 }
```

6.24.5 Field Documentation

6.24.5.1 _connection

```
mrdb::Connection* minerule::CatalogueInstaller::_connection  [protected]
```

Definition at line 26 of file CatalogueInstaller.hpp.

6.24.5.2 _statement

```
mrdb::Statement* minerule::CatalogueInstaller::_statement  [protected]
```

Definition at line 27 of file CatalogueInstaller.hpp.

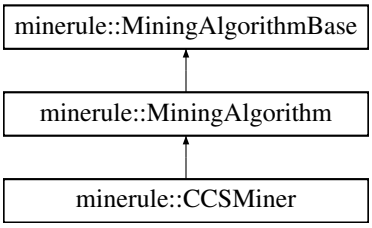
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/CatalogueInstaller.hpp
- /Users/esposito/Software/minerule/src/Optimizer/CatalogueInstaller.cpp

6.25 minerule::CCSMiner Class Reference

`#include <CCSMiner.hpp>`

Inheritance diagram for minerule::CCSMiner:



Public Member Functions

- CCSMiner (const OptimizedMinerule &mr, const AlgorithmsOptions &opts)
- virtual ~CCSMiner ()
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual SourceTableRequirements sourceTableRequirements () const
- bool find (std::vector< CCSMSequence * > *vec, CCSMSequence *elem)
- void combina (std::vector< CCSMSequence::ResultItems > &, std::vector< CCSMSequence * > *k2, size_t k, int min_g, int max_g, double threshold, int check)
- virtual void initialize ()
- virtual void execute ()
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- const OptimizedMinerule & minerule

6.25.1 Detailed Description

Definition at line 26 of file CCSMiner.hpp.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 CCSMiner()

```
minerule::CCSMiner::CCSMiner (
            const OptimizedMinerule & mr,
            const AlgorithmsOptions & opts )  [inline]
```

Definition at line 36 of file CCSMiner.hpp.

```
00036                                                                    :
00037       MiningAlgorithm(mr),  statement(NULL) {
00038         options = opts;
00039         connection.useMRDBConnection(connection.getMRDBConnection());
00040 }
```

6.25.2.2 ~CCSMiner()

```
virtual minerule::CCSMiner::~CCSMiner ( )  [inline], [virtual]
```

Definition at line 42 of file CCSMiner.hpp.

```
00042 {}
```

6.25.3 Member Function Documentation

6.25.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.25.3.2 canHandleMinerule()

```
virtual bool minerule::CCSMiner::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 46 of file CCSMiner.hpp.

```
00046                                          {
00047       return minerule.getParsedMinerule().miningTask == MTMineSequences;
00048     }
```

6.25.3.3 combina()

```
void minerule::CCSMiner::combina (
            std::vector< CCSMSequence::ResultItems > & result,
            std::vector< CCSMSequence * > * k2,
            size_t k,
            int min_g,
            int max_g,
            double threshold,
            int check )
```

Definition at line 69 of file CCSMiner.cpp.

```
00069                                                                                       {
00070           size_t seq_length=0;
00071           time_t init1, end1, init, end;
00072 
00073       std::vector<size_t> pos_canc;
00074       std::vector<CCSMSequence*>* rif_singleton= k2;
00075       std::vector<CCSMSequence*>* k_rif;
00076 
00077 
00078       std::vector<CCSMSequence*>* k_2=k2;
00079       std::vector<CCSMSequence*>* k_1=NULL;
00080 
00081           std::list<CCSMSequence*> suff;
00082           std::list<CCSMSequence*> pre;
00083           std::list<CCSMSequence*>::iterator pre_it;
00084           std::list<CCSMSequence*>::iterator suf_it;
00085       std::vector<CCSMSequence*>* da_cancellare = new std::vector<CCSMSequence*>() ;
00086           bool canContinue=true;
00087           while (seq_length!=k&&canContinue){
00088                   time(&init1);
00089                   //utilizzato per la fase di transaction reduction, mi memorizza le seq che sono di livello k e da cui al passo
00090                   //successivo andro a generare le seq di livello k+1
00091 
00092                   k_rif = new std::vector<CCSMSequence*>();
00093 
00094 
00095                   k_1 = new std::vector<CCSMSequence*>();
00096                   canContinue=false;
00097                   int cont=0;
00098                   for (size_t i=0;i<k_2->size();++i){
00099                           //partiziono lo spazio utilizzando le sequenze che hanno (*k_2)[i] come suffisso e come prefisso
00100                           //(quelle di lunghezza k-1)e le combino per ottenere le sequenze di lunghezza k
00101                           suff = (*k_2)[i]->getSuffixSequence();
00102                           pre =  (*k_2)[i]->getPrefixSequence();
00103 
00104                           for (pre_it=pre.begin();pre_it!=pre.end();++pre_it){
00105                                   for (suf_it=suff.begin();suf_it!=suff.end();++suf_it){

00106                                           time(&init1);
00107                                           CCSMSequence* dino =
00108 CCSMSequence::merge((*suf_it),(*pre_it)->getLastItem(),min_g,max_g,threshold);
00109                                           time(&end1);
00110 /*                        //              std::cout«"tempo di generazione "«difftime(end1,init1)«std::endl;
00111                           //              std::cout«(*suf_it)->toStdString()«" + "«std::endl«" "«(*pre_it)->toStdString();
00112                           //              std::cout«" = "«std::endl;
00113                                           std::cout«" "«dino->toStdString()«" con conteggio "«dino->getCount()«std::endl;
00114 */
00115                                           if (dino->getCount()>=threshold){
00116                                                   //inserisco i punatori alle sequenze di lunghezza k
00117                                                   //nelle Prefix/Suffix list degli elementi k-1
00118 
00119                                                   k_rif->push_back(dino);//utilizzato per la fase di transaction reduction
00120 
00121                                                   (*suf_it)->addPrefixSequence(dino);

00122                                                   (*pre_it)->addSuffixSequence(dino);
00123                                                   //cout«"sequenza "«dino->toStdString()«std::endl;
```

```
00124                                                   result.push_back(dino->getSequenceItems());
00125 //                                                        std::cout<<"OK"<<std::endl;
00126                                                   }else
00127                                                           delete dino;
00128                                           }
00129                                   }
00130                           }
00131                   //controllo se al passo successivo e ancora possibile generare dei possibili
      candidati
00132                   //inserisco nell'insieme k-1 gli elementi che appartengono alla lista dei
      suffissi o dei prefissi
00133                   //delle sequenze degli elementi di k-2
00134
00135
00136                           for (size_t i=0;i<k_2->size();++i){
00137
00138                                   pre =  (*k_2)[i]->getPrefixSequence();
00139
00140                                   for (pre_it=pre.begin();pre_it!=pre.end();++pre_it){
00141                                   //canGenerate testa se per quel determinato elemento c'e
      almeno un elemento nella lista dei suffissi
00142                                   //ed uno in quella dei prefissi, in modo che possa generare
      una possibile sequenza frequente di livello k
00143                                   //cout<<(*pre_it)->toStdString()<<" ";
00144                                           if ((*pre_it)->canGenerate()){
00145                                           //cout<<"puo generare"<<std::endl;
00146                                                   bool trovato=find(k_1,(*pre_it));
00147                                                   if (!trovato)
00148                                                           k_1->push_back((*pre_it));
00149                                                   cont++;
00150                                           //se ci sono sequenze che possono generare seq di
      livello k allora si puo continuare nel ciclo
00151                                                   canContinue=true;
00152                                                   }//se la sequenza non puo essere usata per
      generare ulteriori sequenze allora
00153                                   // viene inserita in un vettore di seq da cancellare
00154                                                   else{
00155                                                           bool trovato =
      find(da_cancellare,(*pre_it));
00156                                                           if (!trovato)
00157
      da_cancellare->push_back((*pre_it));
00158                                           //cout<<"inserisco in da_cancellare"<<std::endl;
00159
00160                                                   }
00161                                           }
00162                                           suff = (*k_2)[i]->getSuffixSequence();
00163
00164                                           for (suf_it=suff.begin();suf_it!=suff.end();++suf_it){
00165                                   //cout<<(*suf_it)->toStdString()<<" ";
00166                                                   if ((*suf_it)->canGenerate()){
00167                                           //se la sequenza non è gia presente allora si
      inserisce altrimenti niente
00168                                           //cout<<"puo generare"<<std::endl;
00169                                                           bool trovato=find(k_1,(*suf_it));
00170                                                           if (!trovato)
00171                                                                   k_1->push_back((*suf_it));
00172                                                           cont++;
00173                                                           canContinue=true;
00174                                                   }
00175                                                   else {
00176                                                           bool trovato =
      find(da_cancellare,(*suf_it));
00177                                                           if (!trovato)
00178
      da_cancellare->push_back((*suf_it));
00179                                           //cout<<"inserisco in da_cancellare"<<std::endl;
00180
00181                                                   }
00182                                           }
00183                                   }
00184
00185                                std::vector<CCSMSequence*>::iterator del;
00186                                   if (seq_length!=0){
00187                                           for (del = k_2->begin();del!=k_2->end();++del){
00188                                                   delete (*del);
00189                                           }
00190                                           delete k_2;
00191                                   }
00192
00193                                   time(&end1);
00194                                   k_2 = k_1;
00195
00196                                   if (check>1&&canContinue){
00197                                           time(&init);
00198
00199                                           pos_canc = CCSMSequence::reduce_tr(k_rif);
```

```
00200                                                   time(&end);
00201                                                   //      std::cout«"tempo per scovare le transazioni
non interessanti "«difftime(end,init)«std::endl;
00202                                                   if (pos_canc.size()>1){
00203                                                           time(&init);
00204                                                           for
(del=rif_singleton->begin();del!=rif_singleton->end();++del){
00205                                   //cout«"ridotto singleton"«std::endl;
00206                                                                   (*del)->reduction(pos_canc);
00207                                                           }
00208                                                           time(&end);
00209                                                           //      std::cout«"tempo riduzione di
"«rif_singleton->size()«" singleton "«difftime(end,init)«std::endl;
00210                                                           time(&init);
00211                                                           for
(del=k_rif->begin();del!=k_rif->end();++del){
00212                                           //cout«"ridotta coppia"«std::endl;
00213                                                                   (*del)->reduction(pos_canc);
00214                                                           }
00215                                                           time(&end);
00216                                                           //      std::cout«"tempo riduzione di
"«k_rif->size()«" non singleton "«difftime(end,init)«std::endl;
00217                                                   }
00218                                           }
00219                                           delete k_rif;
00220
00221                                           //      std::cout«"transazioni eliminate "«pos_canc.size()«"
modalita' "«check«std::endl;
00222                                           seq_length++;
00223         //              std::cout«"da cancellare size "«da_cancellare->size()«std::endl;
00224                                  std::vector<CCSMSequence*>::iterator da_c;
00225                 /*      canContinue=false;*/
00226
00227                                           if (seq_length!=k&&canContinue){
00228                                                   for ( da_c=
da_cancellare->begin();da_c!=da_cancellare->end();++da_c){
00229         //                        //      std::cout«"cancello "«(*da_c)->toStdString()«" che hano sup
"«(*da_c)->getCount()«std::endl;
00230                                                           delete (*da_c);
00231                                                   }
00232         //                      std::cout«"finita cancellazione: "«da_cancellare->size()«std::endl;
00233                                                   delete da_cancellare;
00234                                                   da_cancellare = new std::vector<CCSMSequence*>();
00235                                           }
00236
00237
00238                                   }
00239                           std::vector<CCSMSequence*>::iterator del, it_kk;
00240                                   std::list<CCSMSequence*>::iterator del1;
00241                                   for (del = k_1->begin();del!=k_1->end();++del)
00242                                           delete (*del);
00243         //      std::cout«"fuori dal ciclo"«std::endl;
00244
00245                           std::vector<CCSMSequence*>* temp = new std::vector<CCSMSequence*>();
00246
00247                                   for (it_kk=
da_cancellare->begin();it_kk!=da_cancellare->end();++it_kk){
00248                                           suff = (*it_kk)->getSuffixSequence();
00249                                           pre =  (*it_kk)->getPrefixSequence();
00250
00251                                           for (del1=suff.begin();del1!=suff.end();del1++)
00252                                                   temp->push_back(*del1);
00253
00254                                           for (del1=pre.begin();del1!=pre.end();++del1)
00255                                                   temp->push_back(*del1);
00256                                   }
00257                                   for (it_kk = temp->begin();it_kk!=temp->end();++it_kk){
00258
00259                                           bool trov1 =find(da_cancellare,(*it_kk));
00260                                           if (!trov1)
00261                                                   da_cancellare->push_back((*it_kk));
00262                                   }
00263                                   delete temp;
00264
00265                                   for (del= da_cancellare->begin();del!=da_cancellare->end();++del){
00266                                           delete (*del);
00267                                   }
00268         //      std::cout«"finita cancellazione: "«da_cancellare->size()«std::endl;
00269                                   delete da_cancellare;
00270                                   delete k_1;
00271                           }
```

6.25.3.4 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                   {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.25.3.5 find()

```
bool minerule::CCSMiner::find (
            std::vector< CCSMSequence * > * vec,
            CCSMSequence * elem )
```

Definition at line 57 of file CCSMiner.cpp.

```
00057                                                                                   {
00058          std::vector<CCSMSequence*>::iterator it;
00059                 bool trovato = false;
00060                 for (it=vec->begin();it!=vec->end()&&!trovato;++it){
00061                         if ((*(*it)) == *elem)
00062                                 trovato =true;
00063                 }
00064                 return trovato;
00065         }
```

6.25.3.6 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                     {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
        connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
        connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
        connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
        connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
        minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.25.3.7 mineRules()

```
void minerule::CCSMiner::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 276 of file CCSMiner.cpp.

```
00276                                   {
00277                                 double time_tot=0;
00278                                 time_t init, end;
00279
00280                                 const ParsedMinerule& pm = minerule.getParsedMinerule();
00281
00282                                 int min_g=pm.sequenceAllowedGaps.getMin();
00283                                 int max_g=pm.sequenceAllowedGaps.getMax();
00284
00285                                 double threshold=pm.sup;
00286                                 if (min_g>max_g)
00287                                         assert(false);
00288
00289                                 int reduce = 1;
00290                           std::vector<CCSMSequence::ResultItems> result;
00291                           std::vector<CCSMSequence*> singleton;
00292                                 CCSMSequence input;
00293                                 CCSMSequence* single;
00294                           std::vector<CCSMSequence*>::iterator it;
00295                                 size_t number=0;
00296                                 time(&init);
00297                                 int riga=0;
00298
00299
00300                                 prepareData();
00301                                 mrdb::ResultSet* rs = statement->executeQuery();
00302
00303                                 rs->next();
00304                                 while (!rs->isAfterLast()){
00305                                         // SourceRow hbsr(rs,rowDes);
00306                                         input.read(rs, rowDes);
00307                                         // std::cout<<"LEGGO SEQUENZA: "<<input.toStdString()<<std::endl;
00308                                         for(size_t i=0;i<input.size();++i){
00309
00310                                                 single= new CCSMSequence(input,i,1);
00311
00312                                                 single->setLastItem(single);
00313                                                 bool trovato=false;
00314
00315                                                 for (it =
      singleton.begin();it!=singleton.end()&&!trovato;it++){
00316                                                         CCSMSequence* suppSingle=(*it);
00317                                                         if (*suppSingle == *single){
00318                                                                 trovato = true;
00319                                                                 (*it)->setPresent(number,1);
00320                                                                 (*it)->setEid(number,i);
00321                                                         }
00322                                                 }
00323                                                 if (!trovato){
00324                                                         single->setPresent(number,1);
00325                                                         single->setEid(number,i);
00326                                                         singleton.push_back(single);
00327                                                 }
00328                                                 else
00329                                                         delete single;
00330                                         }
00331                                         input.svuota();
00332                                         number++;
00333                                         riga++;
00334                                 }
00335
00336                                 std::cout<<std::endl;
00337                                 for (size_t i=0;i<singleton.size();i++){
00338                                         singleton[i]->setPresent(number-1,0);
00339                                 }
00340                                 //      std::cout<<"numbe e uguale a: "<<number<<std::endl;
00341                                 //                              in.close();
00342                                 //      std::cout<<"size "<<singleton.size()<<std::endl;
00343                                 time(&end);
00344                                 // double ttt = difftime(end,init);
00345                                 //                      std::cout<<"tempo finora "<<ttt<<std::endl;
00346                                 //      std::cout<<"tempo medio per riga"<<ttt/number<<std::endl;
00347                                 time_tot=time_tot+difftime(end,init);
00348                                 time(&init);
00349
00350                                 threshold=number*threshold;
```

```
00351                                   if (threshold == 0)
00352                                           threshold = 0.00001;
00353
00354                           std::vector<CCSMSequence*>* suppSingleton= new std::vector<CCSMSequence*>();
00355                           std::vector<CCSMSequence*>::iterator itt;
00356
00357                                   for (itt=singleton.begin();itt!=singleton.end();++itt){
00358                                           if ((*itt)->getCount()>=threshold){
00359                                                   suppSingleton->push_back((*itt));
00360                                           }
00361                                           else delete (*itt);
00362                                   }
00363
00364                                   //      std::cout«"tempo finora "«difftime(end,init)«std::endl;
00365                                   time_tot=time_tot+difftime(end,init);
00366
00367                                   if (reduce>0){
00368                                           time(&init);
00369                                           std::vector<size_t> temp =
      CCSMSequence::reduce_tr(suppSingleton);
00370
00371                                           if (temp.size()!=0){
00372                                                   for
      (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt){
00373                                                           (*itt)->reduction(temp);
00374                                                   }
00375                                           }
00376                                           time(&end);
00377                                           //      std::cout«"transazioni eliminate sui
      singleton"«temp.size()«std::endl;
00378                                           //cout«"tempo della riduzione delle transazioni
      "«difftime(end,init)«std::endl;
00379                                           time_tot=time_tot+difftime(end,init);
00380                                   }
00381
00382                                   time(&init);
00383                                   time_t init1, end1;
00384                           std::vector<CCSMSequence*>* coppie= new std::vector<CCSMSequence*>();
00385                                   //      std::cout«"la threshold e uguale a: "«threshold«std::endl;
00386                                   for (size_t i=0;i<suppSingleton->size();++i){
00387                                           for (size_t j=0;j<suppSingleton->size();++j){
00388
00389                                                   time(&init1);
00390
00391                                                   CCSMSequence* dino =
00392
      CCSMSequence::merge((*suppSingleton)[i],(*suppSingleton)[j]->getLastItem(),min_g,max_g,threshold);
00393                                                   time(&end1);
00394                                                   if (dino->getCount()>=threshold){
00395                                                           coppie->push_back(dino);
00396                                                           (*suppSingleton)[i]->addPrefixSequence(dino);
00397                                                           (*suppSingleton)[j]->addSuffixSequence(dino);
00398                                                           result.push_back(dino->getSequenceItems());
00399
00400                                                           }else
00401                                                                   delete dino;
00402                                                   }
00403                                           }
00404                                           time(&end);
00405                                           time_tot=time_tot+difftime(end,init);
00406                                           //cout«"tempo x generare gli insiemi F1 ed F2
      "«difftime(end,init)«std::endl;
00407
00408
00409                                           if (reduce>0){
00410                                                   time(&init);
00411                           //std::vector<size_t>
00412                                                   std::vector<size_t> temp =
      CCSMSequence::reduce_tr(coppie);
00413                                                   if (temp.size()!=0){
00414                                                           for
      (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt){
00415                                   //cout«"ridotto singleton"«std::endl;
00416                                                                   (*itt)->reduction(temp);
00417                                                           }
00418                                                           for
      (itt=coppie->begin();itt!=coppie->end();++itt){
00419                                   //cout«"ridotta coppia"«std::endl;
00420                                                                   (*itt)->reduction(temp);
00421                                                           }
00422                                                   }
00423                                                   time(&end);
00424                                                   //      std::cout«"transazioni eliminate
      "«temp.size()«std::endl;
00425                                                   //cout«"tempo della riduzione delle transazioni
      "«difftime(end,init)«std::endl;
00426                                                   time_tot=time_tot+difftime(end,init);
```

```
00427                                       }
00428                                       //
00429                                       //              std::cout«"prima di combina"«std::endl;
00430                                       //cout«"cardinalita F1 : "«suppSingleton->size()«std::endl;
00431                                       time(&init);
00432
00433                                       combina(result, suppSingleton ,10,
      min_g,max_g,threshold,reduce);
00434
00435                                       //cout«"dopo combina"«std::endl;
00436                                       time(&end);
00437
00438                                       for (size_t i=0;i<result.size();++i){
00439                                        std::vector<ItemType>::iterator it_list =
      result[i].first.begin();
00440                                               for (;it_list!=result[i].first.end();++it_list)
00441                                                       std::cout«(*it_list)«" ";
00442                                               std::cout«" con conteggio: "«result[i].second;
00443                                               std::cout«std::endl;
00444                                       }
00445
00446                                       time_tot=time_tot+difftime(end,init);
00447                                       //      std::cout«"tempo x generare insiemi da F3....
      "«difftime(end,init)«std::endl;
00448                                       //cout«"tempo totale "«time_tot«std::endl;
00449                                       for
      (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt)
00450                                               delete (*itt);
00451                                       delete coppie;
00452 /*      for (itt=coppie.begin();itt!=coppie.end();++itt)
00453                                       delete (*itt);*/
00454                                               delete suppSingleton;
00455                                       std::cout«"cardinalita insieme risultato
      "«result.size()«std::endl;
00456                                       //      std::cout«"threshold "«threshold«std::endl;
00457
00458                                }
```

6.25.3.8 optimizedMinerule()

virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const [inline], [virtual], [inherited]

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.25.3.9 sourceTableRequirements()

virtual SourceTableRequirements minerule::CCSMiner::sourceTableRequirements ( ) const [inline], [virtual]

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 50 of file CCSMiner.hpp.

```
00050                                                                   {
00051                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00052                 };
```

6.25.4 Field Documentation

6.25.4.1 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMiner.hpp
- /Users/esposito/Software/minerule/src/Algorithms/CCSMiner.cpp

6.26 minerule::CCSMSequence Class Reference

```
#include <CCSMSequence.hpp>
```

Public Types

- typedef std::vector< std::pair< int, int > > Eid_List
- typedef std::pair< std::vector< ItemType >, int > ResultItems

Public Member Functions

- CCSMSequence (CCSMSequence *l=NULL)
- CCSMSequence (const CCSMSequence &in, size_t start, size_t n_item, CCSMSequence *l=NULL)
- CCSMSequence (const CCSMSequence &in)
- void reduction (std::vector< size_t > &k)
- size_t size ()
- bool operator< (const CCSMSequence &in)
- bool operator== (const CCSMSequence &in)
- void add (const ItemType &s)
- std::string toStdString ()
- void setPresent (size_t sid, bool isHere)
- void setEid (size_t sid, size_t eid)
- void stampaEid (size_t sid)
- void bitMaptoString ()
- void svuota ()
- ~CCSMSequence ()
- int getCount ()
- bool isSingleton ()
- ItemType getFirstItem ()
- CCSMSequence * getLastItem ()
- void setLastItem (CCSMSequence *in)
- ResultItems getSequenceItems ()
- void addPrefixSequence (CCSMSequence *in)
- std::list< CCSMSequence * > getPrefixSequence ()
- void addSuffixSequence (CCSMSequence *in)
- std::list< CCSMSequence * > getSuffixSequence ()
- bool canGenerate ()
- void read (mrdb::ResultSet *rs, const SourceRowColumnIds &rowDes)

Static Public Member Functions

- static CCSMSequence * merge (CCSMSequence *first, CCSMSequence *second, int min_gap, int max_↩ gap, double threshold)
- static std::vector< size_t > reduce_tr (std::vector< CCSMSequence * > *vec)

6.26.1 Detailed Description

Definition at line 32 of file CCSMSequence.hpp.

6.26.2 Member Typedef Documentation

6.26.2.1 Eid_List

```
typedef std::vector<std::pair<int,int> > minerule::CCSMSequence::Eid_List
```

Definition at line 36 of file CCSMSequence.hpp.

6.26.2.2 ResultItems

```
typedef std::pair<std::vector<ItemType>,int> minerule::CCSMSequence::ResultItems
```

Definition at line 59 of file CCSMSequence.hpp.

6.26.3 Constructor & Destructor Documentation

6.26.3.1 CCSMSequence() [1/3]

```
minerule::CCSMSequence::CCSMSequence (
            CCSMSequence * l = NULL )  [inline]
```

Definition at line 61 of file CCSMSequence.hpp.

```
00061                                           {
00062                 singleton=false;
00063                 count=0;
00064                 seq = new List_Type();
00065                 last=l;
00066                 listSid.reserve(1000000);
00067                 eid_vector.reserve(1000000);
00068                 /*cout<<"bit_vector.capacity "<<listSid.capacity()<<std::endl;
00069                 std::cout<<"bit_vector.max_size "<<listSid.max_size()<<std::endl;
00070                                 std::cout<<"eid_vector.capacity "<<eid_vector.capacity()<<std::endl;
00071                 std::cout<<"eid_vector.max_size "<<eid_vector.max_size()<<std::endl;
00072                 getchar();*/
00073         };
```

6.26.3.2 CCSMSequence() [2/3]

```
minerule::CCSMSequence::CCSMSequence (
            const CCSMSequence & in,
            size_t start,
            size_t n_item,
            CCSMSequence * l = NULL )  [inline]
```

Definition at line 75 of file CCSMSequence.hpp.

```
00075                                                                                                   {
00076                 seq = new List_Type();
00077                 if (n_item==1){
00078                         seq->push_back((*in.seq)[start]);
00079                 }
00080                 else {
00081                         List_Type::iterator it = in.seq->begin();
00082                         for (size_t i=0;i<start;++i){
00083                                 ++it;
00084                         }
00085                         for (size_t i=start;(i<in.seq->size())&&(i<start+n_item);++i){
00086                                 seq->push_back(*it);
00087                                 ++it;
00088                         }
00089                 }
00090                 count=0;
00091                 if (seq->size()==1)
00092                         singleton=true;
00093                 else singleton=false;
00094                 last=l;
00095         }
```

6.26.3.3 CCSMSequence() [3/3]

```
minerule::CCSMSequence::CCSMSequence (
            const CCSMSequence & in )  [inline]
```

Definition at line 97 of file CCSMSequence.hpp.

```
00097                                                 {
00098                 seq= new List_Type();
00099                 *seq=*in.seq;
00100                 count=0;
00101                 if (seq->size()==1)
00102                         singleton=true;
00103                 else singleton=false;
00104                 listSid = in.listSid;
00105                 last = in.last;
00106                 count=in.count;
00107         }
```

6.26.3.4 ~CCSMSequence()

```
minerule::CCSMSequence::~CCSMSequence ( )  [inline]
```

Definition at line 235 of file CCSMSequence.hpp.

```
00235                         {
00236                 delete seq;
00237          std::vector<Eid_List*>::iterator it_list;
00238                 for (it_list=eid_vector.begin();it_list!=eid_vector.end();++it_list)
00239                         delete (*it_list);
00240
00241         };
```

6.26.4 Member Function Documentation

6.26.4.1 add()

```
void minerule::CCSMSequence::add (
            const ItemType & s )  [inline]
```

Definition at line 165 of file CCSMSequence.hpp.

```
00165                                   {
00166                 seq->push_back(s);
00167                 if (seq->size()==1)
00168                         singleton=true;
00169                 else singleton=false;
00170         }
```

6.26.4.2 addPrefixSequence()

```
void minerule::CCSMSequence::addPrefixSequence (
            CCSMSequence * in )  [inline]
```

Definition at line 312 of file CCSMSequence.hpp.

```
00312                                                   {
00313                 prefix.push_back(in);
00314         }
```

6.26.4.3 addSuffixSequence()

```
void minerule::CCSMSequence::addSuffixSequence (
            CCSMSequence * in )  [inline]
```

Definition at line 322 of file CCSMSequence.hpp.

```
00322                                                   {
00323                 suffix.push_back(in);
00324         }
```

6.26.4.4 bitMaptoString()

```
void minerule::CCSMSequence::bitMaptoString ( )  [inline]
```

Definition at line 222 of file CCSMSequence.hpp.

```
00222                                   {
00223         std::string ris ="";
00224                 for (size_t i=0;i<listSid.size();++i){
00225                         std::cout<<listSid[i]<<std::endl;
00226                         stampaEid(i);
00227                 }
00228         }
```

6.26.4.5 canGenerate()

```
bool minerule::CCSMSequence::canGenerate ( )  [inline]
```

Definition at line 332 of file CCSMSequence.hpp.

```
00332                           {
00333                 return suffix.size()>0 && prefix.size()>0;
00334         }
```

6.26.4.6 getCount()

```
int minerule::CCSMSequence::getCount ( )  [inline]
```

Definition at line 243 of file CCSMSequence.hpp.

```
00243                       {
00244                 return count;
00245         }
```

6.26.4.7 getFirstItem()

```
ItemType minerule::CCSMSequence::getFirstItem ( )  [inline]
```

Definition at line 292 of file CCSMSequence.hpp.

```
00292                               {
00293                 List_Type::iterator it=seq->begin();
00294                 return (*it);
00295         }
```

6.26.4.8 getLastItem()

```
CCSMSequence * minerule::CCSMSequence::getLastItem ( )  [inline]
```

Definition at line 297 of file CCSMSequence.hpp.

```
00297                                   {
00298                 return last;
00299         }
```

6.26.4.9 getPrefixSequence()

```
std::list< CCSMSequence * > minerule::CCSMSequence::getPrefixSequence ( )  [inline]
```

Definition at line 316 of file CCSMSequence.hpp.

```
00316                                                   {
00317                 return prefix;
00318         }
```

6.26.4.10 getSequenceItems()

```
ResultItems minerule::CCSMSequence::getSequenceItems ( )  [inline]
```

Definition at line 305 of file CCSMSequence.hpp.

```
00305                                   {
00306                 ResultItems ris(*seq,count);
00307                 return ris;
00308         }
```

6.26.4.11 getSuffixSequence()

```
std::list< CCSMSequence * > minerule::CCSMSequence::getSuffixSequence ( )  [inline]
```

Definition at line 326 of file CCSMSequence.hpp.

```
00326                                                   {
00327                 return suffix;
00328         }
```

6.26.4.12 isSingleton()

```
bool minerule::CCSMSequence::isSingleton ( )  [inline]
```

Definition at line 247 of file CCSMSequence.hpp.

```
00247                           {
00248                 return singleton;
00249         }
```

6.26.4.13 merge()

```
CCSMSequence * minerule::CCSMSequence::merge (
            CCSMSequence * first,
            CCSMSequence * second,
            int min_gap,
            int max_gap,
            double threshold )  [static]
```

Definition at line 90 of file CCSMSequence.cpp.

```
00090
                        {
00091
00092                 CCSMSequence* ris= new CCSMSequence(*first);//=first;
00093                 ris->setLastItem(second);
00094                 //cout<<"ecco il LAST ITEM"<<std::endl;
00095                 //cout<<ris->getLastItem()->toStdString()<<std::endl;
00096                 //List_Type::iterator it;
00097                 /*
00098                 for (it = second->seq->begin();it!=second->seq->end();it++)
00099                         ris->seq->push_back(*it);
00100                 */
00101                 ris->seq->push_back((*second->seq)[0]);
00102
00103                 ris->count=0;
00104                 BitVector::iterator it_ris, it_second;
00105                 it_ris=ris->listSid.begin();
00106                 it_second = second->listSid.begin();
```

```
00108 /*
00109                 for (size_t i=0;i<first->listSid.size();i++,++it_second,++it_ris){
00110                         bool ris_i =(*it_ris)&(*it_second);
00111                         (*it_ris)=ris_i;
00112                         if (ris_i)
00113                                 ris->count++;
00114                 }
00115                 if (ris->count<threshold){
00116                         //      std::cout«"ESCO 1"«std::endl;
00117                                 return ris;
00118                 }
00120
00121                 it_ris=ris->listSid.begin();
00122                 it_second = second->listSid.begin();
00123                 ris->count=0;
00124 */
00125                 for (size_t i=0;i<first->listSid.size();i++,++it_second,++it_ris){
00126                         bool ris_i =(*it_ris)&(*it_second);
00127                         (*it_ris)=ris_i;
00128
00129                         Eid_List *f=first->eid_vector[i];
00130                         Eid_List *s=second->eid_vector[i];
00131                         if (ris_i && (*s)[s->size()-1].first > (*f)[0].second ){
00132                                 ris->count++;
00133                         }else
00134                                 (*it_ris)=ABSENT;
00135                 }
00136
00137                 if (ris->count<threshold){
00138                         //      std::cout«"ESCO 2"«std::endl;
00139                                 return ris;
00140                 }
00141
00142                 it_ris=ris->listSid.begin();
00143                 bool canContinue=true;
00144                 for (size_t i=0;i<first->listSid.size()&&canContinue;++i,++it_ris){
00145                         Eid_List* event_id;
00146                         if ((*it_ris)){//se la sequenza è presente allora calcolo intersezione tra le
      event_list
00147
      event_id=CCSMSequence::mergeEidRtoLeft(*first->eid_vector[i],*second->eid_vector[i],min_gap,max_gap);
00148                         }
00149                         else
00150                                 event_id=new Eid_List();
00151                         //CORREZIONE DELla lista dei SID per tener conto dell ordine
00152                         //siccome se second appare prima di first il sid per quella transazione e
      messo ad uno erroneamente
00153                         //xcio lo correggo controllando se sono stati generati event_id
00154                         //se non sono stati generati setto il listSid[i] ad ABSENT ed incremento count
      solo quando ho degli event_id
00155
00156                         if ((*it_ris) && event_id->size()==0){
00157                                 (*it_ris)=ABSENT;
00158                                 ris->count--;
00159                         }
00160
00161                         if (ris->count<threshold){
00162                                 canContinue=false;
00163                         }
00164                         ris->eid_vector.push_back(event_id);
00165                 }
00166                 //cout«"fusioni di event list "«cc«std::endl;
00167                 return ris;
00168
00169         };
```

6.26.4.14 operator<()

```
bool minerule::CCSMSequence::operator< (
            const CCSMSequence & in )
```

Definition at line 65 of file CCSMSequence.cpp.

```
00065                                                           {
00066                 return lexicographical_compare(seq->begin(),
00067                                                seq->end(),
00068                                                in.seq->begin(),
00069                                                in.seq->end());
00070
00071         };
```

6.26.4.15 operator==()

```
bool minerule::CCSMSequence::operator== (
            const CCSMSequence & in )
```

Definition at line 73 of file CCSMSequence.cpp.

```
00073                                                   {
00074                 bool first=
00075                         lexicographical_compare(seq->begin(),
00076                                                 seq->end(),
00077                                                 in.seq->begin(),
00078                                                 in.seq->end());
00079                 bool second=
00080                         lexicographical_compare(in.seq->begin(),
00081                                                  in.seq->end(),
00082                                                  seq->begin(),
00083                                                  seq->end());
00084                 return (!first)&&(!second);
00085
00086         };
```

6.26.4.16 read()

```
void minerule::CCSMSequence::read (
            mrdb::ResultSet * rs,
            const SourceRowColumnIds & rowDes )
```

Definition at line 26 of file CCSMSequence.cpp.

```
00026                                                                                                   {
00027         SourceRow hbsr(rs,rowDes);
00028         ItemType gid;
00029         if(!rs->isAfterLast())
00030                 gid = hbsr.getGroup();
00031         while (!rs->isAfterLast() && gid == hbsr.getGroup()) {
00032                 seq->push_back( hbsr.getBody() );
00033
00034                 rs->next();
00035                 if(!rs->isAfterLast())
00036                         hbsr.init(rs,rowDes);
00037         }
00038   }
```

6.26.4.17 reduce_tr()

```
static std::vector< size_t > minerule::CCSMSequence::reduce_tr (
            std::vector< CCSMSequence * > * vec )  [inline], [static]
```

Definition at line 256 of file CCSMSequence.hpp.

```
00256                                                                                                   {
00257                 //cout«"ENTRO NELLA REDUCE_TR"«std::endl;
00258                 time_t init,end;
00259         std::vector<size_t> k;
00260         std::vector<CCSMSequence*>::iterator it;
00261                 BitVector ris;
00262                 if (vec->size()==0)
00263                         return k;
00264                 it=vec->begin();
00265                 //cout«"ecco le sequenze su cui applico la reduce"«std::endl;
00266                 //cout«(*it)->toStdString()«std::endl;
00267                 ris=(*it)->listSid;
00268
00269                 ++it;
00270                 time(&init);
00271                 for (;it!=vec->end();++it){
00272                         //cout«(*it)->toStdString()«std::endl;
```

```
00273                           ris=ris|(*it)->listSid;
00274                   }
00275                   time(&end);
00276
      //cout«"----------"«vec->size()«"------"«difftime(end,init)«"----------------------"«std::endl;
00277                   BitVector::iterator b_it;
00278                   b_it=ris.begin();
00279                   size_t i =0;
00280
00281                   time(&init);
00282                   for (;b_it!=ris.end();++b_it,++i){
00283                           if (!(*b_it))
00284                                   k.push_back(i);
00285                   }
00286                   time(&end);
00287                   //cout«"----------secondo ciclo
      interno------"«difftime(end,init)«"----------------------"«std::endl;
00288                   return k;
00289
00290           }
```

6.26.4.18 reduction()

```
void minerule::CCSMSequence::reduction (
            std::vector< size_t > & k )  [inline]
```

Definition at line 109 of file CCSMSequence.hpp.

```
00109                                           {
00110                   //elimino dalla lista degli Eventi gli eventi corrispondenti alle transazioni che non
      sono utili al fine del conteggio
00111                   //delle frequenze del livello a cui siamo arrivati in modo da migliorare
00112                   //le performance dell'algoritmo e l'occupazione di memoria
00113
00114                   //time_t init,end;
00115                   //time(&init);
00116           std::vector<Eid_List*>::iterator it;
00117                   size_t i =0;
00118                   size_t j=0;
00119                   it = eid_vector.begin();
00120                   //size_t s=eid_vector.size();
00121           std::vector<Eid_List*> eid_temp;
00122
00123                   for (it=eid_vector.begin();it!=eid_vector.end();++it,i++){
00124                           if (k[j]!=i)
00125                                   eid_temp.push_back((*it));
00126                           else {
00127                                   delete (*it);
00128
00129                                   j++;
00130                                }
00131
00132                   }
00133                   /*for (it = eid_canc.begin();it!=eid_canc.end();++it)
00134                                   delete (*it);
00135                   */
00136                   eid_vector=eid_temp;
00137                   //elimino dalla bitmap le sequenze che non sono utili al fine del conteggio delle
      frequenze delle sequenze, in modo da migliorare
00138                   //le performance dell'algoritmo e l'occupazione di memoria
00139                   //time(&init);
00140                   i=0;
00141                   BitVector ris;
00142                   for (size_t f =0;f<listSid.size();++f){
00143                           if (f!=k[i]){
00144                                   ris.push_back(listSid[f]);
00145                           }
00146                           else i++;
00147                   }
00148                   //time(&end);
00149                   //cout«"ridotta listSid in tempo "«difftime(end,init)«std::endl;
00150                   listSid=ris;
00151
00152           }
```

6.26.4.19 setEid()

```
void minerule::CCSMSequence::setEid (
            size_t sid,
            size_t eid )  [inline]
```

Definition at line 198 of file CCSMSequence.hpp.

```
00198                                           {
00199                 //inserisco la posizione dell occorrenza (eventIdentifier)
00200                 //del singleton per
00201                 // questo determinato sid sequenceIdentifier
00202                 std::pair<int,int> couple(eid,eid);
00203                 if (eid_vector.size() == sid)
00204                         eid_vector.push_back(new Eid_List());
00205                 eid_vector[sid]->push_back(couple);
00206
00207         }
```

6.26.4.20 setLastItem()

```
void minerule::CCSMSequence::setLastItem (
            CCSMSequence * in )  [inline]
```

Definition at line 301 of file CCSMSequence.hpp.

```
00301                                           {
00302                 last=in;
00303         }
```

6.26.4.21 setPresent()

```
void minerule::CCSMSequence::setPresent (
            size_t sid,
            bool isHere )  [inline]
```

Definition at line 180 of file CCSMSequence.hpp.

```
00180                                                   {
00181
00182                 if (sid<listSid.size())
00183                         return;
00184         //questo serve per riempire l'ultimo elemento del vettore degli
00185         //Eid_List per quegli elementi che non sono presenti in ultima posizione
00186                 if (sid>listSid.size()){
00187                         for (size_t i=listSid.size();i < sid;i++){
00188                                 listSid.push_back(ABSENT);
00189                                 eid_vector.push_back(new Eid_List());
00190                         }
00191                 }
00192                 listSid.push_back(isHere);
00193                 eid_vector.push_back(new Eid_List());
00194                 if (isHere)
00195                         count++;
00196         }
```

6.26.4.22 size()

```
size_t minerule::CCSMSequence::size ( )  [inline]
```

Definition at line 157 of file CCSMSequence.hpp.

```
00157                   {
00158                 return seq->size();
00159         };
```

6.26.4.23 stampaEid()

```
void minerule::CCSMSequence::stampaEid (
            size_t sid )  [inline]
```

Definition at line 209 of file CCSMSequence.hpp.

```
00209                                   {
00210         std::string ris="";
00211                 Eid_List* temp = eid_vector[sid];
00212                 int a, b;
00213                 for (size_t i=0;i<temp->size();++i){
00214                         a=(*temp)[i].first;
00215                         b=(*temp)[i].second;
00216                         std::cout«"  ("«a«","«b«")";
00217                 }
00218                 std::cout«std::endl;
00219
00220         }
```

6.26.4.24 svuota()

```
void minerule::CCSMSequence::svuota ( )  [inline]
```

Definition at line 230 of file CCSMSequence.hpp.

```
00230                     {
00231                 delete seq;
00232                 seq = new List_Type();
00233                 }
```

6.26.4.25 toStdString()

```
std::string minerule::CCSMSequence::toStdString ( )  [inline]
```

Definition at line 172 of file CCSMSequence.hpp.

```
00172                                   {
00173                 std::string ris="";
00174                 List_Type::const_iterator it = seq->begin();
00175                 for (;it!=seq->end();++it)
00176                         ris=ris+(*it).asString()+" ";
00177                 return ris;
00178         }
```

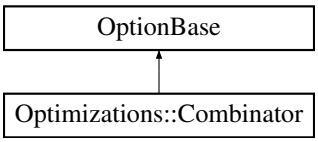
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMSequence.hpp
- /Users/esposito/Software/minerule/src/Algorithms/CCSMSequence.cpp

6.27 Optimizations::Combinator Class Reference

```
#include <optimizations.hpp>
```

Inheritance diagram for Optimizations::Combinator:



Public Member Functions

- Combinator ()
- virtual ~Combinator ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- float getTimeOutThreshold () const
- size_t getMaxDisjuncts () const
- size_t getMaxQueries () const
- size_t getMaxDistinctPredicates () const
- void setMaxDisjuncts (size_t m)
- void setMaxQueries (size_t m)
- void setMaxDistinctPredicates (size_t m)
- void setTimeOutThreshold (float to)

6.27.1 Detailed Description

Definition at line 24 of file optimizations.hpp.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Combinator()

```
Optimizations::Combinator::Combinator ( )  [inline]
```

Definition at line 31 of file optimizations.hpp.

```
00031                         :
00032           timeOut(4.0),
00033           maxDisjuncts(3),
00034           maxQueries(5),
00035           maxDistinctPredicates(10) {}
```

6.27.2.2 ~Combinator()

```
virtual Optimizations::Combinator::~Combinator ( )  [inline], [virtual]
```

Definition at line 36 of file optimizations.hpp.

```
00036 {};
```

6.27.3 Member Function Documentation

6.27.3.1 className()

```
virtual std::string Optimizations::Combinator::className ( ) const  [inline], [virtual]
```

Definition at line 39 of file optimizations.hpp.

```
00039                                           {
00040           return "combinator";
00041         }
```

6.27.3.2 getMaxDisjuncts()

```
size_t Optimizations::Combinator::getMaxDisjuncts ( ) const  [inline]
```

Definition at line 47 of file optimizations.hpp.

```
00047 {return maxDisjuncts;}
```

6.27.3.3 getMaxDistinctPredicates()

```
size_t Optimizations::Combinator::getMaxDistinctPredicates ( ) const  [inline]
```

Definition at line 49 of file optimizations.hpp.

```
00049 {return maxDistinctPredicates;}
```

6.27.3.4 getMaxQueries()

```
size_t Optimizations::Combinator::getMaxQueries ( ) const  [inline]
```

Definition at line 48 of file optimizations.hpp.

```
00048 {return maxQueries;}
```

6.27.3.5 getTimeOutThreshold()

```
float Optimizations::Combinator::getTimeOutThreshold ( ) const  [inline]
```

Definition at line 46 of file optimizations.hpp.

```
00046 { return timeOut; }
```

6.27.3.6 setMaxDisjuncts()

```
void Optimizations::Combinator::setMaxDisjuncts (
             size_t m )  [inline]
```

Definition at line 51 of file optimizations.hpp.

```
00051 { maxDisjuncts=m; }
```

6.27.3.7 setMaxDistinctPredicates()

```
void Optimizations::Combinator::setMaxDistinctPredicates (
            size_t m )  [inline]
```

Definition at line 53 of file optimizations.hpp.
```
00053 { maxDistinctPredicates=m; }
```

6.27.3.8 setMaxQueries()

```
void Optimizations::Combinator::setMaxQueries (
            size_t m )  [inline]
```

Definition at line 52 of file optimizations.hpp.
```
00052 { maxQueries=m; }
```

6.27.3.9 setOption()

```
virtual void Optimizations::Combinator::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.27.3.10 setTimeOutThreshold()

```
void Optimizations::Combinator::setTimeOutThreshold (
            float to )  [inline]
```

Definition at line 54 of file optimizations.hpp.
```
00054 { timeOut=to; }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/optimizations.hpp

6.28 minerule::Connection Class Reference

`#include <Connection.hpp>`

Public Types

- enum TableKind {
  RulesTable , HeadsTable , BodiesTable , SeqTable ,
  SeqElTable , ElemTable }

Public Member Functions

- Connection ()
- ~Connection ()
- void setBodyCardinalities (const MinMaxPair &rhs)
- void setHeadCardinalities (const MinMaxPair &rhs)
- void useMRDBConnection (mrdb::Connection *newConnection)
- mrdb::Connection * getMRDBConnection ()
- bool tableExists (const char *tableName)
- void deleteTable (const char *tableName)
- void deleteDestTables ()
- void createResultTables (const SourceRowMetaInfo &)
- void createResultTables ()
- void insert (const char *what)
- void insert (const ItemSet &body, const ItemSet &head, double support, double confidence, bool saveBody=true)
- void insert (int seqId, int seqEl, double support, int pos, bool saveSeqId)
- void delete_tmp_db ()
- void create_tmp_db (int sintax, const SourceRowAttrCollectionDescriptor &body, const SourceRowAttrCollectionDescriptor &head)
- void setOutTableName (const std::string &str)
- void init ()
- void finalize ()
- void finalize (bool b)
- std::string getTableName (TableKind kind) const

6.28.1 Detailed Description

Definition at line 25 of file Connection.hpp.

6.28.2 Member Enumeration Documentation

6.28.2.1 TableKind

```
enum minerule::Connection::TableKind
```

Enumerator

RulesTable	
HeadsTable	
BodiesTable	
SeqTable	
SeqElTable	
ElemTable	

Definition at line 27 of file Connection.hpp.

```
00027 { RulesTable, HeadsTable, BodiesTable, SeqTable, SeqElTable, ElemTable } TableKind;
```

6.28.3 Constructor & Destructor Documentation

6.28.3.1 Connection()

```
minerule::Connection::Connection ( )  [inline]
```

Definition at line 91 of file Connection.hpp.

```
00091                               : bodyCard(1,1000), headCard(1,1000)  /*algoOptions(NULL)*/ {
00092                           if (MineruleOptions::getSharedOptions().getMRDB().getCacheWrites())
00093                                   dbInserter = new CachedDBInserter(*this);
00094                           else dbInserter = new DirectDBInserter(*this);
00095                   }
```

6.28.3.2 ~Connection()

```
minerule::Connection::~Connection ( )  [inline]
```

Definition at line 97 of file Connection.hpp.

```
00097 {       delete dbInserter; }
```

6.28.4 Member Function Documentation

6.28.4.1 create_tmp_db()

```
void minerule::Connection::create_tmp_db (
            int sintax,
            const SourceRowAttrCollectionDescriptor & body,
            const SourceRowAttrCollectionDescriptor & head )
```

Definition at line 454 of file Connection.cpp.

```
00457           {
00458              mrdb::Statement* statement=connection->createStatement();
00459              std::string create;
00460
00461              delete_tmp_db();
00462
00463              switch (sintax)
00464              {
00465                  case 0 : create="CREATE TABLE tmp_Rule_Base(id int AUTO_INCREMENT PRIMARY KEY,level
       int, "
00466                                + body.getSQLDataDefinition() +");";
00467                           break;
00468                  case 1 : create=
00469                             "CREATE TABLE tmp_Rule_Ext(id int AUTO_INCREMENT PRIMARY KEY,level int,"
00470                             + body.getSQLDataDefinition() + ", id_head char(13));";
00471                           break;
00472              }
00473
00474              statement->execute(create);
00475
00476              if (sintax==1)
00477              {
00478                create="CREATE TABLE tmp_Rule_Head_Ext(id int AUTO_INCREMENT PRIMARY KEY,id_head
       char(13),level int," + head.getSQLDataDefinition()+");";
00479                statement->execute(create);
00480              }
00481              delete statement;
00482           }
```

6.28.4.2 createResultTables() [1/2]

```
void minerule::Connection::createResultTables ( )
```

Definition at line 121 of file Connection.cpp.

```
00121                                           {
00122
00123                 std::cout«"Calling mine sequence dest table function creation"«std::endl;
00124
00125                 if(tableExists(getTableName(SeqTable).c_str()) &&
      MineruleOptions::getSharedOptions().getSafety().getOverwriteHomonymMinerules())
00126                                 deleteDestTables();
00127
00128                 mrdb::Statement* statement=connection->createStatement();
00129                 std::string create, create_index;
00130
00131                 // Creating the seq table
00132                 if(!tableExists(getTableName(SeqTable).c_str())){
00133                         create=std::string("CREATE TABLE ")+getTableName(SeqTable)+" (ID int, supp
      float );";
00134                         create_index = " CREATE INDEX "+getTableName(SeqTable)+"_index ON " +
      getTableName(SeqTable) + " (id);";
00135                         statement->execute(create);
00136                         statement->execute(create_index);
00137                 }
00138
00139                 // Creating the sequences' elements table
00140                 if(!tableExists(getTableName(SeqElTable).c_str())){
00141                         create=std::string("CREATE TABLE ")+ getTableName(SeqElTable) + " (id int,idEl
      int,pos int )";
00142                         create_index = " CREATE INDEX "+getTableName(SeqElTable)+"_index ON " +
      getTableName(SeqElTable) + " (id);";
00143
00144                         statement->execute(create);
00145                         statement->execute(create_index);
00146                 }
00147
00148                 std::string seqInserterQuery = "INSERT INTO " + getTableName(SeqTable) + " VALUES (?,
      ?)";
00149                 dbInserter->setSeqInserter(connection->prepareStatement(seqInserterQuery));
00150
00151                 std::string seqElInserterQuery = "INSERT INTO " + getTableName(SeqElTable) + " VALUES
      (?, ?, ?)";
00152                 dbInserter->setSeqElInserter(connection->prepareStatement(seqElInserterQuery));
00153         }
```

6.28.4.3 createResultTables() [2/2]

```
void minerule::Connection::createResultTables (
             const SourceRowMetaInfo & srd )
```

Definition at line 74 of file Connection.cpp.

```
00074                                                                   {
00075
00076         std::cout«"Calling mine rule dest table function creation"«std::endl;
00077
00078         if(MineruleOptions::getSharedOptions().getSafety().getOverwriteHomonymMinerules())
00079                         deleteDestTables();
00080
00081                 mrdb::Statement* statement=connection->createStatement();
00082                 std::string create, create_index;
00083
00084                 // Creating the rules table
00085                 if(!tableExists(getTableName(RulesTable).c_str())){
00086                         create=std::string("CREATE TABLE ")+getTableName(RulesTable)+" (bodyId int,
      headId int, supp float, con float, cardBody int, cardHead int );";
00087                         statement->execute(create);
00088                 }
00089
00090                 // Creating the body elements table
00091                 if(!tableExists(getTableName(BodiesTable).c_str())){
00092                         create=std::string("CREATE TABLE ")+ getTableName(BodiesTable) + " (id int, "
      + srd.getBody().getSQLDataDefinition() +")";
00093                         create_index = " CREATE INDEX "+getTableName(BodiesTable)+"_index ON " +
      getTableName(BodiesTable) + " (id);";
```

```
00094
00095                         statement->execute(create);
00096                         statement->execute(create_index);
00097                 }
00098                 if(srd.getHead().getColumnsCount() > 0) {
00099                                 // Creating the head elements table
00100                                 if(!tableExists(getTableName(HeadsTable).c_str())){
00101                                         create=std::string("CREATE TABLE ")+ getTableName(HeadsTable)
      +  " (id int, " + srd.getHead().getSQLDataDefinition() +")";
00102
00103                                         create_index = " CREATE INDEX
      "+getTableName(HeadsTable)+"_index ON " + getTableName(HeadsTable) + " (id);";
00104
00105                                         statement->execute(create);
00106                                         statement->execute(create_index);
00107                                 }
00108
00109                                 std::string headInserterQuery = "INSERT INTO " +
      getTableName(HeadsTable) + " VALUES (?,"+ srd.getHead().questionMarks() +")";
00110
      dbInserter->setHeadInserter(connection->prepareStatement(headInserterQuery));
00111                 }
00112                 std::string bodyInserterQuery = "INSERT INTO " + getTableName(BodiesTable) + " VALUES
      (?," + srd.getBody().questionMarks() + ")";
00113                 dbInserter->setBodyInserter(connection->prepareStatement(bodyInserterQuery));
00114
00115                 delete statement;
00116
00117
00118         }
```

6.28.4.4 delete_tmp_db()

```
void minerule::Connection::delete_tmp_db ( )
```

Definition at line 484 of file Connection.cpp.

```
00485         {
00486
00487          deleteTable("tmp_Rule_Base");
00488          deleteTable("tmp_Rule_Ext");
00489          deleteTable("tmp_Rule_Head_Ext");
00490
00491         }
```

6.28.4.5 deleteDestTables()

```
void minerule::Connection::deleteDestTables ( )
```

Definition at line 54 of file Connection.cpp.

```
00054                                           {
00055          deleteTable(getTableName(RulesTable).c_str());
00056          deleteTable(getTableName(HeadsTable).c_str());
00057          deleteTable(getTableName(BodiesTable).c_str());
00058
00059       deleteTable(getTableName(SeqTable).c_str());
00060       deleteTable(getTableName(SeqElTable).c_str());
00061       deleteTable(getTableName(ElemTable).c_str());
00062    }
```

6.28.4.6 deleteTable()

```
void minerule::Connection::deleteTable (
            const char * tableName )
```

Definition at line 64 of file Connection.cpp.

```
00065     {
00066             mrdb::Statement* stmt=connection->createStatement();
00067             MRLog() « "Dropping table " « tableName « std::endl;
00068             stmt->execute(std::string("DROP TABLE IF EXISTS ")+tableName);
00069             delete stmt;
00070
00071     }
```

6.28.4.7 finalize() [1/2]

```
void minerule::Connection::finalize ( )  [inline]
```

Definition at line 135 of file Connection.hpp.

```
00135 { dbInserter->finalize(); }
```

6.28.4.8 finalize() [2/2]

```
void minerule::Connection::finalize (
            bool b )  [inline]
```

Definition at line 136 of file Connection.hpp.

```
00136 { dbInserter->finalize(b); }
```

6.28.4.9 getMRDBConnection()

```
mrdb::Connection * minerule::Connection::getMRDBConnection ( )  [inline]
```

Definition at line 103 of file Connection.hpp.

```
00103 { return connection; }
```

6.28.4.10 getTableName()

```
std::string minerule::Connection::getTableName (
            TableKind kind ) const
```

Definition at line 16 of file Connection.cpp.

```
00016                                                                   {
00017                 switch(kind) {
00018                         case RulesTable:
00019                                 return outTableName;
00020                                 break;
00021                         case HeadsTable:
00022                                 return outTableName + "_head_elems";
00023                                 break;
00024                         case BodiesTable:
00025                                 return outTableName + "_body_elems";
00026                                 break;
00027                         case SeqTable:
00028           return outTableName + "_Seq_support";
00029           break;
00030       case SeqElTable:
00031           return outTableName + "_Seq";
00032           break;
00033       case ElemTable:
00034           return outTableName + "_Seq1";
00035           break;
00036                         default:
00037                                 throw MineruleException(MR_ERROR_INTERNAL, "Unknown TableKind -- this
      is a bug, please report it!");
00038                 }
00039         }
```

6.28.4.11 init()

```
void minerule::Connection::init ( )  [inline]
```

Definition at line 131 of file Connection.hpp.

```
00131                           {
00132                         dbInserter->init();
00133                 }
```

6.28.4.12 insert() [1/3]

```
void minerule::Connection::insert (
            const char * what )
```

Definition at line 447 of file Connection.cpp.

```
00447                                           {
00448             static mrdb::Statement* statement=connection->createStatement();
00449             statement->execute(what);
00450             delete statement;
00451           }
```

6.28.4.13 insert() [2/3]

```
void minerule::Connection::insert (
            const ItemSet & body,
            const ItemSet & head,
            double support,
            double confidence,
            bool saveBody = true )  [inline]
```

Definition at line 116 of file Connection.hpp.

```
00116                                {
00117                     dbInserter->insert(body, head, support, confidence, saveBody);
00118           }
```

6.28.4.14 insert() [3/3]

```
void minerule::Connection::insert (
            int seqId,
            int seqEl,
            double support,
            int pos,
            bool saveSeqId )  [inline]
```

Definition at line 120 of file Connection.hpp.

```
00120                                                                     {
00121        dbInserter->insert(seqId, seqEl, support, pos, saveSeqId);
00122     }
```

6.28.4.15 setBodyCardinalities()

```
void minerule::Connection::setBodyCardinalities (
            const MinMaxPair & rhs )  [inline]
```

Definition at line 99 of file Connection.hpp.

```
00099 { bodyCard=rhs; }
```

6.28.4.16 setHeadCardinalities()

```
void minerule::Connection::setHeadCardinalities (
            const MinMaxPair & rhs )  [inline]
```

Definition at line 100 of file Connection.hpp.

```
00100 { headCard=rhs; }
```

6.28.4.17 setOutTableName()

```
void minerule::Connection::setOutTableName (
            const std::string & str )  [inline]
```

Definition at line 127 of file Connection.hpp.

```
00127                                                       {
00128                         outTableName=str;
00129                 }
```

6.28.4.18 tableExists()

```
bool minerule::Connection::tableExists (
            const char * tableName )
```

Definition at line 43 of file Connection.cpp.

```
00043                                                       {
00044                         std::string table(tableName, strlen(tableName));
00045                         std::string query = "SELECT relname FROM pg_class WHERE upper(relname) =
      upper('"+table+"');";
00046                         mrdb::Statement* stmt=connection->createStatement();
00047                         mrdb::ResultSet* rs = stmt->executeQuery(query);
00048                         bool result = rs->next();
00049                         delete rs;
00050                         delete stmt;
00051                         return result;
00052         }
```

6.28.4.19 useMRDBConnection()

```
void minerule::Connection::useMRDBConnection (
            mrdb::Connection * newConnection )
```

Definition at line 12 of file Connection.cpp.

```
00012                                                                   {
00013           connection = newConnection;
00014         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/Connection.hpp
- /Users/esposito/Software/minerule/src/Database/Connection.cpp

6.29 mrdb::Connection Class Reference

`#include <Connection.hpp>`

Public Member Functions

- virtual ~Connection ()
- virtual Statement * createStatement ()=0
- virtual PreparedStatement * prepareStatement (const std::string &sql)=0
- virtual DatabaseMetaData * getMetaData ()=0

6.29.1 Detailed Description

Definition at line 12 of file Connection.hpp.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 ~Connection()

```
virtual mrdb::Connection::~Connection ( )  [inline], [virtual]
```

Definition at line 14 of file Connection.hpp.
```
00014 {}
```

6.29.3 Member Function Documentation

6.29.3.1 createStatement()

```
virtual Statement * mrdb::Connection::createStatement ( )  [pure virtual]
```

Returns

a newly created Statement.

Exceptions

a	MineruleException with an error code MR_ERROR_DATABASE_ERROR in case the creation fails. @memory the returned object needs to be dealloced by the user once no longer needed.

6.29.3.2 getMetaData()

```
virtual DatabaseMetaData * mrdb::Connection::getMetaData ( )  [pure virtual]
```

Returns

the database meta data for this connection The returned object is owned by this class. The user must not release it. It will be freed along with this connection.

6.29.3.3 prepareStatement()

```
virtual PreparedStatement * mrdb::Connection::prepareStatement (
            const std::string & sql )  [pure virtual]
```

Parameters

sql	the sql command to be prepared. Parameters need to be specified using question marks, e.g., SELECT * FROM table WHERE id=?

Returns

a newly created PreparedStatement.

Exceptions

a	MineruleException with an error code MR_ERROR_DATABASE_ERROR in case the creation fails. @memory the returned object needs to be dealloced by the user once no longer needed.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/Connection.hpp

6.30 minerule::Constraints Struct Reference

`#include <STSMiner.hpp>`

Public Member Functions

- void clear ()
- Constraints ()

Data Fields

- std::vector< BitString > bem
- std::vector< std::vector< std::pair< int, int > > > mid_counters

6.30.1 Detailed Description

Struct to manage context_dependent BEM constraints

Definition at line 32 of file STSMiner.hpp.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 Constraints()

```
minerule::Constraints::Constraints ( )  [inline]
```

Definition at line 41 of file STSMiner.hpp.

```
00041 {}
```

6.30.3 Member Function Documentation

6.30.3.1 clear()

```
void minerule::Constraints::clear ( )  [inline]
```

Definition at line 36 of file STSMiner.hpp.

```
00036                 {
00037         bem.clear();
00038         mid_counters.clear();
00039     }
```

6.30.4 Field Documentation

6.30.4.1 bem

```
std::vector<BitString> minerule::Constraints::bem
```

Definition at line 33 of file STSMiner.hpp.

6.30.4.2 mid_counters

```
std::vector<std::vector<std::pair<int,int> > > minerule::Constraints::mid_counters
```

Definition at line 34 of file STSMiner.hpp.

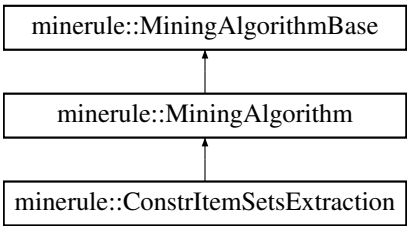
The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp

6.31 minerule::ConstrItemSetsExtraction Class Reference

`#include <ConstrItemSetsExtraction.hpp>`

Inheritance diagram for minerule::ConstrItemSetsExtraction:



Public Member Functions

- ConstrItemSetsExtraction (const OptimizedMinerule &mr)
- virtual ~ConstrItemSetsExtraction ()
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual void initialize ()
- virtual void execute ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- const OptimizedMinerule & minerule

6.31.1 Detailed Description

Definition at line 25 of file ConstrItemSetsExtraction.hpp.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 ConstrItemSetsExtraction()

```
minerule::ConstrItemSetsExtraction::ConstrItemSetsExtraction (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 36 of file ConstrItemSetsExtraction.hpp.

```
00036                                                    :
00037       MiningAlgorithm(mr) {}
```

6.31.2.2 ~ConstrItemSetsExtraction()

```
virtual minerule::ConstrItemSetsExtraction::~ConstrItemSetsExtraction ( )  [inline], [virtual]
```

Definition at line 39 of file ConstrItemSetsExtraction.hpp.

```
00039 {}
```

6.31.3 Member Function Documentation

6.31.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.31.3.2 canHandleMinerule()

```
virtual bool minerule::ConstrItemSetsExtraction::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 43 of file ConstrItemSetsExtraction.hpp.

```
00043                                          {
00044       return
00045         !minerule.getParsedMinerule().hasCrossConditions() &&
00046         !minerule.getParsedMinerule().requiresClusters() &&
00047         !minerule.getParsedMinerule().hasDisjuctionsInMC();
00048     }
```

6.31.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                 {
00094                       initialize();
00095                       mineRules();
00096                 }
```

6.31.3.4 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                   {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.31.3.5 mineRules()

```
void minerule::ConstrItemSetsExtraction::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 55 of file ConstrItemSetsExtraction.cpp.

```
00055                                    {
00056     MRLogPush("Starting ConstrItemSetsExtraction mining algorithm...");
00057
00058     MRLog() « "Preparing data sources..." « std::endl;
00059     prepareData();
00060
00061     float support = options.getSupport();
00062     int maxBody = options.getBodyCardinalities().getMax();
00063
00064     ItemType gid1;
00065     BodyMap bodyMap(connection,1);
00066
00067     int howManyGroups = 0;
00068     int totalGroups = sourceTable->getTotGroups();
00069     int howManyRows = 0;
00070
00071     while (!bodyIterator.isAfterLast()) {
00072                 ItemType gid = bodyIterator->getGroup();
00073                 Transaction t1, t2;
00074
00075                 t1.loadBody(gid,bodyIterator);
00076                 howManyRows += bodyMap.add(t1,howManyGroups);
00077                 howManyGroups++;
00078     }
00079     MRLog() « "Total rows: " « howManyRows « std::endl;
00080     MRLog() « "Total groups: " « totalGroups « std::endl;
00081     MRLogPop();
00082
00083     MRLogPush("Starting itemset extraction...");
00084
00085     bodyMap.updateCount();
```

```
00086     MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00087     bodyMap.pruneMap(support*totalGroups,true);
00088     bodyMap.setTotalGroups(totalGroups);
00089     std::vector<BodyMap::iterator> v;
00090     std::multimap<int, BodyMap::iterator> temp;
00091     for (BodyMap::iterator i = bodyMap.begin(); i!=bodyMap.end(); i++)
00092         if (!i->second.done) temp.insert(std::pair<int, BodyMap::iterator>(i->second.count(),i));
00093     for (std::multimap<int, BodyMap::iterator>::iterator i = temp.begin(); i!=temp.end(); i++)
00094         v.insert(v.end(),i->second);
00095     MRLog() « "Total bodies after pruning: " « v.size() « std::endl;
00096
00097     NewRule r;
00098     NewRuleSet rs;
00099     int nrules = bodyMap.generateStartItemSets(rs,r,v,maxBody,support*totalGroups,-1);
00100     MRLog() « "After extracting itemsets: " « nrules « std::endl;
00101     MRLogPop();
00102
00103     connection.finalize();
00104   }
```

6.31.3.6 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.31.3.7 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::MiningAlgorithmBase::sourceTableRequirements ( )
const  [inline], [virtual], [inherited]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CCSMiner, minerule::FSMiner, and minerule::STSMiner.

Definition at line 43 of file MiningAlgorithmBase.hpp.

```
00043                                                                    {
00044                         return SourceTableRequirements();
00045                 };
```

6.31.4 Field Documentation

6.31.4.1 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

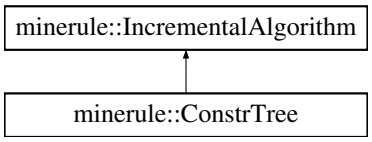
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrItemSetsExtraction.hpp
- /Users/esposito/Software/minerule/src/Algorithms/ConstrItemSetsExtraction.cpp

6.32 minerule::ConstrTree Class Reference

`#include <ConstrTree.hpp>`

Inheritance diagram for minerule::ConstrTree:



Public Member Functions

- ConstrTree (const OptimizedMinerule &mr)
- virtual ~ConstrTree ()
- virtual void execute ()

Static Public Member Functions

- static IncrementalAlgorithm * newIncrementalAlgorithm (const OptimizedMinerule &mr)

Protected Member Functions

- void adjustSuppRSet ()
- void adjustSuppMIndex ()
- void insertRulesInStructure ()
- void adjustSupp ()
- void prepareData ()
- size_t buildAttrStr (const ParsedMinerule::AttrVector &attr, size_t startIndex, std::string &attrStr, std::vector< int > &des) const
- std::string buildQry (const std::string &groupAttrStr, const std::string &attrStr, const std::string &constraints) const
- Body * getRoot ()

Protected Attributes

- Body * root
- size_t ngroups
- mrdb::ResultSet * rb2
- mrdb::ResultSet * rh2
- SourceRowColumnIds bodyDes
- SourceRowColumnIds headDes
- mrdb::Statement * stateb2
- mrdb::Statement * stateh2
- const OptimizedMinerule * minerule

6.32.1 Detailed Description

This class implements <COMPLETE>

Definition at line 32 of file ConstrTree.hpp.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 ConstrTree()

```
minerule::ConstrTree::ConstrTree (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 60 of file ConstrTree.hpp.

```
00060 : IncrementalAlgorithm(mr), root(new Body()), ngroups(0) /*,mb2(NULL), mh2(NULL)*/, rb2(NULL),
       rh2(NULL) { }
```

6.32.2.2 ~ConstrTree()

```
virtual minerule::ConstrTree::~ConstrTree ( )  [inline], [virtual]
```

Definition at line 62 of file ConstrTree.hpp.

```
00062                       {
00063     // Trashing the trashable
00064     if(rh2!=NULL) delete rh2;
00065     if(stateh2!=NULL) delete stateh2;
00066     if(rb2!=NULL) delete rb2;
00067     if(stateb2!=NULL) delete stateb2;
00068 
00069     rb2=rh2=NULL;
00070     stateh2=stateb2=NULL;
00071 
00072         delete root;
00073   }
```

6.32.3 Member Function Documentation

6.32.3.1 adjustSupp()

```
void minerule::ConstrTree::adjustSupp ( )  [protected]
```

Definition at line 123 of file ConstrTree.cpp.

```
00123                               {
00124     MRLogPusher _("Starting the mining algorithm...");
00125 
00126     MRLog() « "Reading previous result and preparing data structures..." « std::endl;
00127     insertRulesInStructure();
00128 
00129     Connection connection;
00130     connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00131     connection.useMRDBConnection(
00132        MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00133     connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule->getParsedMinerule()));
00134 
00135     // if (mb2!=NULL  &&  mh2!=NULL) {
00136     //    adjustSuppMIndex();
00137     //    std::vector<ItemType> body;
00138     //    getRoot()->extractRules(body,
00139     //                            minerule->getParsedMinerule().sup,
00140     //                            minerule->getParsedMinerule().conf,
00141     //                            ngroups,
00142     //                            &connection);
00143     //  }
00144     //  else {
00145       if (rb2!=NULL  &&  rh2!=NULL) {
00146         MRLog() « "Evaluating constraints and adjusting the support counts in the data structure..." «
      std::endl;
00147         adjustSuppRSet();
00148 std::vector<ItemType> body;
00149 
00150         MRLog() « "Pruning rules and writing results..." « std::endl;
00151         getRoot()->extractRules(body,
00152                                 minerule->getParsedMinerule().sup,
00153                                 minerule->getParsedMinerule().conf,
00154                                 ngroups,
00155                                 &connection);
00156       } else
00157         throw MineruleException (MR_ERROR_INTERNAL,
00158                                  " cannot create rules ");
00159     // }
00160   }
```

6.32.3.2 adjustSuppMIndex()

```
void minerule::ConstrTree::adjustSuppMIndex ( )  [protected]
```

6.32.3.3 adjustSuppRSet()

```
void minerule::ConstrTree::adjustSuppRSet ( )  [protected]
```

Definition at line 78 of file ConstrTree.cpp.

```
00078                                  {
00079     ItemType gid;
00080     ItemType gidb;
00081     ItemType item;
00082 
00083     bool bodynotend=rb2->next();
00084     bool headnotend=rh2->next();
00085     while( bodynotend ) {
00086 
00087       SourceRow curRowb(rb2, bodyDes);
00088       //std::string g=mb2->getCurrentGID();
00089       gidb=curRowb.getGroup();
00090 
```

```
00091       ItemSet* body=new ItemSet();
00092       //cout«"in adjustSupp:"«std::endl;
00093
00094       while(bodynotend && ItemType(curRowb.getGroup())==gidb){
00095         body->push_back(curRowb.getBody());
00096         if((bodynotend=rb2->next())) {
00097           curRowb.init(rb2, bodyDes);
00098         }
00099       }
00100
00101       if(headnotend) {
00102         SourceRow curRowh(rh2, headDes);
00103         if (gidb!=curRowh.getGroup())
00104           this->root->findBodiesInTree(body);
00105         else{
00106           ItemSet* head=new ItemSet();
00107           while(headnotend && ItemType(curRowh.getGroup())==gidb ){
00108             head->push_back(curRowh.getHead());
00109             if ((headnotend=rh2->next())) {
00110               curRowh.init(rh2, headDes);
00111             }
00112           }
00113
00114           this->root->findRulesInTree(body,head);
00115           delete head;
00116         }
00117       }
00118       delete body;
00119     }
00120   }
```

6.32.3.4 buildAttrStr()

```
size_t minerule::ConstrTree::buildAttrStr (
            const ParsedMinerule::AttrVector & attr,
            size_t startIndex,
            std::string & attrStr,
            std::vector< int > & des ) const  [protected]
```

Definition at line 163 of file ConstrTree.cpp.

```
00166                                                                                                    {
00167     ParsedMinerule::AttrVector::const_iterator it = attr.begin();
00168     for( ; it!=attr.end(); it++ ) {
00169       if(it!=attr.begin()) {
00170         attrStr+=",";
00171       }
00172
00173       attrStr+=*it;
00174       des.push_back(++startIndex);
00175     }
00176
00177     return startIndex;
00178   }
```

6.32.3.5 buildQry()

```
std::string minerule::ConstrTree::buildQry (
            const std::string & groupAttrStr,
            const std::string & attrStr,
            const std::string & constraints ) const  [protected]
```

Definition at line 183 of file ConstrTree.cpp.

```
00185                                                                                                    {
00186
00187     return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00188                        "FROM "+minerule->getParsedMinerule().tab_source+" "+
00189                        (constraints.size()>0 ?
00190                        "WHERE "+constraints+" " :
00191                        "")
00192                        +"ORDER BY "+groupAttrStr);
00193   }
```

6.32.3.6 execute()

```
void minerule::ConstrTree::execute ( )  [virtual]
```

Implements minerule::IncrementalAlgorithm.

Definition at line 236 of file ConstrTree.cpp.

```
00237     {
00238     assert( minerule->getOptimizationInfo().relationship == OptimizedMinerule::Dominance );
00239     assert( minerule->getParsedMinerule().mc!=NULL &&
00240             minerule->getParsedMinerule().mc->next==NULL);
00241
00242     MRLogPusher _("This is the Context Dependent Constructive Mining Algorithm...");
00243
00244         prepareData();
00245     adjustSupp();
00246   }
```

6.32.3.7 getRoot()

```
Body * minerule::ConstrTree::getRoot ( )  [inline], [protected]
```

Definition at line 58 of file ConstrTree.hpp.

```
00058 {return root;}
```

6.32.3.8 insertRulesInStructure()

```
void minerule::ConstrTree::insertRulesInStructure ( )  [protected]
```

Definition at line 28 of file ConstrTree.cpp.

```
00028                                       {
00029     QueryResult::Iterator qit;
00030     //il primo paramentro e' il nome della tabella dei ris della vecchia query
00031     OptimizerCatalogue::getMRQueryResultIterator(minerule->getOptimizationInfo().minerule.tab_result,
      qit, -1, 0.0);
00032     Head* newhead;
00033     while( qit.next()) {
00034       Rule r;
00035       //crea body e head che sono due vettori ItemSet
00036       //ovvero due vettori di ItemType
00037       //ogni ItemType contiene un puntatore ad un SourceRowElement
00038       qit.getRule(r);
00039       //insertRuleInStructure(r);
00040       //body e head sono due puntatori a ItemSet
00041       newhead=root->insertItemSetB(r.getBody(),0.0);
00042       newhead->insertItemSetH(r.getHead(),0.0);
00043     }
00044
00045   }
```

6.32.3.9 newIncrementalAlgorithm()

```
IncrementalAlgorithm * minerule::IncrementalAlgorithm::newIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.cpp.

```
00025                                                                           {
00026     // if mr has only ItemDependent constraints
00027                 MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028
00029                 if( mr.hasIDConstraints() ) {
00030                         MRLog() « "The minerule is item dependent!" « std::endl;
00031                         if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                 return new ResultCombinator(mr);
00033                         else
00034                                 return new IDIncrementalAlgorithm(mr);
00035                 }
00036
00037                 MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038
00039                 if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040
00041                         IncrementalAlgorithm* incrAlgo = NULL;
00042
00043                         switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                 case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                         incrAlgo = new ConstrTree(mr);
00046                                         break;
00047                                 case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                         incrAlgo = new DestrTree(mr);
00049                                         break;
00050                                 case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                         incrAlgo = new ConstrTree(mr);
00052                                         break;
00053                         }
00054
00055                         return incrAlgo;
00056                 } else {
00057                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                 « " to the system yet." « std::endl;
00059                         return NULL;
00060                 }
00061         }
```

6.32.3.10 prepareData()

```
void minerule::ConstrTree::prepareData ( )  [protected]
```

Definition at line 196 of file ConstrTree.cpp.

```
00196                                 {
00197       MRLogPusher _("Building source information");
00198
00199           MRLog() « "Separating the constraints in the HEAD and BODY parts..."«std::endl;
00200           std::string bodyConstraints;
00201           std::string headConstraints;
00202           HeadBodyPredicatesSeparator::separate(minerule->getParsedMinerule().mc->l_and,
00203                   bodyConstraints,
00204                           headConstraints);
00205
00206           MRLog() « "Building db queries" « std::endl;
00207           size_t index;
00208           std::string groupAttr;
00209           std::string bodyAttr;
00210           std::string headAttr;
00211           index=buildAttrStr(minerule->getParsedMinerule().ga, 0, groupAttr, bodyDes.groupElems );
00212
00213           headDes.groupElems=bodyDes.groupElems;
00214
00215           buildAttrStr(minerule->getParsedMinerule().ba, index, bodyAttr, bodyDes.bodyElems);
00216           buildAttrStr(minerule->getParsedMinerule().ha, index, headAttr, headDes.headElems);
00217
00218           std::string bodyQry = buildQry( groupAttr, bodyAttr, bodyConstraints);
00219
```

```
00220           std::string headQry = buildQry( groupAttr, headAttr, headConstraints);
00221
00222           MRLog() « "Body query" « bodyQry « std::endl;
00223           MRLog() « "Head query" « headQry « std::endl;
00224           MRLog() « "Executing queries" « std::endl;
00225           mrdb::Connection* con = MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00226
00227           stateb2 = con->createStatement();
00228           rb2 = stateb2->executeQuery(bodyQry);
00229
00230
00231           stateh2 = con->createStatement();
00232           rh2 = stateh2->executeQuery(headQry);
00233           ngroups = PrepareDataUtils::evaluateTotGroups(minerule->getParsedMinerule());
00234   }
```

6.32.4 Field Documentation

6.32.4.1 bodyDes

```
SourceRowColumnIds minerule::ConstrTree::bodyDes  [protected]
```

Definition at line 38 of file ConstrTree.hpp.

6.32.4.2 headDes

```
SourceRowColumnIds minerule::ConstrTree::headDes  [protected]
```

Definition at line 39 of file ConstrTree.hpp.

6.32.4.3 minerule

```
const OptimizedMinerule* minerule::IncrementalAlgorithm::minerule  [protected], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.hpp.

6.32.4.4 ngroups

```
size_t minerule::ConstrTree::ngroups  [protected]
```

Definition at line 35 of file ConstrTree.hpp.

6.32.4.5 rb2

```
mrdb::ResultSet* minerule::ConstrTree::rb2  [protected]
```

Definition at line 36 of file ConstrTree.hpp.

6.32.4.6 rh2

```
mrdb::ResultSet* minerule::ConstrTree::rh2  [protected]
```

Definition at line 37 of file ConstrTree.hpp.

6.32.4.7 root

```
Body* minerule::ConstrTree::root  [protected]
```

Definition at line 34 of file ConstrTree.hpp.

6.32.4.8 stateb2

```
mrdb::Statement* minerule::ConstrTree::stateb2  [protected]
```

Definition at line 40 of file ConstrTree.hpp.

6.32.4.9 stateh2

```
mrdb::Statement* minerule::ConstrTree::stateh2  [protected]
```

Definition at line 41 of file ConstrTree.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrTree.hpp
- /Users/esposito/Software/minerule/src/Algorithms/ConstrTree.cpp

6.33 minerule::Converter Class Reference

```
#include <Converter.hpp>
```

Public Member Functions

- Converter (const char *val)
- Converter (std::string val)
- Converter (double val)
- Converter (int val)
- Converter (long val)
- Converter (unsigned int val)
- Converter (unsigned long val)
- Converter (bool val)
- std::string toString () const
- double toDouble (unsigned int flags=ALLOW_QUOTES_IN_NUMBERS) const
- long toLong (unsigned int flags=ALLOW_QUOTES_IN_NUMBERS) const
- bool toBool () const
- bool isNumber () const

6.33.1 Detailed Description

Definition at line 28 of file Converter.hpp.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 Converter() [1/8]

```
minerule::Converter::Converter (
            const char * val )  [inline], [explicit]
```

Definition at line 33 of file Converter.hpp.

```
00033 : value(val){};
```

6.33.2.2 Converter() [2/8]

```
minerule::Converter::Converter (
            std::string val )  [inline], [explicit]
```

Definition at line 34 of file Converter.hpp.

```
00034 : value(val){};
```

6.33.2.3 Converter() [3/8]

```
minerule::Converter::Converter (
            double val )  [inline], [explicit]
```

Definition at line 35 of file Converter.hpp.

```
00035                                {
00036     std::stringstream ss;
00037     ss « val; // « std::ends;
00038     value = ss.str();
00039   }
```

6.33.2.4 Converter() [4/8]

```
minerule::Converter::Converter (
            int val )  [inline], [explicit]
```

Definition at line 41 of file Converter.hpp.

```
00041                              {
00042     std::stringstream ss;
00043     ss « val;
00044     value = ss.str();
00045   }
```

6.33.2.5 Converter() [5/8]

```
minerule::Converter::Converter (
            long val )  [inline], [explicit]
```

Definition at line 47 of file Converter.hpp.

```
00047                               {
00048     std::stringstream ss;
00049     ss « val;
00050     value = ss.str();
00051   }
```

6.33.2.6 Converter() [6/8]

```
minerule::Converter::Converter (
            unsigned int val )  [inline], [explicit]
```

Definition at line 53 of file Converter.hpp.

```
00053                                       {
00054     std::stringstream ss;
00055     ss « val; // « std::ends;
00056     value = ss.str();
00057   }
```

6.33.2.7 Converter() [7/8]

```
minerule::Converter::Converter (
            unsigned long val )  [inline], [explicit]
```

Definition at line 59 of file Converter.hpp.

```
00059                                       {
00060     std::stringstream ss;
00061     ss « val; // « std::ends;
00062     value = ss.str();
00063   }
```

6.33.2.8 Converter() [8/8]

```
minerule::Converter::Converter (
            bool val )  [inline], [explicit]
```

Definition at line 65 of file Converter.hpp.

```
00065                                {
00066     if (val)
00067       value = "True";
00068     else
00069       value = "False";
00070   }
```

6.33.3 Member Function Documentation

6.33.3.1 isNumber()

```
bool minerule::Converter::isNumber ( ) const  [inline]
```

Definition at line 144 of file Converter.hpp.

```
00144                          {
00145     try {
00146       toDouble();
00147     } catch (MineruleException &e) {
00148       return false;
00149     }
00150
00151     return true;
00152   }
```

6.33.3.2 toBool()

```
bool minerule::Converter::toBool ( ) const  [inline]
```

Definition at line 130 of file Converter.hpp.

```
00130                         {
00131     if (value == "True")
00132       return true;
00133     else if (value == "False")
00134       return false;
00135     else if (isNumber()) {
00136       return toDouble() > 0;
00137     } else
00138       throw MineruleException(MR_ERROR_INTERNAL,
00139                               "Converting error while converting string:"
00140                               "'" +
00141                                   value + "' to a bool value");
00142   }
```

6.33.3.3 toDouble()

```
double minerule::Converter::toDouble (
            unsigned int flags = ALLOW_QUOTES_IN_NUMBERS ) const  [inline]
```

Definition at line 74 of file Converter.hpp.

```
00075       {
00076     char *endPtr = NULL;
00077     double result;
00078     const char *startChar;
00079     char endChar;
00080     errno = 0;
00081
00082     if ((flags & ALLOW_QUOTES_IN_NUMBERS) && value.length() > 1 &&
00083         value[0] == '\"' && value[value.length() - 1] == '\") {
00084       startChar = &value.c_str()[1];
00085       endChar = '\";
00086     } else {
00087       startChar = value.c_str();
00088       endChar = '\0';
00089     }
00090
00091     result = strtod(startChar, &endPtr);
00092     if (*endPtr != endChar || errno == ERANGE || errno == EINVAL)
00093       throw MineruleException(MR_ERROR_INTERNAL,
00094                               "Convertion error while converting string:" +
00095                                   value + " to a Double value");
00096     return result;
00097   }
```

6.33.3.4 toLong()

```
long minerule::Converter::toLong (
            unsigned int flags = ALLOW_QUOTES_IN_NUMBERS ) const  [inline]
```

Definition at line 99 of file Converter.hpp.

```
00099                                                                  {
00100     char *endPtr = NULL;
00101     errno = 0;
00102     long result;
00103     const char *startChar;
00104     char endChar;
00105
00106     if ((flags & ALLOW_QUOTES_IN_NUMBERS) && value.length() > 1 &&
00107         value[0] == '\"' && value[value.length() - 1] == '\") {
00108       startChar = &value.c_str()[1];
00109       endChar = '\";
00110     } else {
00111       startChar = value.c_str();
00112       endChar = '\0';
00113     }
00114
00115     result = strtol(startChar, &endPtr, 10);
00116     if (*endPtr != endChar || errno == ERANGE || errno == EINVAL) {
00117       std::stringstream ss;
00118       ss « "Convertion error while converting string:" « value
00119          « " to a long value. lenght:" « value.length()
00120          « " strlen:" « strlen(value.c_str());
00121       if (errno != 0)
00122         ss « " errno string:" « strerror(errno);
00123
00124       throw MineruleException(MR_ERROR_INTERNAL, ss.str());
00125     }
00126
00127     return result;
00128   }
```

6.33.3.5 toString()

```
std::string minerule::Converter::toString ( ) const  [inline]
```

Definition at line 72 of file Converter.hpp.

```
00072 { return value; }
```

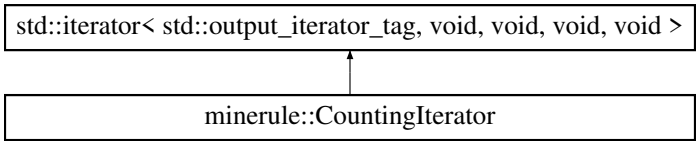
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/Converter.hpp

6.34 minerule::CountingIterator Class Reference

```
#include <PredicateUtils.hpp>
```

Inheritance diagram for minerule::CountingIterator:



Public Member Functions

- CountingIterator (size_t &counter, list_AND_node *&l)
- CountingIterator (const CountingIterator &c)
- CountingIterator & operator= (SimplePredicate *const &pred)
- CountingIterator & operator* ()
- CountingIterator & operator++ (int)
- CountingIterator & operator++ ()

Static Public Member Functions

- static list_AND_node * new_list_AND_node (const char *v1, const char *op, const char *v2, list_AND_node *oldList)
- static void delete_list_AND_node (list_AND_node *&)

6.34.1 Detailed Description

A simple output_iterator which does nothing but using the metod setVarId in SimplePredicate to assign to each predicate a unique identifier. The unique identifier is obtained by counting the number of times that the asssignment operator is called. Note the strange way the curNum is treated! This is as hack which is needed since the set_union function takes the output_iterator by value. Then in order to allow a proper counting we need to use a reference to an external variable (in this way all object created by copying this object will share the same reference).

Definition at line 45 of file PredicateUtils.hpp.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 CountingIterator() [1/2]

```
minerule::CountingIterator::CountingIterator (
            size_t & counter,
            list_AND_node *& l )  [inline]
```

Definition at line 56 of file PredicateUtils.hpp.

```
00056                                                   : curNum(counter), l_and(l) {
00057     }
```

6.34.2.2 CountingIterator() [2/2]

```
minerule::CountingIterator::CountingIterator (
            const CountingIterator & c )  [inline]
```

Definition at line 59 of file PredicateUtils.hpp.

```
00059 : curNum(c.curNum), l_and(c.l_and) {}
```

6.34.3 Member Function Documentation

6.34.3.1 delete_list_AND_node()

```
void minerule::CountingIterator::delete_list_AND_node (
            list_AND_node *& l )  [static]
```

Definition at line 40 of file PredicateUtils.cpp.

```
00040                                                   {
00041     while( l!=NULL ) {
00042       list_AND_node* tmp = l->next;;
00043
00044       delete l->sp;
00045       delete l;
00046
00047       l=tmp;
00048     }
00049   }
```

6.34.3.2 new_list_AND_node()

```
list_AND_node * minerule::CountingIterator::new_list_AND_node (
            const char * v1,
            const char * op,
            const char * v2,
            list_AND_node * oldList )  [static]
```

Definition at line 27 of file PredicateUtils.cpp.

```
00030                                                                                {
00031     list_AND_node* tmp = new list_AND_node;
00032     tmp->sp = new simple_pred;
00033     tmp->sp->val1 = const_cast<char*>(v1);
00034     tmp->sp->op = const_cast<char*>(op);
00035     tmp->sp->val2 = const_cast<char*>(v2);
00036     tmp->next = oldList;
00037     return tmp;
00038   }
```

6.34.3.3 operator*()

```
CountingIterator & minerule::CountingIterator::operator* ( )  [inline]
```

Definition at line 71 of file PredicateUtils.hpp.

```
00071                                  {
00072       return *this;
00073     }
```

6.34.3.4 operator++() [1/2]

```
CountingIterator & minerule::CountingIterator::operator++ ( )  [inline]
```

Definition at line 80 of file PredicateUtils.hpp.

```
00080                                   {
00081       curNum++;
00082       return *this;
00083     }
```

6.34.3.5 operator++() [2/2]

```
CountingIterator & minerule::CountingIterator::operator++ (
            int  )  [inline]
```

Definition at line 75 of file PredicateUtils.hpp.

```
00075                                   {
00076       curNum++;
00077       return *this;
00078     }
```

6.34.3.6 operator=()

```
CountingIterator & minerule::CountingIterator::operator= (
            SimplePredicate *const & pred )  [inline]
```

Definition at line 61 of file PredicateUtils.hpp.

```
00061                                                                {
00062       pred->setVarId(curNum);
00063       list_AND_node* tmp = new_list_AND_node( pred->getVal1().c_str(),
00064                                               pred->getOp().c_str(),
00065                                               pred->getVal2().c_str(), l_and );
00066       l_and =tmp;
00067
00068       return *this;
00069     }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/PredicateUtils.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/PredicateUtils.cpp

6.35 mrdb::DatabaseMetaData Class Reference

`#include <DatabaseMetaData.hpp>`

Public Member Functions

- virtual ~DatabaseMetaData ()
- virtual ResultSet * getTables (const std::string &tableNamePattern)=0
- virtual Types::SQLType getColumnType (const std::string &tableName, const std::string &columnName)=0
- virtual ResultSet * getColumns ()=0

6.35.1 Detailed Description

Definition at line 10 of file DatabaseMetaData.hpp.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 ~DatabaseMetaData()

```
virtual mrdb::DatabaseMetaData::~DatabaseMetaData ( )  [inline], [virtual]
```

Definition at line 12 of file DatabaseMetaData.hpp.

```
00012 {}
```

6.35.3 Member Function Documentation

6.35.3.1 getColumns()

```
virtual ResultSet * mrdb::DatabaseMetaData::getColumns ( )  [pure virtual]
```

Returns

a ResultSet containing a table with three fields: field1: TABLE_NAME is a table name field2: COLUMN_↩ NAME is a column name in <TABLE_NAME> field3: TYPE_NAME is the type name of column <COLUMN↩ _NAME>// results are sorted by TABLE_NAME

6.35.3.2 getColumnType()

```
virtual Types::SQLType mrdb::DatabaseMetaData::getColumnType (
            const std::string & tableName,
            const std::string & columnName )  [pure virtual]
```

Parameters

tableName	the table name to query
columnName	the column name whose type is sought

Returns

the SQLType of the given column for the given table

6.35.3.3 getTables()

```
virtual ResultSet * mrdb::DatabaseMetaData::getTables (
            const std::string & tableNamePattern )  [pure virtual]
```

Searches the current db for given table names.

Parameters

tableNamePattern	a pattern to be used to filter the result

Returns

a ResultSet with a row for each found table, first column of the result contains the name of the found table. Usage example: getTables("mr_%") --> retuns all tables having a name starting with mr_

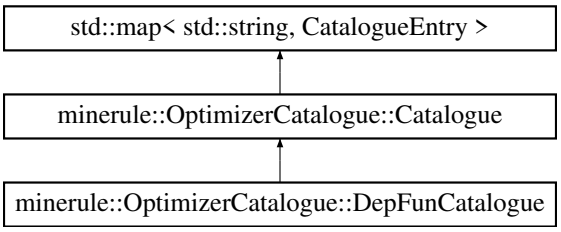
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/DatabaseMetaData.hpp

6.36 minerule::OptimizerCatalogue::DepFunCatalogue Class Reference

`#include <OptimizerCatalogue.hpp>`

Inheritance diagram for minerule::OptimizerCatalogue::DepFunCatalogue:



Public Member Functions

- DepFunCatalogue ()
- DepFunCatalogue (const DepFunCatalogue &catalogue)
- virtual ~DepFunCatalogue ()
- virtual const std::string & getSchemaInfo (const std::string &schemaKey) const
- void insertMapping (const std::string &table, const KeyCols &origKeyCols, int refKeyId, OrderType orderType)
- void initialize ()

Static Public Member Functions

- static OrderType stringToOrder (const std::string &)

Data Fields

- K keys
  STL member.
- T elements
  STL member.

6.36.1 Detailed Description

Definition at line 119 of file OptimizerCatalogue.hpp.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 DepFunCatalogue() [1/2]

```
minerule::OptimizerCatalogue::DepFunCatalogue::DepFunCatalogue ( )  [inline]
```

Definition at line 122 of file OptimizerCatalogue.hpp.

```
00122                         : Catalogue() {
00123                     schema["tab_main"]="mr_dep_fun";
00124                     schema["tab_main_tab_name"]="lhs_tab_name";
00125                     schema["tab_main_lhs_key_id"]="lhs_att_list_id";
00126                     schema["tab_main_rhs_key_id"]="rhs_att_list_id";
00127                     schema["tab_main_order_type"]="order_type";
00128                     schema["tab_cols"]="mr_dep_fun_col";
00129                     schema["tab_cols_col_name"]="col_name";
00130                     schema["tab_cols_key_id"]="col_id";
00131                     initialize();
00132       }
```

6.36.2.2 DepFunCatalogue() [2/2]

```
minerule::OptimizerCatalogue::DepFunCatalogue::DepFunCatalogue (
            const DepFunCatalogue & catalogue )  [inline]
```

Definition at line 134 of file OptimizerCatalogue.hpp.

```
00134                                                       :
00135         Catalogue(catalogue), schema(catalogue.schema) { }
```

6.36.2.3 ~DepFunCatalogue()

```
virtual minerule::OptimizerCatalogue::DepFunCatalogue::~DepFunCatalogue ( )  [inline], [virtual]
```

Definition at line 137 of file OptimizerCatalogue.hpp.

```
00137 {}
```

6.36.3 Member Function Documentation

6.36.3.1 getSchemaInfo()

```
virtual const std::string & minerule::OptimizerCatalogue::DepFunCatalogue::getSchemaInfo (
            const std::string & schemaKey ) const  [inline], [virtual]
```

Implements minerule::OptimizerCatalogue::Catalogue.

Definition at line 138 of file OptimizerCatalogue.hpp.

```
00138                                                                                {
00139         std::map<std::string,std::string>::const_iterator it;
00140         it=schema.find(schemaKey);
00141         if(it==schema.end())
00142           throw MineruleException( MR_ERROR_CATALOGUE_ERROR,
00143                                    "Requested for the unknown schema key:"+schemaKey);
00144         return it->second;
00145       }
```

6.36.3.2 initialize()

```
void minerule::OptimizerCatalogue::Catalogue::initialize ( )  [inherited]
```

Definition at line 153 of file OptimizerCatalogue.cpp.

```
00153                                                {
00154   mrdb::Connection *connection =
00155       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00156
00157   try {
00158     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00159     std::string getCatalogueQuery =
00160         (std::string) "SELECT A." + getSchemaInfo("tab_main_tab_name") + ",B." +
00161         getSchemaInfo("tab_cols_col_name") + ",A." +
00162         getSchemaInfo("tab_main_rhs_key_id") + ",A." +
00163         getSchemaInfo("tab_main_order_type") + " " + "FROM " +
00164         getSchemaInfo("tab_main") + " AS A," + getSchemaInfo("tab_cols") +
00165         " AS B " + "WHERE A." + getSchemaInfo("tab_main_lhs_key_id") + "=B." +
00166         getSchemaInfo("tab_cols_key_id");
00167
00168     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getCatalogueQuery));
00169     KeyCols origKey;
00170     unsigned long refKeyId = 0;
00171     std::string curTable;
00172     OrderType orderType;
00173
00174     // We start building the first of the two col l lists.
00175     // we accumulate the result in the set origKey.
00176
00177     if (rs->next()) {
00178       curTable = rs->getString(1);
00179       origKey.insert(rs->getString(2));
00180       refKeyId = rs->getInt(3);
00181       orderType = stringToOrder(rs->getString(4));
00182     } else {
00183       return;
00184     }
00185
00186     while (rs->next()) {
00187       if (curTable != rs->getString(1) ||
00188           refKeyId != (unsigned long)rs->getInt(3)) {
00189         // here the current rule changed, in fact either the table
00190         // or the refKeyId are different
00191         insertMapping(curTable, origKey, refKeyId, orderType);
00192         origKey.clear();
00193
00194         curTable = rs->getString(1);
00195         origKey.insert(rs->getString(2));
00196         refKeyId = rs->getInt(3);
00197         orderType = stringToOrder(rs->getString(4));
00198       } else {
00199         // here we continue to accumulate the columns in origKey
00200         origKey.insert(rs->getString(2));
00201       }
00202     } // end while
00203
00204     assert(!origKey.empty());
00205     insertMapping(curTable, origKey, refKeyId, orderType);
00206   } catch (mrdb::SQLException &e) {
00207     throw MineruleException(
00208         MR_ERROR_DATABASE_ERROR,
00209         std::string(
00210             "Cannot access the db while "
00211             "initializing a member of the OptimizerCatalogue, the reason is:") +
00212             e.what());
00213   }
00214 }
```

6.36.3.3 insertMapping()

```
void minerule::OptimizerCatalogue::Catalogue::insertMapping (
              const std::string & table,
              const KeyCols & origKeyCols,
              int refKeyId,
              OrderType orderType )  [inherited]
```

Definition at line 104 of file OptimizerCatalogue.cpp.

```
00106                           {
00107   mrdb::Connection *connection =
00108       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00109
00110   try {
00111     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00112
00113     std::string getKeyQuery =
00114         std::string("SELECT " + getSchemaInfo("tab_cols_col_name") + " " +
00115                     "FROM " + getSchemaInfo("tab_cols") + " " + "WHERE " +
00116                     getSchemaInfo("tab_cols_key_id") + "=") +
00117         Converter((long)refKey).toString();
00118
00119     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getKeyQuery));
00120     KeyCols refKeyCols;
00121
00122     while (rs->next()) {
00123       refKeyCols.insert(rs->getString(1));
00124     }
00125
00126     //      OrderType orderType = getOrderType(origKey,refKey);
00127     // (*this)[table][cols] = pair<KeyCols,OrderType>(refKeyCols,orderType);
00128     std::pair< KeyCols, std::pair<KeyCols, OrderType> > insElem(
00129         cols, std::pair<KeyCols, OrderType>(refKeyCols, orderType));
00130     (*this)[table].insert(insElem);
00131
00132   } catch (mrdb::SQLException &e) {
00133     throw MineruleException(
00134         MR_ERROR_DATABASE_ERROR,
00135         std::string("Cannot access the db while "
00136                     "initializing a member of the OptimizerCatalogue,"
00137                     "the reason is:") +
00138             e.what());
00139   }
00140 }
```

6.36.3.4 stringToOrder()

```
OptimizerCatalogue::OrderType minerule::OptimizerCatalogue::Catalogue::stringToOrder (
            const std::string & str )  [static], [inherited]
```

Definition at line 143 of file OptimizerCatalogue.cpp.

```
00143                                                          {
00144   if (str == "r")
00145     return Reversed;
00146
00147   if (str == "e")
00148     return Equal;
00149
00150   return None;
00151 }
```

6.36.4 Field Documentation

6.36.4.1 elements

```
T std::map< K, T >::elements  [inherited]
```

STL member.

6.36.4.2 keys

```
K std::map< K, T >::keys  [inherited]
```

STL member.

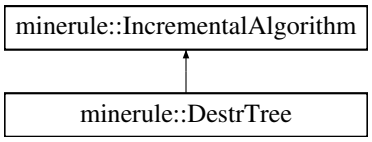
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp

6.37 minerule::DestrTree Class Reference

```
#include <DestrTree.hpp>
```

Inheritance diagram for minerule::DestrTree:



Public Member Functions

- DestrTree (const OptimizedMinerule &mr)
- virtual ~DestrTree ()
- virtual void execute ()

Static Public Member Functions

- static IncrementalAlgorithm * newIncrementalAlgorithm (const OptimizedMinerule &mr)

Protected Member Functions

- void adjustSuppRSet ()
- void adjustSuppMIndex ()
- void insertRulesInStructure ()
- void adjustSupp ()
- void prepareData ()
- size_t buildAttrStr (const ParsedMinerule::AttrVector &attr, size_t startIndex, std::string &attrStr, std::vector< int > &des) const
- std::string buildQry (const std::string &groupAttrStr, const std::string &attrStr, const std::string &constraints) const
- std::string buildQry1NotQry2 (const std::string &groupAttrStr, const std::string &attrStr, const std::string &constraint1, const std::string &constraint2) const
- Body * getRoot ()

Protected Attributes

- Body * root
- size_t ngroups
- mrdb::ResultSet * rb1
- mrdb::ResultSet * rh1
- mrdb::ResultSet * rb1nb2
- mrdb::ResultSet * rh1nh2
- SourceRowColumnIds bodyDes
- SourceRowColumnIds headDes
- mrdb::Statement * stateb1
- mrdb::Statement * stateb1nb2
- mrdb::Statement * stateh1
- mrdb::Statement * stateh1nh2
- const OptimizedMinerule * minerule

6.37.1 Detailed Description

This class implements <COMPLETE>

Definition at line 32 of file DestrTree.hpp.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 DestrTree()

```
minerule::DestrTree::DestrTree (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 69 of file DestrTree.hpp.

```
00069 : IncrementalAlgorithm(mr), root(new Body()), ngroups(0), rb1(NULL), rh1(NULL), rb1nb2(NULL),
       rh1nh2(NULL) { }
```

6.37.2.2 ~DestrTree()

```
virtual minerule::DestrTree::~DestrTree ( )  [inline], [virtual]
```

Definition at line 71 of file DestrTree.hpp.

```
00071                         {
00072   // Trashing the trashable
00073         if(rh1!=NULL) delete rh1;
00074         if(stateh1!=NULL) delete stateh1;
00075         if(rb1!=NULL) delete rb1;
00076         if(stateb1!=NULL) delete stateb1;
00077         rb1=rh1=NULL;
00078         stateh1=stateb1=NULL;
00079
00080     delete root;
00081   }
```

6.37.3 Member Function Documentation

6.37.3.1 adjustSupp()

```
void minerule::DestrTree::adjustSupp ( )  [protected]
```

Definition at line 190 of file DestrTree.cpp.

```
00190                                   {
00191                 MRLogPusher _("Starting the mining algorithm...");
00192
00193                 MRLog() « "Preparing the data structures..." « std::endl;
00194                 insertRulesInStructure();
00195                 MRLog() « "Done!" « std::endl;
00196
00197                 Connection connection;
00198                 connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00199                 connection.useMRDBConnection(
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00200                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule->getParsedMinerule()));
00201
00202                 if (rb1!=NULL  &&  rh1!=NULL) {
00203                                 MRLog() « "Adjusting support..." « std::endl;
00204                                 adjustSuppRSet();
00205                                 MRLog() « "Done!" « std::endl;
00206
00207                                 std::vector<ItemType> body;
00208                                 MRLog() « "Extracting rules..." « std::endl;
00209                                 getRoot()->extractRules(body, minerule->getParsedMinerule().sup,
      minerule->getParsedMinerule().conf, ngroups, &connection);
00210                                 MRLog() « "Done!" « std::endl;
00211                 } else throw MineruleException (MR_ERROR_INTERNAL, " cannot create rules ");
00212         }
```

6.37.3.2 adjustSuppMIndex()

```
void minerule::DestrTree::adjustSuppMIndex ( )  [protected]
```

6.37.3.3 adjustSuppRSet()

```
void minerule::DestrTree::adjustSuppRSet ( )  [protected]
```

Definition at line 106 of file DestrTree.cpp.

```
00106                                       {
00107                 MRLog() « "findRulesInTree da adjustSuppRSet()..."«std::endl;
00108                 ItemType gb;
00109                 ItemType gh;
00110                 bool newbody=1;
00111                 bool newhead=1;
00112                 bool notend=1;
00113                 bool b1notend=rb1->next();
00114                 bool h1notend=rh1->next();
00115                 bool b1nb2notend=rb1nb2->next();
00116                 bool h1nh2notend=rh1nh2->next();
00117                 ItemSet* b1=new ItemSet();
00118                 ItemSet* b1nb2=new ItemSet();
00119                 ItemSet* h1=new ItemSet();
00120                 ItemSet* h1nh2=new ItemSet();
00121
00122                 while( notend ) {
00123                         if (b1notend && newbody){
00124                                 delete b1; delete b1nb2;
```

```
00125                                   b1=new ItemSet();
00126                                   b1nb2=new ItemSet();
00127                                   SourceRow curRowb1(rb1, bodyDes);
00128                                   gb=curRowb1.getGroup();
00129
00130                                   while(b1notend && ItemType(curRowb1.getGroup())==gb){
00131                                           b1->push_back(curRowb1.getBody());
00132                                           if((b1notend=rb1->next())) {
00133                                                   curRowb1.init(rb1, bodyDes);
00134                                           }
00135                                   }
00136                                   if(b1nb2notend){
00137                                           SourceRow curRowb12(rb1nb2, bodyDes);
00138                                           while(b1nb2notend && ItemType(curRowb12.getGroup())==gb){
00139                                                   b1nb2->push_back(curRowb12.getBody());
00140                                                   if((b1nb2notend=rb1nb2->next())) {
00141                                                           curRowb12.init(rb1nb2, bodyDes);
00142                                                   }
00143                                           }
00144                                   }
00145                           }
00146
00147
00148
00149                           if(h1notend && newhead) {
00150                                   delete h1; delete h1nh2;
00151                                   h1=new ItemSet();
00152                                   h1nh2=new ItemSet();
00153                                   SourceRow curRowh1(rh1, headDes);
00154                                   gh=curRowh1.getGroup();
00155
00156                                   while(h1notend && ItemType(curRowh1.getGroup())==gh ){
00157                                           h1->push_back(curRowh1.getHead());
00158                                           if ((h1notend=rh1->next())) {
00159                                                   curRowh1.init(rh1, headDes);
00160                                           }
00161                                   }
00162                                   if(h1nh2notend){
00163                                           SourceRow curRowh12(rh1nh2, headDes);
00164                                           while(h1nh2notend && ItemType(curRowh12.getGroup())==gh ){
00165                                                   h1nh2->push_back(curRowh12.getHead());
00166                                                   if ((h1nh2notend=rh1nh2->next())) {
00167                                                           curRowh12.init(rh1nh2, headDes);
00168                                                   }
00169                                           }
00170                                   }
00171                           }
00172
00173
00174                           if (!(gb<gh) && !(gb==gh)) {
00175                                   root->findRulesInTree(NULL, NULL, h1, h1nh2);
00176                                   newbody=0;newhead=1;
00177                           } else if (gb<gh) {
00178                                   root->findRulesInTree(b1, b1nb2, NULL, NULL);
00179                                   newbody=1;newhead=0;
00180                           } else if (gb==gh) {
00181                                   root->findRulesInTree(b1, b1nb2, h1, h1nh2);
00182                                   newbody=1;newhead=1;
00183                           }
00184
00185                           if (!b1notend && !h1notend) notend=0;
00186                   }
00187           }
```

6.37.3.4 buildAttrStr()

```
size_t minerule::DestrTree::buildAttrStr (
            const ParsedMinerule::AttrVector & attr,
            size_t startIndex,
            std::string & attrStr,
            std::vector< int > & des ) const  [protected]
```

Definition at line 215 of file DestrTree.cpp.

```
00218                                   {
00219                   ParsedMinerule::AttrVector::const_iterator it = attr.begin();
00220                   for( ; it!=attr.end(); it++ ) {
00221                           if(it!=attr.begin()) {
```

```
00222                                   attrStr+=",";
00223                           }
00224
00225                           attrStr+=*it;
00226                           des.push_back(++startIndex);
00227                   }
00228
00229                   return startIndex;
00230           }
```

6.37.3.5 buildQry()

```
std::string minerule::DestrTree::buildQry (
            const std::string & groupAttrStr,
            const std::string & attrStr,
            const std::string & constraints ) const  [protected]
```

Definition at line 235 of file DestrTree.cpp.

```
00235
                                {
00236
00237                   return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00238                           "FROM "+minerule->getParsedMinerule().tab_source+" "+
00239                           (constraints.size()>0 ? "WHERE "+constraints+" " :"")
00240                           +"ORDER BY "+groupAttrStr);
00241           }
```

6.37.3.6 buildQry1NotQry2()

```
std::string minerule::DestrTree::buildQry1NotQry2 (
            const std::string & groupAttrStr,
            const std::string & attrStr,
            const std::string & constraint1,
            const std::string & constraint2 ) const  [protected]
```

Definition at line 244 of file DestrTree.cpp.

```
00244
                                                                        {
00245
00246                   return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00247                           "FROM "+minerule->getParsedMinerule().tab_source+" "
00248                           "WHERE "+constraint1+ (constraint2.empty() ? " " : " AND NOT("+constraint2+")
      ") +
00249                           "ORDER BY "+groupAttrStr);
00250           }
```

6.37.3.7 execute()

```
void minerule::DestrTree::execute ( )  [virtual]
```

Implements minerule::IncrementalAlgorithm.

Definition at line 311 of file DestrTree.cpp.

```
00312           {
00313                   assert( minerule->getOptimizationInfo().relationship == OptimizedMinerule::Dominance
      );
00314                   assert( minerule->getParsedMinerule().mc!=NULL &&
      minerule->getParsedMinerule().mc->next==NULL);
00315
00316                   MRLogPush("This is the Context Dependent Destructive Mining Algorithm...");
00317                   prepareData();
00318                   adjustSupp();
00319                   MRLogPop();
00320           }
```

6.37.3.8 getRoot()

```
Body * minerule::DestrTree::getRoot ( )  [inline], [protected]
```

Definition at line 67 of file DestrTree.hpp.

```
00067 {return root;}
```

6.37.3.9 insertRulesInStructure()

```
void minerule::DestrTree::insertRulesInStructure ( )  [protected]
```

Definition at line 28 of file DestrTree.cpp.

```
00028                                                {
00029                 QueryResult::Iterator qit;
00030 
00031 
      OptimizerCatalogue::getMRQueryResultIterator(minerule->getOptimizationInfo().minerule.tab_result,
      qit, -1, 0.0);
00032                 Head* newhead;
00033                 while( qit.next()) {
00034                         Rule r;
00035                         qit.getRule(r);
00036                         double suppr=round(r.getSupport()*ngroups);
00037                         double suppb=round(suppr/r.getConfidence());
00038                         newhead=root->insertItemSetB(r.getBody(),suppb);
00039                         newhead->insertItemSetH(r.getHead(),suppr);
00040                 }
00041 
00042         }
```

6.37.3.10 newIncrementalAlgorithm()

```
IncrementalAlgorithm * minerule::IncrementalAlgorithm::newIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.cpp.

```
00025                                                                                {
00026     // if mr has only ItemDependent constraints
00027                 MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028 
00029                 if( mr.hasIDConstraints() ) {
00030                         MRLog() « "The minerule is item dependent!" « std::endl;
00031                         if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                 return new ResultCombinator(mr);
00033                         else
00034                                 return new IDIncrementalAlgorithm(mr);
00035                 }
00036 
00037                 MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038 
00039                 if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040 
00041                         IncrementalAlgorithm* incrAlgo = NULL;
00042 
00043                         switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                 case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                         incrAlgo = new ConstrTree(mr);
00046                                         break;
00047                                 case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                         incrAlgo = new DestrTree(mr);
00049                                         break;
00050                                 case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                         incrAlgo = new ConstrTree(mr);
00052                                         break;
00053                         }
00054 
```

```
00055                         return incrAlgo;
00056                 } else {
00057                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                 « " to the system yet." « std::endl;
00059                         return NULL;
00060                 }
00061         }
```

6.37.3.11 prepareData()

```
void minerule::DestrTree::prepareData ( )  [protected]
```

Definition at line 252 of file DestrTree.cpp.

```
00252                                   {
00253                 MRLogPush("Building source information");
00254
00255                 MRLog() « "Separating the constraints in the HEAD and BODY parts..."«std::endl;
00256                 std::string q1BodyConstraints;
00257                 std::string q1HeadConstraints;
00258                 std::string q2BodyConstraints;
00259                 std::string q2HeadConstraints;
00260
      HeadBodyPredicatesSeparator::separate(minerule->getOptimizationInfo().minerule.mc!=NULL?
      minerule->getOptimizationInfo().minerule.mc->l_and:NULL, q1BodyConstraints, q1HeadConstraints);
00261
00262                 MRLog() « "Separating the constraints in the HEAD and BODY parts (query 2)..." «
      std::endl;
00263
00264                 HeadBodyPredicatesSeparator::separate(minerule->getParsedMinerule().mc->l_and,
      q2BodyConstraints, q2HeadConstraints);
00265
00266                 MRLog() « "Building db queries" « std::endl;
00267                 size_t index;
00268                 std::string groupAttr;
00269                 std::string bodyAttr;
00270                 std::string headAttr;
00271                 index=buildAttrStr(minerule->getParsedMinerule().ga,0,groupAttr,bodyDes.groupElems );
00272
00273                 headDes.groupElems=bodyDes.groupElems;
00274
00275                 buildAttrStr(minerule->getParsedMinerule().ba,index,bodyAttr,bodyDes.bodyElems);
00276                 buildAttrStr(minerule->getParsedMinerule().ha,index,headAttr,headDes.headElems);
00277
00278                 std::string bodyQry1 =buildQry( groupAttr,bodyAttr,q1BodyConstraints);
00279                 std::string headQry1 =buildQry( groupAttr,headAttr,q1HeadConstraints);
00280
00281                 std::string bodyQry1NotQry2 = buildQry1NotQry2(
      groupAttr,bodyAttr,q1BodyConstraints,q2BodyConstraints);
00282                 std::string headQry1NotQry2 = buildQry1NotQry2( groupAttr, headAttr,
      q1HeadConstraints, q2HeadConstraints);
00283
00284                 MRLog() « "Executing queries" « std::endl;
00285                 mrdb::Connection* con =
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00286
00287
00288                 stateb1 = con->createStatement();
00289                 MRDebug("bodyQry1:"+bodyQry1);
00290                 rb1 = stateb1->executeQuery(bodyQry1);
00291
00292                 stateh1 = con->createStatement();
00293                 MRDebug("headQry1:"+headQry1);
00294                 rh1 = stateh1->executeQuery(headQry1);
00295
00296
00297                 stateb1nb2 = con->createStatement();
00298                 MRDebug("bodyQry1NotQry2:"+bodyQry1NotQry2);
00299                 rb1nb2 = stateb1nb2->executeQuery(bodyQry1NotQry2);
00300
00301                 stateh1nh2 = con->createStatement();
00302                 MRDebug("headQry1NotQry2:"+headQry1NotQry2);
00303                 rh1nh2 = stateh1nh2->executeQuery(headQry1NotQry2);
00304
00305                 ngroups = PrepareDataUtils::evaluateTotGroups(minerule->getParsedMinerule());
00306
00307                 MRLogPop();
00308         }
```

6.37.4 Field Documentation

6.37.4.1 bodyDes

```
SourceRowColumnIds minerule::DestrTree::bodyDes  [protected]
```

Definition at line 40 of file DestrTree.hpp.

6.37.4.2 headDes

```
SourceRowColumnIds minerule::DestrTree::headDes  [protected]
```

Definition at line 41 of file DestrTree.hpp.

6.37.4.3 minerule

```
const OptimizedMinerule* minerule::IncrementalAlgorithm::minerule  [protected], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.hpp.

6.37.4.4 ngroups

```
size_t minerule::DestrTree::ngroups  [protected]
```

Definition at line 35 of file DestrTree.hpp.

6.37.4.5 rb1

```
mrdb::ResultSet* minerule::DestrTree::rb1  [protected]
```

Definition at line 36 of file DestrTree.hpp.

6.37.4.6 rb1nb2

```
mrdb::ResultSet* minerule::DestrTree::rb1nb2  [protected]
```

Definition at line 38 of file DestrTree.hpp.

6.37.4.7 rh1

```
mrdb::ResultSet* minerule::DestrTree::rh1  [protected]
```

Definition at line 37 of file DestrTree.hpp.

6.37.4.8 rh1nh2

```
mrdb::ResultSet* minerule::DestrTree::rh1nh2  [protected]
```

Definition at line 39 of file DestrTree.hpp.

6.37.4.9 root

```
Body* minerule::DestrTree::root  [protected]
```

Definition at line 34 of file DestrTree.hpp.

6.37.4.10 stateb1

```
mrdb::Statement* minerule::DestrTree::stateb1  [protected]
```

Definition at line 43 of file DestrTree.hpp.

6.37.4.11 stateb1nb2

```
mrdb::Statement * minerule::DestrTree::stateb1nb2  [protected]
```

Definition at line 43 of file DestrTree.hpp.

6.37.4.12 stateh1

```
mrdb::Statement* minerule::DestrTree::stateh1  [protected]
```

Definition at line 44 of file DestrTree.hpp.

6.37.4.13 stateh1nh2

```
mrdb::Statement * minerule::DestrTree::stateh1nh2  [protected]
```

Definition at line 44 of file DestrTree.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/DestrTree.hpp
- /Users/esposito/Software/minerule/src/Algorithms/DestrTree.cpp

6.38 minerule::Dist_cond Class Reference

```
#include <ParsedMinerule.hpp>
```

Public Member Functions

- Dist_cond ()
- Dist_cond (const Dist_cond &copy_me)

Static Public Member Functions

- static std::vector< Dist_cond * > copyDistCond (std::vector< Dist_cond * > copy_me)

Data Fields

- std::string function
- std::vector< std::string > attr
- std::string range

6.38.1 Detailed Description

Definition at line 109 of file ParsedMinerule.hpp.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Dist_cond() [1/2]

```
minerule::Dist_cond::Dist_cond ( )  [inline]
```

Definition at line 115 of file ParsedMinerule.hpp.

```
00115                 {
00116         function="";
00117         range="";
00118     }
```

6.38.2.2 Dist_cond() [2/2]

```
minerule::Dist_cond::Dist_cond (
            const Dist_cond & copy_me )  [inline]
```

Definition at line 120 of file ParsedMinerule.hpp.

```
00120                                       {
00121         function=copy_me.function;
00122         range=copy_me.range;
00123         for(int i=0; i<copy_me.attr.size(); ++i) {
00124             attr.push_back(copy_me.attr[i]);
00125         }
00126     }
```

6.38.3 Member Function Documentation

6.38.3.1 copyDistCond()

```
static std::vector< Dist_cond * > minerule::Dist_cond::copyDistCond (
            std::vector< Dist_cond * > copy_me )  [inline], [static]
```

Definition at line 128 of file ParsedMinerule.hpp.

```
00128                                                                       {
00129         std::vector<Dist_cond*> out;
00130         for(int i=0; i<copy_me.size(); ++i)
00131             out.push_back(new Dist_cond(*copy_me[i]));
00132         return out;
00133     }
```

6.38.4 Field Documentation

6.38.4.1 attr

```
std::vector<std::string> minerule::Dist_cond::attr
```

Definition at line 112 of file ParsedMinerule.hpp.

6.38.4.2 function

```
std::string minerule::Dist_cond::function
```

Definition at line 111 of file ParsedMinerule.hpp.

6.38.4.3 range

```
std::string minerule::Dist_cond::range
```

Definition at line 113 of file ParsedMinerule.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp

6.39 minerule::QueryNormalizer::SubstEntryHead::Elem Class Reference

```
#include <QueryNormalizer.hpp>
```

Public Member Functions

- Elem ()
- Elem (const Elem &elem)

Data Fields

- OptimizerCatalogue::OrderType order
- std::string colName
- std::string value
- std::string op

6.39.1 Detailed Description

Definition at line 63 of file QueryNormalizer.hpp.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 Elem() [1/2]

```
minerule::QueryNormalizer::SubstEntryHead::Elem::Elem ( )  [inline]
```

Definition at line 71 of file QueryNormalizer.hpp.

```
00071                {
00072         }
```

6.39.2.2 Elem() [2/2]

```
minerule::QueryNormalizer::SubstEntryHead::Elem::Elem (
             const Elem & elem )  [inline]
```

Definition at line 74 of file QueryNormalizer.hpp.

```
00074                                :
00075           order(elem.order),
00076           colName(elem.colName),
00077           value(elem.value),
00078           op(elem.op)  {}
```

6.39.3 Field Documentation

6.39.3.1 colName

```
std::string minerule::QueryNormalizer::SubstEntryHead::Elem::colName
```

Definition at line 67 of file QueryNormalizer.hpp.

6.39.3.2 op

```
std::string minerule::QueryNormalizer::SubstEntryHead::Elem::op
```

Definition at line 69 of file QueryNormalizer.hpp.

6.39.3.3 order

```
OptimizerCatalogue::OrderType minerule::QueryNormalizer::SubstEntryHead::Elem::order
```

Definition at line 65 of file QueryNormalizer.hpp.

6.39.3.4 value

```
std::string minerule::QueryNormalizer::SubstEntryHead::Elem::value
```

Definition at line 68 of file QueryNormalizer.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp

6.40 minerule::EncodedNF Class Reference

```
#include <ExpressionNFCoder.hpp>
```

Public Types

- enum CodesRelationship { FirstMoreGeneral , FirstMoreSpecific , Equivalent , Unrelated }

Public Member Functions

- bool operator== (const EncodedNF &) const

Static Public Member Functions

- static CodesRelationship getCodesRelationship (const EncodedNF &, const EncodedNF &)
- static size_t computeHammingDistance (const EncodedNF &nf1, const EncodedNF &nf2)

Data Fields

- const EncodingBaseType * encVector
- size_t numVars
- size_t vecSize
- size_t codingLen

6.40.1 Detailed Description

Definition at line 34 of file ExpressionNFCoder.hpp.

6.40.2 Member Enumeration Documentation

6.40.2.1 CodesRelationship

```
enum minerule::EncodedNF::CodesRelationship
```

Enumerator

FirstMoreGeneral	
FirstMoreSpecific	
Equivalent	
Unrelated	

Definition at line 36 of file ExpressionNFCoder.hpp.

```
00036                   {
00037       FirstMoreGeneral,
00038       FirstMoreSpecific,
00039       Equivalent,
00040       Unrelated
00041     } CodesRelationship;
```

6.40.3 Member Function Documentation

6.40.3.1 computeHammingDistance()

```
size_t minerule::EncodedNF::computeHammingDistance (
            const EncodedNF & nf1,
            const EncodedNF & nf2 )  [static]
```

The result of this function is equivalent to the following algorithm:

1. write down the sequence s1 of the ones and zeros of the truth table of the predicate that nf1 represents.
2. Do the same with nf2 and call the resultingstd::string s2.
3. Return the Hamming distance of s1 w.r.t. s2 in other words the function count the number of variable configurations for which the two predicates return different answers.

The implementation does not match exactly the above algorithm only because nf1 and nf2 does not contain the s1 and s2 strings (and building s1 and s2 from them would be costly at execution time).

The implementation, instead iterates thru nf1 and nf2 counting the number of times they do not match.

Definition at line 323 of file ExpressionNFCoder.cpp.

```
00324                                                                   {
00325     assert(nf1.numVars == nf2.numVars);
00326     size_t edt = 0;
00327
00328     EncodedNFIterator it1(nf1);
00329     EncodedNFIterator it2(nf2);
00330     it1++;
00331     it2++;
00332     while(it1.ok() && it2.ok()) {
00333       if( *it1 == *it2 ) {
00334         it1++;
00335         it2++;
00336         continue;
00337       }
00338
00339       if(*it1 < *it2) {
00340         edt++;
00341         it1++;
00342         continue;
00343       }
00344
00345       assert(*it1>*it2);
```

```
00346       edt++;
00347       it2++;
00348     }
00349
00350     while(it1.ok()) {
00351       edt++;
00352       it1++;
00353     }
00354
00355     while(it2.ok()) {
00356       edt++;
00357       it2++;
00358     }
00359
00360     return edt;
00361   }
```

6.40.3.2 getCodesRelationship()

```
EncodedNF::CodesRelationship minerule::EncodedNF::getCodesRelationship (
              const EncodedNF & nf1,
              const EncodedNF & nf2 )  [static]
```

Definition at line 238 of file ExpressionNFCoder.cpp.

```
00239                                                                    {
00240     if(nf1.numVars   != nf2.numVars)
00241       return Unrelated;
00242
00243     CodesRelationship relationship = Equivalent;
00244
00245     EncodedNFIterator it1(nf1);
00246     EncodedNFIterator it2(nf2);
00247     it1++;
00248     it2++;
00249     while(it1.ok() && it2.ok()) {
00250       if(*it1 == *it2) {
00251         it1++;
00252         it2++;
00253         continue;
00254       }
00255
00256       if(*it1 < *it2 ) {
00257         // in nf2 there is *not* the current value of
00258         // it1. Then nf1 does not contain at least one
00259         // disjunct that is present in nf2, and hence
00260         // this is evidence that nf1 is more general
00261         if(relationship==FirstMoreSpecific)
00262           return Unrelated;
00263         else
00264           relationship=FirstMoreGeneral;
00265
00266         it1++;
00267         continue;
00268       }
00269
00270       if(*it1 > *it2 ) {
00271         // in nf1 there is *not* the current value of
00272         // it2. Then nf1 does not contain at least one
00273         // disjunct that is present in nf2, and hence
00274         // this is evidence that nf2 is more general
00275         if(relationship==FirstMoreGeneral)
00276           return Unrelated;
00277         else
00278           relationship=FirstMoreSpecific;
00279
00280         it2++;
00281       }
00282     }
00283
00284     if( !it1.ok() && !it2.ok() )
00285       return relationship;
00286
00287     if( it1.ok() ) {
00288       // it1.ok() && !it2.ok() suggests
00289       // nf1 is more general
00290       if( relationship==FirstMoreSpecific )
00291         return Unrelated;
00292       else
```

```
00293         return FirstMoreGeneral;
00294     }
00295
00296     // !it1.ok() & it2.ok() suggests that
00297     //  nf2 is more general
00298     if( relationship==FirstMoreGeneral )
00299       return Unrelated;
00300     else
00301       return FirstMoreSpecific;
00302   }
```

6.40.3.3 operator==()

```
bool minerule::EncodedNF::operator== (
             const EncodedNF & nf ) const
```

Definition at line 139 of file ExpressionNFCoder.cpp.

```
00139                                                {
00140     if(numVars   != nf.numVars ||
00141        vecSize   != nf.vecSize ||
00142        codingLen != nf.codingLen )
00143       return false;
00144
00145     for(size_t i=0; i<vecSize; i++) {
00146       if( encVector[i]!=nf.encVector[i] )
00147         return false;
00148     }
00149
00150     return true;
00151   }
```

6.40.4 Field Documentation

6.40.4.1 codingLen

```
size_t minerule::EncodedNF::codingLen
```

Definition at line 47 of file ExpressionNFCoder.hpp.

6.40.4.2 encVector

```
const EncodingBaseType* minerule::EncodedNF::encVector
```

Definition at line 43 of file ExpressionNFCoder.hpp.

6.40.4.3 numVars

```
size_t minerule::EncodedNF::numVars
```

Definition at line 44 of file ExpressionNFCoder.hpp.

6.40.4.4 vecSize

```
size_t minerule::EncodedNF::vecSize
```

Definition at line 46 of file ExpressionNFCoder.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp

6.41 minerule::EncodedNFIterator Class Reference

```
#include <ExpressionNFCoder.hpp>
```

Public Member Functions

- EncodedNFIterator (const EncodedNF &er)
- bool ok () const
- void setOk (bool val)
- bool operator++ (int)
- const VarSet & operator* () const

6.41.1 Detailed Description

Definition at line 66 of file ExpressionNFCoder.hpp.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 EncodedNFIterator()

```
minerule::EncodedNFIterator::EncodedNFIterator (
            const EncodedNF & er )  [inline]
```

Definition at line 73 of file ExpressionNFCoder.hpp.

```
00073                                       :
00074       target(er), currentVarSet(er.numVars), cellCounter(0), elemCounter(0), itok(false) {
00075
00076     }
```

6.41.3 Member Function Documentation

6.41.3.1 ok()

```
bool minerule::EncodedNFIterator::ok ( ) const  [inline]
```

Definition at line 78 of file ExpressionNFCoder.hpp.

```
00078                       {
00079       return itok;
00080     }
```

6.41.3.2 operator∗()

```
const VarSet & minerule::EncodedNFIterator::operator* ( ) const  [inline]
```

Definition at line 88 of file ExpressionNFCoder.hpp.

```
00088                                  {
00089       return currentVarSet;
00090     }
```

6.41.3.3 operator++()

```
bool minerule::EncodedNFIterator::operator++ (
             int  )
```

Definition at line 109 of file ExpressionNFCoder.cpp.

```
00109                                  {
00110     static const size_t baseTypeSize = (sizeof(EncodingBaseType)*8);
00111
00112     if(cellCounter*baseTypeSize+elemCounter>=target.codingLen) {
00113       setOk(false);
00114       return false;
00115     }
00116
00117     setOk(true);
00118
00119     for( size_t i=0; i<target.numVars; i++ ) {
00120       bool elem = target.encVector[cellCounter] & (1<<elemCounter);
00121       currentVarSet.setVar(i, elem);
00122
00123       if(++elemCounter >= baseTypeSize) {
00124         ++cellCounter;
00125         elemCounter=0;
00126       }
00127     }
00128
00129     return true;
00130   }
```

6.41.3.4 setOk()

```
void minerule::EncodedNFIterator::setOk (
             bool val )  [inline]
```

Definition at line 82 of file ExpressionNFCoder.hpp.

```
00082                          {
00083       itok=val;
00084     }
```

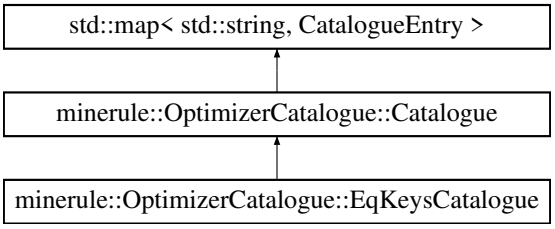
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp

6.42 minerule::OptimizerCatalogue::EqKeysCatalogue Class Reference

`#include <OptimizerCatalogue.hpp>`

Inheritance diagram for minerule::OptimizerCatalogue::EqKeysCatalogue:



Public Member Functions

- EqKeysCatalogue ()
- EqKeysCatalogue (const EqKeysCatalogue &catalogue)
- virtual ~EqKeysCatalogue ()
- virtual const std::string & getSchemaInfo (const std::string &schemaKey) const
- void insertMapping (const std::string &table, const KeyCols &origKeyCols, int refKeyId, OrderType orderType)
- void initialize ()

Static Public Member Functions

- static OrderType stringToOrder (const std::string &)

Data Fields

- K keys
  STL member.
- T elements
  STL member.

6.42.1 Detailed Description

Definition at line 91 of file OptimizerCatalogue.hpp.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 EqKeysCatalogue() [1/2]

```
minerule::OptimizerCatalogue::EqKeysCatalogue::EqKeysCatalogue ( )  [inline]
```

Definition at line 94 of file OptimizerCatalogue.hpp.

```
00094                       : Catalogue() {
00095                   schema["tab_main"]="mr_eq_keys";
00096                   schema["tab_main_tab_name"]="tab_name";
00097                   schema["tab_main_lhs_key_id"]="key_id";
00098                   schema["tab_main_rhs_key_id"]="ref_key_id";
00099                   schema["tab_main_order_type"]="order_type";
00100                   schema["tab_cols"]="mr_eq_keys_col";
00101                   schema["tab_cols_col_name"]="col_name";
00102                   schema["tab_cols_key_id"]="key_id";
00103                   initialize();
00104           }
```

6.42.2.2 EqKeysCatalogue() [2/2]

```
minerule::OptimizerCatalogue::EqKeysCatalogue::EqKeysCatalogue (
            const EqKeysCatalogue & catalogue )  [inline]
```

Definition at line 106 of file OptimizerCatalogue.hpp.

```
00106 : Catalogue(catalogue), schema(catalogue.schema) { }
```

6.42.2.3 ~EqKeysCatalogue()

```
virtual minerule::OptimizerCatalogue::EqKeysCatalogue::~EqKeysCatalogue ( )  [inline], [virtual]
```

Definition at line 107 of file OptimizerCatalogue.hpp.

```
00107 {}
```

6.42.3 Member Function Documentation

6.42.3.1 getSchemaInfo()

```
virtual const std::string & minerule::OptimizerCatalogue::EqKeysCatalogue::getSchemaInfo (
            const std::string & schemaKey ) const  [inline], [virtual]
```

Implements minerule::OptimizerCatalogue::Catalogue.

Definition at line 109 of file OptimizerCatalogue.hpp.

```
00109                                                                   {
00110                   std::map<std::string,std::string>::const_iterator it;
00111                   it=schema.find(schemaKey);
00112                   if(it==schema.end())
00113                           throw MineruleException( MR_ERROR_CATALOGUE_ERROR,
00114                                   "Requested for the unknown schema key:"+schemaKey);
00115                   return it->second;
00116       }
```

6.42.3.2 initialize()

```
void minerule::OptimizerCatalogue::Catalogue::initialize ( )  [inherited]
```

Definition at line 153 of file OptimizerCatalogue.cpp.

```
00153                                           {
00154   mrdb::Connection *connection =
00155       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00156
00157   try {
00158     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00159     std::string getCatalogueQuery =
00160         (std::string) "SELECT A." + getSchemaInfo("tab_main_tab_name") + ",B." +
00161         getSchemaInfo("tab_cols_col_name") + ",A." +
00162         getSchemaInfo("tab_main_rhs_key_id") + ",A." +
00163         getSchemaInfo("tab_main_order_type") + " " + "FROM " +
00164         getSchemaInfo("tab_main") + " AS A," + getSchemaInfo("tab_cols") +
00165         " AS B " + "WHERE A." + getSchemaInfo("tab_main_lhs_key_id") + "=B." +
00166         getSchemaInfo("tab_cols_key_id");
00167
00168     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getCatalogueQuery));
00169     KeyCols origKey;
00170     unsigned long refKeyId = 0;
00171     std::string curTable;
00172     OrderType orderType;
00173
00174     // We start building the first of the two col l lists.
00175     // we accumulate the result in the set origKey.
00176
00177     if (rs->next()) {
00178       curTable = rs->getString(1);
00179       origKey.insert(rs->getString(2));
00180       refKeyId = rs->getInt(3);
00181       orderType = stringToOrder(rs->getString(4));
00182     } else {
00183       return;
00184     }
00185
00186     while (rs->next()) {
00187       if (curTable != rs->getString(1) ||
00188           refKeyId != (unsigned long)rs->getInt(3)) {
00189         // here the current rule changed, in fact either the table
00190         // or the refKeyId are different
00191         insertMapping(curTable, origKey, refKeyId, orderType);
00192         origKey.clear();
00193
00194         curTable = rs->getString(1);
00195         origKey.insert(rs->getString(2));
00196         refKeyId = rs->getInt(3);
00197         orderType = stringToOrder(rs->getString(4));
00198       } else {
00199         // here we continue to accumulate the columns in origKey
00200         origKey.insert(rs->getString(2));
00201       }
00202     } // end while
00203
00204     assert(!origKey.empty());
00205     insertMapping(curTable, origKey, refKeyId, orderType);
00206   } catch (mrdb::SQLException &e) {
00207     throw MineruleException(
00208         MR_ERROR_DATABASE_ERROR,
00209         std::string(
00210             "Cannot access the db while "
00211             "initializing a member of the OptimizerCatalogue, the reason is:") +
00212             e.what());
00213   }
00214 }
```

6.42.3.3 insertMapping()

```
void minerule::OptimizerCatalogue::Catalogue::insertMapping (
              const std::string & table,
              const KeyCols & origKeyCols,
              int refKeyId,
              OrderType orderType )  [inherited]
```

Definition at line 104 of file OptimizerCatalogue.cpp.

```
00106                           {
00107   mrdb::Connection *connection =
00108       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00109
00110   try {
00111     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00112
00113     std::string getKeyQuery =
00114         std::string("SELECT " + getSchemaInfo("tab_cols_col_name") + " " +
00115                     "FROM " + getSchemaInfo("tab_cols") + " " + "WHERE " +
00116                     getSchemaInfo("tab_cols_key_id") + "=") +
00117         Converter((long)refKey).toString();
00118
00119     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getKeyQuery));
00120     KeyCols refKeyCols;
00121
00122     while (rs->next()) {
00123       refKeyCols.insert(rs->getString(1));
00124     }
00125
00126     //      OrderType orderType = getOrderType(origKey,refKey);
00127     // (*this)[table][cols] = pair<KeyCols,OrderType>(refKeyCols,orderType);
00128     std::pair< KeyCols, std::pair<KeyCols, OrderType> > insElem(
00129         cols, std::pair<KeyCols, OrderType>(refKeyCols, orderType));
00130     (*this)[table].insert(insElem);
00131
00132   } catch (mrdb::SQLException &e) {
00133     throw MineruleException(
00134         MR_ERROR_DATABASE_ERROR,
00135         std::string("Cannot access the db while "
00136                     "initializing a member of the OptimizerCatalogue,"
00137                     "the reason is:") +
00138             e.what());
00139   }
00140 }
```

6.42.3.4 stringToOrder()

```
OptimizerCatalogue::OrderType minerule::OptimizerCatalogue::Catalogue::stringToOrder (
            const std::string & str )  [static], [inherited]
```

Definition at line 143 of file OptimizerCatalogue.cpp.

```
00143                                                          {
00144   if (str == "r")
00145     return Reversed;
00146
00147   if (str == "e")
00148     return Equal;
00149
00150   return None;
00151 }
```

6.42.4 Field Documentation

6.42.4.1 elements

```
T std::map< K, T >::elements  [inherited]
```

STL member.

6.42.4.2 keys

```
K std::map< K, T >::keys  [inherited]
```

STL member.

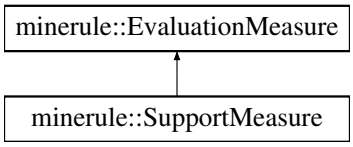
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp

6.43 minerule::EvaluationMeasure Class Reference

```
#include <EvaluationMeasure.hpp>
```

Inheritance diagram for minerule::EvaluationMeasure:



Public Types

- enum MeasureType { MONOTONE , ANTIMONOTONE , OTHER }
- enum RelOperator { LessEqual , Less , Greater , GreaterEqual }

Public Member Functions

- virtual MeasureType getMeasureType ()=0

6.43.1 Detailed Description

Definition at line 7 of file EvaluationMeasure.hpp.

6.43.2 Member Enumeration Documentation

6.43.2.1 MeasureType

```
enum minerule::EvaluationMeasure::MeasureType
```

Enumerator

MONOTONE	
ANTIMONOTONE	
OTHER	

Definition at line 13 of file EvaluationMeasure.hpp.

```
00013                {
00014     MONOTONE,
00015     ANTIMONOTONE,
00016     OTHER
00017   } MeasureType;
```

6.43.2.2 RelOperator

```
enum minerule::EvaluationMeasure::RelOperator
```

Enumerator

LessEqual	
Less	
Greater	
GreaterEqual	

Definition at line 19 of file EvaluationMeasure.hpp.

```
00019                {
00020     LessEqual,
00021     Less,
00022     Greater,
00023     GreaterEqual
00024   } RelOperator;
```

6.43.3 Member Function Documentation

6.43.3.1 getMeasureType()

```
virtual MeasureType minerule::EvaluationMeasure::getMeasureType ( )  [pure virtual]
```

Implemented in minerule::SupportMeasure.

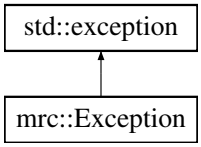
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/EvaluationMeasure.hpp

6.44 mrc::Exception Class Reference

`#include <Exception.hpp>`

Inheritance diagram for mrc::Exception:



Public Member Functions

- Exception (mrc::Results res, const std::string &msg)
- virtual ~Exception () _NOEXCEPT
- virtual const char * what () const _NOEXCEPT
- virtual mrc::Results getResultID () const

6.44.1 Detailed Description

Definition at line 29 of file Exception.hpp.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Exception()

```
mrc::Exception::Exception (
            mrc::Results res,
            const std::string & msg )  [inline]
```

Definition at line 33 of file Exception.hpp.

```
00034                                           :  result(res),errmsg(msg) { }
```

6.44.2.2 ~Exception()

```
virtual mrc::Exception::~Exception ( )  [inline], [virtual]
```

Definition at line 36 of file Exception.hpp.

```
00036 {};
```

6.44.3 Member Function Documentation

6.44.3.1 getResultID()

```
virtual mrc::Results mrc::Exception::getResultID ( ) const  [inline], [virtual]
```

Definition at line 42 of file Exception.hpp.

```
00042                                                    {
00043                         return result;
00044                 }
```

6.44.3.2 what()

```
virtual const char * mrc::Exception::what ( ) const  [inline], [virtual]
```

Definition at line 38 of file Exception.hpp.

```
00038                                                  {
00039                         return errmsg.c_str();
00040                 }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Exception.hpp

6.45 minerule::ExpressionNFCoder Class Reference

```
#include <ExpressionNFCoder.hpp>
```

Public Member Functions

- ExpressionNFCoder (InvalidConfigurationFilter &f)
- ~ExpressionNFCoder ()
- EncodedNF encode (Predicate &)

Protected Attributes

- InvalidConfigurationFilter & filter

Static Protected Attributes

- static const size_t bitsPerCell = (sizeof(EncodingBaseType)<<3)

6.45.1 Detailed Description

Definition at line 97 of file ExpressionNFCoder.hpp.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 ExpressionNFCoder()

```
minerule::ExpressionNFCoder::ExpressionNFCoder (
            InvalidConfigurationFilter & f )  [inline]
```

Definition at line 112 of file ExpressionNFCoder.hpp.

```
00112                                                 :  filter(f), buf(NULL) {
00113
00114     }
```

6.45.2.2 ~ExpressionNFCoder()

```
minerule::ExpressionNFCoder::~ExpressionNFCoder ( )  [inline]
```

Definition at line 116 of file ExpressionNFCoder.hpp.

```
00116                          {
00117       cleanBuf();
00118     }
```

6.45.3 Member Function Documentation

6.45.3.1 encode()

```
EncodedNF minerule::ExpressionNFCoder::encode (
            Predicate & ee )
```

Definition at line 28 of file ExpressionNFCoder.cpp.

```
00029     {
00030
00031     cleanBuf();
00032
00033     size_t nvars = ee.getNumVariables();
00034     VarSetEnumerator vse(nvars);
00035
00036     if( nvars > 31 ) {
00037      std::string error =
00038         "ExpressionNFCoder::encode - the current implementation "
00039         "does not support expression with more than 31 variables";
00040
00041       throw MineruleException(MR_ERROR_INTERNAL,error);
00042     }
00043
00044
00045
00046     // We are now going to evaluate the maximal storage required for the
00047     // encoding. Its value can be evaluated as nvars*numSets/bitsPerCell
00048     // Where bitsPerCells is the number of bits that each cell can hold
00049     // (which is sizeof(EncodingBaseType)*8 == sizeof(EncodingBaseType)«3);
00050     // nvars is the number of variables and numSets is the value returned
00051     // by numEnumerations (i.e., 2^numVars).
00052     // In order to avoid overflows in the integer representation of the
00053     // result, we postpone the multiplication after the division. Thanks to
00054     // this little trick, we can manage up to 31 variables (in fact the maximal
00055     // value we can hold in a size_t element is 2^32. Assuming 31 variables we
```

```
00056     // end up to the following number: (2^31/2^5) * 31, the division output
00057     // 2^26 which can be safely multiplied by 31 without causing any overflow.
00058     // Note that the calculation has been performed assuming that
00059     // sizeof(EncodingBaseType)*8 == 32, which is true in the current
00060     // implementation but should be checked when the system is ported to other
00061     // platforms or when EncodingBaseType is set to be different from uint.
00062
00063
00064     assert(sizeof(EncodingBaseType)==4);
00065     size_t numCells = (1«nvars) / bitsPerCell;
00066     numCells *= nvars;
00067     if((1«nvars)%bitsPerCell > 0)
00068       numCells++;
00069
00070     // we use malloc instead of new, because we need to use realloc later on
00071     buf = (EncodingBaseType*) calloc( numCells, sizeof(EncodingBaseType));
00072
00073     size_t cellCounter = 0;
00074     size_t elemCounter = 0;
00075
00076     while( vse++ ) {
00077       if(ee.evaluate(*vse) && !filter(*vse)) {
00078         for( size_t i=0; i<(*vse).size(); i++ ) {
00079           if((*vse).getVar(i))
00080             buf[cellCounter] |=  1«elemCounter;
00081           else
00082             buf[cellCounter] &= ~(1«elemCounter);
00083
00084           if(++elemCounter>=bitsPerCell) {
00085             cellCounter++;
00086             elemCounter=0;
00087           }
00088         }
00089       }
00090     }
00091
00092     buf = (EncodingBaseType*) realloc(buf,sizeof(EncodingBaseType) * (cellCounter+1));
00093     EncodedNF result;
00094     result.encVector = buf;
00095     result.vecSize = cellCounter+1;
00096     result.codingLen = cellCounter*bitsPerCell+elemCounter;
00097     result.numVars = ee.getNumVariables();
00098
00099     return  result;
00100   }
```

6.45.4 Field Documentation

6.45.4.1 bitsPerCell

```
const size_t minerule::ExpressionNFCoder::bitsPerCell = (sizeof(EncodingBaseType)<<3)  [static],
[protected]
```

Definition at line 99 of file ExpressionNFCoder.hpp.

6.45.4.2 filter

```
InvalidConfigurationFilter& minerule::ExpressionNFCoder::filter  [protected]
```

Definition at line 100 of file ExpressionNFCoder.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp

6.46 minerule::QueryResult::FastSorter Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2)

6.46.1 Detailed Description

Definition at line 87 of file QueryResult-header.hpp.

6.46.2 Member Function Documentation

6.46.2.1 operator()()

```
bool minerule::QueryResult::FastSorter::operator() (
            const Rule & r1,
            const Rule & r2 )  [inline]
```

Definition at line 89 of file QueryResult-header.hpp.

```
00089                                                                   {
00090                               return &r1 < &r2;
00091                       }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.47 minerule::RuleFormatter::FieldWidths Class Reference

`#include <RuleFormatter.hpp>`

Public Member Functions

- FieldWidths (size_t b, size_t h, size_t s, size_t c)

Data Fields

- size_t body
- size_t head
- size_t supp
- size_t conf

6.47.1 Detailed Description

Definition at line 31 of file RuleFormatter.hpp.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 FieldWidths()

```
minerule::RuleFormatter::FieldWidths::FieldWidths (
            size_t b,
            size_t h,
            size_t s,
            size_t c )  [inline]
```

Definition at line 38 of file RuleFormatter.hpp.

```
00038 : body(b), head(h), supp(s), conf(c) {}
```

6.47.3 Field Documentation

6.47.3.1 body

```
size_t minerule::RuleFormatter::FieldWidths::body
```

Definition at line 33 of file RuleFormatter.hpp.

6.47.3.2 conf

```
size_t minerule::RuleFormatter::FieldWidths::conf
```

Definition at line 36 of file RuleFormatter.hpp.

6.47.3.3 head

```
size_t minerule::RuleFormatter::FieldWidths::head
```

Definition at line 34 of file RuleFormatter.hpp.

6.47.3.4 supp

```
size_t minerule::RuleFormatter::FieldWidths::supp
```

Definition at line 35 of file RuleFormatter.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp

6.48 minerule::FileUtils Class Reference

```
#include <FileUtils.hpp>
```

Static Public Member Functions

- static bool fileExists (const std::string &filename)

6.48.1 Detailed Description

Definition at line 22 of file FileUtils.hpp.

6.48.2 Member Function Documentation

6.48.2.1 fileExists()

```
bool minerule::FileUtils::fileExists (
            const std::string & filename )  [static]
```

Definition at line 22 of file FileUtils.cpp.

```
00022                                                       {
00023                 struct stat trash;
00024                 if( stat(fname.c_str(),&trash)==0 )
00025                         return true;
00026                 else
00027                         return false;
00028         }
```

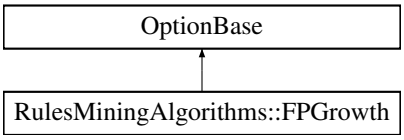
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/FileUtils.hpp
- /Users/esposito/Software/minerule/src/Utils/FileUtils.cpp

6.49 RulesMiningAlgorithms::FPGrowth Class Reference

`#include <rulemining.hpp>`

Inheritance diagram for RulesMiningAlgorithms::FPGrowth:



Public Types

- enum FPAlgoType { Original , SingleReorder }

Public Member Functions

- virtual ~FPGrowth ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- FPAlgoType getAlgoType () const
- void setAlgoType (FPAlgoType type)

6.49.1 Detailed Description

Definition at line 57 of file rulemining.hpp.

6.49.2 Member Enumeration Documentation

6.49.2.1 FPAlgoType

`enum RulesMiningAlgorithms::FPGrowth::FPAlgoType`

Enumerator

Original	
SingleReorder	

Definition at line 59 of file rulemining.hpp.

```
00059                     {
00060             Original,
00061             SingleReorder
00062           } FPAlgoType;
```

6.49.3 Constructor & Destructor Documentation

6.49.3.1 ~FPGrowth()

```
virtual RulesMiningAlgorithms::FPGrowth::~FPGrowth ( )  [inline], [virtual]
```

Definition at line 67 of file rulemining.hpp.

```
00067 {};
```

6.49.4 Member Function Documentation

6.49.4.1 className()

```
virtual std::string RulesMiningAlgorithms::FPGrowth::className ( ) const  [inline], [virtual]
```

Definition at line 68 of file rulemining.hpp.

```
00068                                                {
00069             return "fpgrowth";
00070           }
```

6.49.4.2 getAlgoType()

```
FPAlgoType RulesMiningAlgorithms::FPGrowth::getAlgoType ( ) const  [inline]
```

Definition at line 75 of file rulemining.hpp.

```
00075                                          {
00076             return algoType;
00077           }
```

6.49.4.3 setAlgoType()

```
void RulesMiningAlgorithms::FPGrowth::setAlgoType (
            FPAlgoType type )  [inline]
```

Definition at line 78 of file rulemining.hpp.

```
00078                                            {
00079             algoType = type;
00080           }
```

6.49.4.4 setOption()

```
virtual void RulesMiningAlgorithms::FPGrowth::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

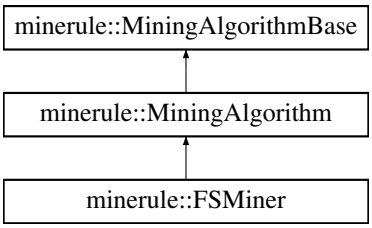
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp

6.50 minerule::FSMiner Class Reference

`#include <FSMiner.hpp>`

Inheritance diagram for minerule::FSMiner:



Public Member Functions

- FSMiner (const OptimizedMinerule &mr, const AlgorithmsOptions &opts)
- virtual ~FSMiner ()
- virtual void mineRules ()
- virtual bool canHandleMinerule () const
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual void initialize ()
- virtual void execute ()
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- const OptimizedMinerule & minerule

6.50.1 Detailed Description

Definition at line 25 of file FSMiner.hpp.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 FSMiner()

```
minerule::FSMiner::FSMiner (
            const OptimizedMinerule & mr,
            const AlgorithmsOptions & opts )  [inline]
```

Definition at line 36 of file FSMiner.hpp.

```
00036                                                                    :
00037       MiningAlgorithm(mr), options(opts), statement(NULL) {}
```

6.50.2.2 ~FSMiner()

```
virtual minerule::FSMiner::~FSMiner ( )  [inline], [virtual]
```

Definition at line 39 of file FSMiner.hpp.

```
00039 {}
```

6.50.3 Member Function Documentation

6.50.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                    {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.50.3.2 canHandleMinerule()

```
virtual bool minerule::FSMiner::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 43 of file FSMiner.hpp.

```
00043                                       {
00044       return minerule.getParsedMinerule().miningTask == MTMineSequences;
00045     }
```

6.50.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                   {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.50.3.4 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                   {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.50.3.5 mineRules()

```
void minerule::FSMiner::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 61 of file FSMiner.cpp.

```
00061                   {
00062                 MRLogPush("This is FSMiner. Starting...");
00063 
00064                 prepareData();
00065                 FSTree* h=new FSTree(minerule);
00066 
00067                 mrdb::ResultSet* rs = statement->executeQuery();
00068                 MRLogPush("Building frequent bigram link table...");
00069                 h->createLinkSTable(rs, rowDes);
00070                 MRLogPop();
00071 
00072                 h->setThreshold(options.getSupport());
00073                 MRLog() << "Support threshold is:" << h->getThreshold() << std::endl;
00074 
00075                 MRLogPush("Building unfrequent bigram link table...");
00076                 h->createLinkNSTable();
00077                 MRLogPop();
00078 
00079                 delete rs;
00080                 rs = statement->executeQuery();
00081 
00082                 MRLogPush("Building FSTree...");
00083                 h->construct_Tree(rs, rowDes);
00084                 MRLogPop();
00085 
00086                 delete rs;
00087 
00088                 MRLogPush("Starting core mining...");
00089                 h->mine();
00090                 MRLogPop();
00091 
00092                 MRLogPop();
00093 
00094                 std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00095                 std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* ris = h->getResult();
00096         //std::string s=""
00097                 std::cout<<"ris->size "<<ris->size()<<std::endl;
00098                 MRLogPush(std::string("contiguos frequent sequence find:
       ")+Converter(ris->size()).toString());
00099                 for (res_it=ris->begin();res_it!=ris->end();res_it++){
00100                         FSTreeSequence temp=(*res_it).first;
00101                         MRLogPush(temp.toStdString());
00102                 }
00103 
00104 
00105         // thrashing the thrashable
00106                 delete ris;
00107                 delete h;
00108                 delete statement;
00109         }
```

6.50.3.6 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.50.3.7 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::FSMiner::sourceTableRequirements ( ) const  [inline],
[virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 47 of file FSMiner.hpp.

```
00047                                                                                {
00048                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00049                 };
```

6.50.4 Field Documentation

6.50.4.1 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/FSMiner.hpp
- /Users/esposito/Software/minerule/src/Algorithms/FSMiner.cpp

6.51 minerule::FSTree Class Reference

```
#include <FSTree.hpp>
```

Public Member Functions

- FSTree (const OptimizedMinerule &opt)
- void setThreshold (double t)
- int getN_nodi ()
- void resetN_nodi ()
- std::vector< FSTreeSequence * > * fraziona (FSTreeSequence *s)
- void createLinkSTable (mrdb::ResultSet *, const SourceRowColumnIds &)
- void createLinkNSTable ()
- void insertLink (FSTreeSequence *s)
- void addResult (FSTreeNode *n)
- int countPath (FSTreeNode *n)
- void stampa (FSTreeNode *n)
- void addResults (std::vector< FSTreeNode * > *vec)
- void mine ()
- FSTreeNode * appendChild (FSTreeNode *r, const ItemType &t)
- void addList (FSTreeNode *father, FSTreeNode *children)
- void insertTree (FSTreeNode *r, FSTreeSequence *t)
- void construct_Tree (mrdb::ResultSet *, const SourceRowColumnIds &)
- ~FSTree ()
- double getThreshold ()
- std::map< FSTreeSequence, int, FSTreeSequence::less_sequence > * getResult ()

6.51.1 Detailed Description

this class implement the algorithm of FS-MINER (without the incremental part)

Definition at line 34 of file FSTree.hpp.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 FSTree()

```
minerule::FSTree::FSTree (
            const OptimizedMinerule & opt )  [inline]
```

the constructor of this structure

Parameters

in	the name of source file

Definition at line 87 of file FSTree.hpp.

```
00087                                       :
00088     root(new FSTreeNode()),
00089     num(0),
00090     threshold(0) {
00091     root= new FSTreeNode();
00092     num=0;
00093     threshold=0;
00094
00095     max_length = opt.getParsedMinerule().bodyCardinalities.getMax();
00096     link_S= new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00097     link_NS=new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00098     result = new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00099   }
```

6.51.2.2 ~FSTree()

```
minerule::FSTree::~FSTree ( )
```

the destructor of the FSTree

Definition at line 344 of file FSTree.cpp.

```
00344                  {
00345   //cout«"sono nel distruttore di FSTree"«std::endl;
00346   delete root;
00347   //    std::cout«"esco dal distruttore di FSTree"«std::endl;
00348   delete link_NS;
00349   delete link_S;
00350
00351   //delete s_result;
00352   //delete m_list;
00353
00354   //delete s_result;
00355   delete result;
00356
00357 }
```

6.51.3 Member Function Documentation

6.51.3.1 addList()

```
void minerule::FSTree::addList (
            FSTreeNode * father,
            FSTreeNode * children )
```

this method add a new pointer to a node, in m_lsit, from the sequence of two elements that is formed with thestd::string of the father plus thestd::string of the children

Parameters

father	a node that contains the firststd::string of the sequence of two elements
children	a node that contains the secondstd::string of the sequence of two elements

Definition at line 278 of file FSTree.cpp.

```
00278                                                       {
00279   //creo la sequenza lunga 2
00280   FSTreeSequence* m_string=new FSTreeSequence();
00281   m_string->add(father->getLabel());
00282   m_string->add(children->getLabel());
00283   //se la sequenza ha come father la radice allora esco dal metodo
00284   if (father==root) {
00285     delete m_string;
00286     return;
00287   }
00288   //se l'elemento non e gia presente lo aggiungo e creo il collegamento
00289   // inserendo il riferimento al dodo
00290   //m_list->add(m_string,children);
00291   if (m_list[*m_string]==0)
00292     m_list[*m_string]=new std::vector<FSTreeNode*>();
00293   (m_list[*m_string])->push_back(children);
00294   delete m_string;
00295 }
```

6.51.3.2 addResult()

```
void minerule::FSTree::addResult (
            FSTreeNode * n )
```

this method read a branch of the tree from the bottom and add all the sequence that it reads in the set of possible frequent sequence

Parameters

n	the node from i starting to read a possible frequent sequent, i start to read from this node a sequence with size 2, after a sequence with size 3, ecc..

Definition at line 194 of file FSTree.cpp.

```
00194                                   {
00195   FSTreeSequence* s;
00196   //setto il primo UpperLimit per prendere come prima sequenza quella lunga 2
00197   FSTreeNode* UpperLimit=n->getParent();
```

```
00198   UpperLimit=UpperLimit->getParent();
00199   FSTreeNode* temp;
00200   int c = countPath(n);
00201   temp=n;
00202   s=new FSTreeSequence();
00203   //inserisco l'elemento coda presente in tutte le sottosequenze di questo cammino
00204   s->insertHead(temp->getLabel());
00205   if (c>=max_length && max_length !=0)
00206     c=max_length-1;
00207
00208   for (int i=0;i<c;i++){//prendo tutte le sequenze di lunghezza >=2
00209     temp=temp->getParent();
00210     s->insertHead(temp->getLabel());
00211     (*result)[*s]=(*result)[*s]+n->getCount();
00212     UpperLimit=UpperLimit->getParent();//estendo l' UpperLimit per prendere la sequenza lunga |s|+1
00213   }
00214   delete s;
00215 }
```

6.51.3.3 addResults()

```
void minerule::FSTree::addResults (
            std::vector< FSTreeNode * > * vec )
```

See also

addResult(node* n) this method call the method addResult for each element in the vector of node, the vector of node is in relation with a frequent couple in the m_list

Parameters

vec	the vector of node in relation with a frequent couple in the m_list

Definition at line 232 of file FSTree.cpp.

```
00232                                                   {
00233   FSTreeNode* temp;
00234   for (size_t i=0;i<vec->size();i++)    {
00235     temp= (*vec)[i];//inserisco ogni elemento del vettore nel risultato
00236     addResult(temp);
00237   }
00238 }
```

6.51.3.4 appendChild()

```
FSTreeNode * minerule::FSTree::appendChild (
            FSTreeNode * r,
            const ItemType & t )
```

See also

addList(node* father, node* children) this method append a child to node r, the node creates contains thestd::string t

Parameters

r	the father of the node that the method create
t	thestd::string that is contained in the new node

Returns

a pointer to the new node created

Definition at line 300 of file FSTree.cpp.

```
00300                                                                    {
00301   bool notFound=1;
00302   //int i=0;
00303   FSTreeNode* nod = NULL;
00304   std::vector<FSTreeNode*>* ve = r->getChild();
00305   //Cerco l'item t della sequenza tra i figli di r
00306   for (size_t i=0;i<ve->size()&&notFound;i++){
00307     nod = (*ve)[i];
00308     if (nod->getLabel()==t){
00309       notFound=0;
00310     }
00311   }
00312   //se non l'ho trovata creo un nuovo dodo e l'ho aggiungo all albero
00313   if (notFound){
00314     //creo il nuovo nodo
00315     FSTreeNode* children = new FSTreeNode(t);
00316     n_nodi++;
00317     children->setParent(r);
00318     //aggiungo children ai figli di r
00319     std::vector<FSTreeNode*>* temp=r->getChild();
00320     temp->push_back(children);
00321
00322     children->setCount(children->getCount()+1);
00323     /*formo una sequenza e aggiungo la sequenza
00324       alla lista che attraversa l'albero e che ha come riferimento iniziale la std::mappa
00325       il metodo addList controlla che non si formino sequenze lunghe uno e controlla anche
00326       quando si arriva alla radice
00327     */
00328
00329     addList(r,children);
00330
00331     //ritorno il puntatore al dodo creato
00332     return children;
00333   }
00334   //se l'ho trovato incremento il contatore di quello trovato
00335   else{
00336     nod->setCount(nod->getCount()+1);
00337     return nod;
00338   }
00339
00340
00341 }
```

6.51.3.5 construct_Tree()

```
void minerule::FSTree::construct_Tree (
            mrdb::ResultSet * rs,
            const SourceRowColumnIds & rowDes )
```

construct the tree that contains a compressed rappresentation of the dataSet

See also

fraziona(sequence ∗s)

insertTree(node∗ r, sequence∗ t)

Definition at line 90 of file FSTree.cpp.

```
00090                                                                    {
00091   std::vector<FSTreeSequence*>* collezione;
00092   FSTreeSequence* seq;
00093   size_t num=0;
00094   rs->next();
00095   while (!rs->isAfterLast()){
00096     num++;
00097     seq=new FSTreeSequence();
00098     seq->read(rs,rowDes);
```

```
00099
00100     if (seq->size()>1){
00101       collezione=fraziona(seq);
00102
00103       for (size_t i=0; i<collezione->size();i++){
00104         if (((*collezione)[i])->size()>1){
00105           insertTree(root,(*collezione)[i]);
00106         }
00107         delete (*collezione)[i];
00108       }
00109       delete collezione;
00110     }
00111     delete seq;
00112   }
00113 }
```

6.51.3.6 countPath()

```
int minerule::FSTree::countPath (
            FSTreeNode * n )
```

See also

addResults(std::vector<node*>* vec) this method counts how many item there are from the node n and the root, it counts the possible sequence that the method addResult can extract from a branch

Parameters

n	the node from which i count the number of elements

Returns

the number of node between this node and the root

Definition at line 182 of file FSTree.cpp.

```
00182                                   {
00183   //la radice di tutte le sequenze dell'albero e ad altezza -1
00184   //il primo elemento e ad altezza 0
00185   FSTreeNode* temp= n;
00186   int c=0;
00187   while (temp->getParent()!=root){
00188     c++;
00189     temp=temp->getParent();
00190   }
00191   return c;
00192 }
```

6.51.3.7 createLinkNSTable()

```
void minerule::FSTree::createLinkNSTable ( )
```

insert all no frequent couple in the map link_NS

Definition at line 47 of file FSTree.cpp.

```
00047                                {
00048   //COSTRUISCO MAPPA DEI LINK FREQUENTI E DEI LINK NON FREQUENTI RISPETTO ALLA THRESHOLD
00049   //std::vector<sequence*>::iterator it;
```

```
00050   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator it;
00051   int count;
00052   for (it=link_S->begin();it!=link_S->end();it++){
00053
00054     count = (*it).second;
00055     if (count<threshold){
00056       (*link_NS)[(*it).first]=count;
00057     }
00058   }
00059 }
```

6.51.3.8 createLinkSTable()

```
void minerule::FSTree::createLinkSTable (
            mrdb::ResultSet * rs,
            const SourceRowColumnIds & rowDes )
```

insert all couple present in the dataSet in the map link_S

Definition at line 61 of file FSTree.cpp.

```
00061                                                                                {
00062   FSTreeSequence* input = new FSTreeSequence();
00063   rs->next();
00064  while(!rs->isAfterLast()) {
00065     input->read(rs,rowDes);
00066     if (input->size()>1){
00067       insertLink(input);
00068     }
00069
00070
00071     input->svuota();
00072     num++;
00073     //canRead=rs->next();
00074   }
00075
00076   MRLog() « link_S->size() «" bigrams have been read." « std::endl;
00077   std::cout« link_S->size()«" bigrams have been read." « std::endl;
00078   delete input;
00079 }
```

6.51.3.9 fraziona()

```
std::vector< FSTreeSequence * > * minerule::FSTree::fraziona (
            FSTreeSequence * s )
```

this method decompose a sequence in a number of sequences, this number is unknow. With the map of non frequent couple, this method decompose the input sequence s in more sequence. The characteristics of each of these sequences is that:

- the cardinality of each sequence is major of one
- each sequence is a possible frequent sequence, because it no contains an infrequent couple

Parameters

s	a pointer of a sequence

Returns

a vector of possible frequent sequence

Definition at line 114 of file FSTree.cpp.

```
00114                                                                    {
00115   std::vector<int> vec;
00116   std::vector<FSTreeSequence*>* res = new std::vector<FSTreeSequence*>();
00117
00118
00119   //creo un vector con i riferimenti alla posizione in cui ho trovato l'inizio delle
00120   //seq lunghe due non a supp sufficiente
00121   FSTreeSequence* te;
00122   for (size_t i =0;i<s->size()-1;i++){
00123     te=new FSTreeSequence(*s,i,2);
00124     if ((*link_NS)[*te]!=0)
00125       vec.push_back(i);
00126     delete te;
00127   }
00128   //ordino il vettore di riferimenti all'inizio di sequenze non frequenti
00129   sort(vec.begin(),vec.end());
00130
00131   int nchar;
00132   //se non ci sono sequenze e la sequenza e piu lunga di 1 allora ritorno un vector in cui ho inserito
      un solo
00133   //elemento: la sequenza iniziale
00134
00135   if (vec.size()==0){
00136     if (s->size()>1){
00137       FSTreeSequence* sost=new FSTreeSequence(*s);
00138       res->push_back(sost);
00139     }
00140     return res;
00141   }
00142
00143   FSTreeSequence *temp;
00144   //inserisco la prima sottosequenza nel vettore risultato
00145   temp=new FSTreeSequence(*s,0,vec[0]+1);
00146
00147   res->push_back(temp);
00148   //se il vector vec e piu lungo di 1
00149   for (size_t i=0;i<vec.size()-1;i++){
00150     //skip di quelle posizioni consecutive
00151     if (vec[i]+1!=vec[i+1]){
00152       //numero di caratteri da prendere
00153       nchar=vec[i+1]-vec[i];
00154       //creo la sottosequenza che parte da vec[i]1 ed e lunga nchar
00155       //lunga almeno 2
00156       if (nchar>1){
00157         temp=new FSTreeSequence(*s,vec[i]+1,nchar);
00158         res->push_back(temp);
00159       }
00160     }
00161   }
00162   nchar=s->size()-(vec[vec.size()-1]+1);
00163   //se non mi trovo gia alla fine della sequenza
00164   //faccio lo stesso per l'ultima sottosequenza
00165   if (nchar>1){
00166     FSTreeSequence* last=new FSTreeSequence(*s,vec[vec.size()-1]+1,nchar);
00167     res->push_back(last);
00168
00169   }
00170   return res;
00171 }
```

6.51.3.10 getN_nodi()

```
int minerule::FSTree::getN_nodi ( )
```

Returns

the number of the node in the tree

Definition at line 178 of file FSTree.cpp.

```
00178                        {
00179   return n_nodi;
00180 }
```

6.51.3.11 getResult()

```
std::map< FSTreeSequence, int, FSTreeSequence::less_sequence > * minerule::FSTree::getResult (
)
```

Definition at line 364 of file FSTree.cpp.

```
00364                                                                    {
00365   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00366   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence> t(*result);
00367   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* t2 = new
       std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00368   int count;
00369   for (res_it=t.begin();res_it!=t.end();res_it++){
00370     count=(*res_it).second;
00371     if (count>=threshold){
00372       FSTreeSequence temp=(*res_it).first;
00373       (*t2)[temp]=count;
00374     }
00375   }
00376   return t2;
00377 }
```

6.51.3.12 getThreshold()

```
double minerule::FSTree::getThreshold ( )
```

Returns

the threshold,as number , that was set

Definition at line 359 of file FSTree.cpp.

```
00359                          {
00360   return threshold;
00361 }
```

6.51.3.13 insertLink()

```
void minerule::FSTree::insertLink (
            FSTreeSequence * s )
```

this method decompose a sequence in all the couple that it contains for adding the couple in the link_S

See also

createLinkSTable()

Parameters

s	the sequence from which i want extract couples

Definition at line 81 of file FSTree.cpp.

```
00081                                        {
00082   FSTreeSequence* out;
```

```
00083   for (size_t i=0;i<s->size()-1;i++){
00084     out=new FSTreeSequence(*s,i,2);
00085     (*link_S)[*out]=(*link_S)[*out]+1;
00086     delete out;
00087   }
00088 }
```

6.51.3.14 insertTree()

```
void minerule::FSTree::insertTree (
            FSTreeNode * r,
            FSTreeSequence * t )
```

this method insert, in the tree, a sequence t starting from the node r

See also

appendChild(node* r, std::string t)

Definition at line 29 of file FSTree.cpp.

```
00029                                                    {
00030   //se la sequenza e composta da un solo elemento
00031   //la inserisco come figlio
00032
00033   if (t->size()==1){
00034     appendChild(r,t->removeHead());
00035   }
00036   else{
00037     //altrimenti creo un nuovo nodo temp con il primo item della sequenza
00038     //creo il collegamento nella lista e inserisco il nodo con appendChild
00039     //richiamo insertTree sul resto della sequenza
00040
00041     FSTreeNode* nod = appendChild(r,t->removeHead());
00042
00043     insertTree(nod,t);
00044   }
00045 }
```

6.51.3.15 mine()

```
void minerule::FSTree::mine ( )
```

See also

addResults(std::vector<node*>* vec)

addResult(node* n) this method extract the frequent sequence from the tree structure

Definition at line 240 of file FSTree.cpp.

```
00240                   {
00241   //threshold=tsl*num/100;
00242
00243   // std::vector<sequence*>::iterator m_it;
00244   std::map<FSTreeSequence, std::vector<FSTreeNode*>*, FSTreeSequence::less_sequence>::iterator m_it;
00245   //itero sulla mappa che contiene le sequenze lunghe 2 e i puntatori alle sequenze
00246   std::cout<<"prima del ciclo di mine"<<std::endl;
00247   std::cout<<"nella m_list ci sono "<<m_list.size()<<" sequenze"<<std::endl;
00248   //cout<<"nella link_NS ci sono "<<link_NS->size()<<" sequenze"<<std::endl;
00249   std::cout<<"nella link_S ci sono "<<link_S->size()<<" sequenze"<<std::endl;
00250
00251   for (m_it=m_list.begin();m_it!=m_list.end();m_it++){
00252     //addResults(m_list->getList(*(*m_it)));
```

```
00253     addResults((*m_it).second);
00254   }
00255   std::cout«"possibili stringhe frequenti da scremare ulteriormente result.size()"«std::endl;
00256   std::cout«result->size()«std::endl;
00257   //std::vector<sequence*>::iterator res_it;
00258   //std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00259   // itero sulla mappa che contiene il risultato per stamparlo
00260   //int count=0;
00261
00262   //sequenceCount* t= new sequenceCount(*result);
00263   /*
00264     std::map<sequence,int,sequence::less_sequence>* t = new
      std::map<sequence,int,sequence::less_sequence>(*result);
00265     for (res_it=t->begin();res_it!=t->end();res_it++){
00266     count=(*res_it).second;
00267     if (count>=threshold){
00268     sequence temp=(*res_it).first;
00269     temp.stampa();
00270     std::cout«": "«count«std::endl;
00271     }
00272     }
00273     delete t;*/
00274   std::cout«"threshold: "«threshold«std::endl;
00275 }
```

6.51.3.16 resetN_nodi()

```
void minerule::FSTree::resetN_nodi ( )
```

reset the variable that contains the numbers of the node in tree

Definition at line 174 of file FSTree.cpp.

```
00174                          {
00175   n_nodi=0;
00176 }
```

6.51.3.17 setThreshold()

```
void minerule::FSTree::setThreshold (
            double t )  [inline]
```

this method permit to set the threshold

Parameters

t	the threshold as percentage

Definition at line 104 of file FSTree.hpp.

```
00104                                {
00105     threshold=t*num;
00106     //threshold=t;
00107   }
```

6.51.3.18 stampa()

```
void minerule::FSTree::stampa (
            FSTreeNode * n )
```

Definition at line 219 of file FSTree.cpp.

```
00219                                {
00220   if (n!=root)
00221     std::cout<<n->getLabel()<<":"<<n->getCount()<<" ";
00222   std::vector<FSTreeNode*>* temp= n->getChild();
00223   if (temp->size()==0)//se non ha figli stampo un \n e poi esco dal metodo
00224     std::cout<<std::endl;
00225   else {
00226     for (size_t i=0;i<temp->size();i++){
00227       stampa((*temp)[i]);
00228     }
00229   }
00230 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTree.hpp
- /Users/esposito/Software/minerule/src/Algorithms/FSTree.cpp

6.52 minerule::FSTreeNode Class Reference

`#include <FSTreeNode.hpp>`

Public Member Functions

- FSTreeNode (const ItemType &I)
- FSTreeNode ()
- FSTreeNode (FSTreeNode *p)
- ~FSTreeNode ()
- const ItemType & getLabel ()
- std::vector< FSTreeNode * > * getChild ()
- int getCount ()
- FSTreeNode * getParent ()
- void setCount (int n)
- void setParent (FSTreeNode *s)
- void insertChild (FSTreeNode *s)

6.52.1 Detailed Description

the node of the tree

Definition at line 27 of file FSTreeNode.hpp.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 FSTreeNode() [1/3]

```
minerule::FSTreeNode::FSTreeNode (
            const ItemType & I )
```

constructor

Parameters

l	the label of this node

Definition at line 22 of file FSTreeNode.cpp.

```
00022                                       {
00023   child = new std::vector<FSTreeNode*>();
00024   count=0;
00025   label=l;
00026 }
```

6.52.2.2 FSTreeNode() [2/3]

```
minerule::FSTreeNode::FSTreeNode ( )
```

constructor

Definition at line 28 of file FSTreeNode.cpp.

```
00028                           {
00029   child = new std::vector<FSTreeNode*>();
00030   count=0;
00031 }
```

6.52.2.3 FSTreeNode() [3/3]

```
minerule::FSTreeNode::FSTreeNode (
            FSTreeNode * p )
```

constructor

Parameters

p	the father of this node

Definition at line 33 of file FSTreeNode.cpp.

```
00033                                      {
00034   parent = p;
00035   count=0;
00036   child = new std::vector<FSTreeNode*>();
00037 }
```

6.52.2.4 ~FSTreeNode()

```
minerule::FSTreeNode::~FSTreeNode ( )
```

the destructor

Definition at line 39 of file FSTreeNode.cpp.

```
00039                           {
```

```
00040   FSTreeNode* temp;
00041   for (size_t i=0;i<child->size();i++){
00042     temp=(*child)[i];
00043     delete temp;
00044   }
00045   delete child;
00046
00047 }
```

6.52.3 Member Function Documentation

6.52.3.1 getChild()

```
std::vector< FSTreeNode * > * minerule::FSTreeNode::getChild ( )
```

Returns

a pointer to the vector of the child node

Definition at line 53 of file FSTreeNode.cpp.

```
00053                                          {
00054   return child;
00055 }
```

6.52.3.2 getCount()

```
int minerule::FSTreeNode::getCount ( )
```

Returns

value of variable count

Definition at line 57 of file FSTreeNode.cpp.

```
00057                          {
00058   return count;
00059 }
```

6.52.3.3 getLabel()

```
const ItemType & minerule::FSTreeNode::getLabel ( )
```

Returns

the label of this node

Definition at line 49 of file FSTreeNode.cpp.

```
00049                                  {
00050   return label;
00051 }
```

6.52.3.4 getParent()

```
FSTreeNode * minerule::FSTreeNode::getParent ( )
```

Returns

a pointer to the parent node

Definition at line 61 of file FSTreeNode.cpp.

```
00061                                    {
00062   return parent;
00063 }
```

6.52.3.5 insertChild()

```
void minerule::FSTreeNode::insertChild (
            FSTreeNode * s )
```

Parameters

s	the node that i want insert as a children of this node

Definition at line 73 of file FSTreeNode.cpp.

```
00073                                          {
00074   child->push_back(s);
00075 }
```

6.52.3.6 setCount()

```
void minerule::FSTreeNode::setCount (
            int n )
```

Parameters

n	the count of this node

Definition at line 65 of file FSTreeNode.cpp.

```
00065                                   {
00066   count = n;
00067 }
```

6.52.3.7 setParent()

```
void minerule::FSTreeNode::setParent (
            FSTreeNode * s )
```

Parameters

s	the parent node

Definition at line 69 of file FSTreeNode.cpp.

```
00069                                       {
00070   parent = s;
00071 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeNode.hpp
- /Users/esposito/Software/minerule/src/Algorithms/FSTreeNode.cpp

6.53 minerule::FSTreeSequence Class Reference

`#include <FSTreeSequence.hpp>`

Data Structures

- struct less_sequence

Public Types

- typedef std::list< ItemType > ItemVector

Public Member Functions

- FSTreeSequence ()
- FSTreeSequence (const FSTreeSequence &in)
- FSTreeSequence (const FSTreeSequence &in, size_t start, size_t n_item)
- ~FSTreeSequence ()
- ItemVector * getFSTreeSequence ()
- void add (const ItemType &s)
- void stampa ()
- size_t size ()
- ItemType removeHead ()
- void insertHead (const ItemType &s)
- std::string toStdString ()
- void svuota ()
- bool operator== (FSTreeSequence &s)
- void read (mrdb::ResultSet *rs, const SourceRowColumnIds &)
- bool operator< (const FSTreeSequence &in) const

6.53.1 Detailed Description

The class that contains the concept of a FSTreeSequence, a sequenze is a ordered list of item

Definition at line 29 of file FSTreeSequence.hpp.

6.53.2 Member Typedef Documentation

6.53.2.1 ItemVector

```
typedef std::list<ItemType> minerule::FSTreeSequence::ItemVector
```

Definition at line 31 of file FSTreeSequence.hpp.

6.53.3 Constructor & Destructor Documentation

6.53.3.1 FSTreeSequence() [1/3]

```
minerule::FSTreeSequence::FSTreeSequence ( )  [inline]
```

the constructor of an object of type FSTreeSequence

Definition at line 55 of file FSTreeSequence.hpp.

```
00055                         {
00056       seq=new ItemVector();
00057     };
```

6.53.3.2 FSTreeSequence() [2/3]

```
minerule::FSTreeSequence::FSTreeSequence (
            const FSTreeSequence & in )  [inline]
```

A copy constructor

Parameters

in	a FSTreeSequence object

Definition at line 63 of file FSTreeSequence.hpp.

```
00063                                          {
00064       seq=new ItemVector();
00065       *seq = *in.seq;
00066     };
```

6.53.3.3 FSTreeSequence() [3/3]

```
minerule::FSTreeSequence::FSTreeSequence (
            const FSTreeSequence & in,
```

```
            size_t start,
            size_t n_item )
```

This constructor allows to create a new FSTreeSequence object that contains only n_item from FSTreeSequence in starting from the item in position start

Parameters

in	the FSTreeSequence from i want to copy n_item items
start	the position from start to copy items
n_item	the numbers of item that i want to copy

Definition at line 40 of file FSTreeSequence.cpp.

```
00040                                                                                                {
00041     seq= new ItemVector();
00042     ItemVector::const_iterator it = in.seq->begin();
00043
00044     for(size_t i=0; i<start; ++i) {
00045       ++it;
00046     }
00047
00048
00049     for (size_t i=start;(i<in.seq->size())&&(i<start+n_item);++i) {
00050       seq->push_back( *it );
00051       ++it;
00052     }
00053   }
```

6.53.3.4 ~FSTreeSequence()

```
minerule::FSTreeSequence::~FSTreeSequence ( )
```

The distructor of a FSTreeSequence

Definition at line 32 of file FSTreeSequence.cpp.

```
00032                                    {
00033     delete seq;
00034   }
```

6.53.4 Member Function Documentation

6.53.4.1 add()

```
void minerule::FSTreeSequence::add (
            const ItemType & s )
```

with this method you can add an item at the bottom of a FSTreeSequence

Parameters

the	item s that you want to append at the FSTreeSequence

Definition at line 36 of file FSTreeSequence.cpp.

```
00036                                                {
00037     seq->push_back(s);
00038   }
```

6.53.4.2 getFSTreeSequence()

```
FSTreeSequence::ItemVector * minerule::FSTreeSequence::getFSTreeSequence ( )
```

this method return a pointer to the vector that contains the items as string

Returns

a pointer of a vector ofstd::string that contains items

Definition at line 27 of file FSTreeSequence.cpp.

```
00027                                                          {
00028 #warning CONTROLLARE CHE LA COPIA SIA VERAMENTE UTILE (NON SI PUO SEMPLICEMENTE RESTITUIRE seq?).
00029     return new ItemVector(*seq);
00030   }
```

6.53.4.3 insertHead()

```
void minerule::FSTreeSequence::insertHead (
            const ItemType & s )
```

insert an item in the front of the FSTreeSequence

Parameters

s	the item that i want insert

Definition at line 72 of file FSTreeSequence.cpp.

```
00072                                                {
00073     seq->insert(seq->begin(),s);
00074   }
```

6.53.4.4 operator<()

```
bool minerule::FSTreeSequence::operator< (
            const FSTreeSequence & in ) const
```

operator<

Returns

true if this is less than in otherwise return false, a FSTreeSequence is less than another if the size is less than another in the case the size is equal i consider the lexographyc order

Definition at line 127 of file FSTreeSequence.cpp.

```
00127                                                                    {
00128     return lexicographical_compare(seq->begin(),
00129                                    seq->end(),
00130                                    in.seq->begin(),
00131                                    in.seq->end());
00132 /*    ItemVector* vec1 =seq;
00133     ItemVector* vec2 = in.seq;
00134     if (vec1->size()<vec2->size())
00135       return true;
00136     if (vec1->size()>vec2->size())
00137       return false;
00138
00139     size_t i=0;
00140     for (;i<vec1->size()&&((*vec1)[i]==(*vec2)[i]);i++);
00141     // for senza corpo
00142
00143     if (i==vec1->size()) return false;
00144     else return (*vec1)[i]<(*vec2)[i]; */
00145   }
```

6.53.4.5 operator==()

```
bool minerule::FSTreeSequence::operator== (
             FSTreeSequence & s )
```

the operator ==

Returns

true if the two FSTreeSequence are equal (same element and same size), false otherwise

Definition at line 95 of file FSTreeSequence.cpp.

```
00095                                                {
00096     return *seq==*s.seq;
00097     /*
00098
00099     if (seq->size()!=s.size())
00100       return false;
00101
00102     ItemVector* vec = s.getFSTreeSequence();
00103     for (size_t i=0; i < vec->size();i++)
00104       if ((*vec)[i]!=(*seq)[i]){
00105         delete vec;
00106         return false;
00107       }
00108     delete vec;
00109     return true; */
00110
00111   }
```

6.53.4.6 read()

```
void minerule::FSTreeSequence::read (
             mrdb::ResultSet * rs,
             const SourceRowColumnIds & rowDes )
```

decompose astd::string in a FSTreeSequence every item in thestd::string must be separated with a space

Parameters

s	thestd::string that you want decompose

Definition at line 113 of file FSTreeSequence.cpp.

```
00113                                                                                                   {
00114         SourceRow hbsr(rs,rowDes);
00115         ItemType gid;
00116         if(!rs->isAfterLast())
00117                 gid = hbsr.getGroup();
00118         while (!rs->isAfterLast() && gid == hbsr.getGroup()) {
00119                 seq->push_back( hbsr.getBody() );
00120
00121                 rs->next();
00122                 if(!rs->isAfterLast())
00123                         hbsr.init(rs,rowDes);
00124         }
00125   }
```

6.53.4.7 removeHead()

```
ItemType minerule::FSTreeSequence::removeHead ( )
```

remove the first element in the FSTreeSequence

Returns

the first element in the FSTreeSequence

Definition at line 66 of file FSTreeSequence.cpp.

```
00066                                                {
00067     ItemType res = seq->front();
00068     seq->erase(seq->begin());
00069     return res;
00070   }
```

6.53.4.8 size()

```
size_t minerule::FSTreeSequence::size ( )
```

the number of items in this FSTreeSequence

Returns

the number of items in this FSTreeSequence

Definition at line 62 of file FSTreeSequence.cpp.

```
00062                                    {
00063     return seq->size();
00064   }
```

6.53.4.9 stampa()

```
void minerule::FSTreeSequence::stampa ( )
```

a simple method for print to standard output the items

Definition at line 55 of file FSTreeSequence.cpp.

```
00055                                {
00056     ItemVector::const_iterator it;
00057     for(it=seq->begin(); it!=seq->end(); ++it) {
00058       MRLog() « *it « std::endl;
00059     }
00060   }
```

6.53.4.10 svuota()

```
void minerule::FSTreeSequence::svuota ( )
```

delete all the items in this FSTreeSequence, after this operation the FSTreeSequence is empty

Definition at line 90 of file FSTreeSequence.cpp.

```
00090                               {
00091     delete seq;
00092     seq=new ItemVector();
00093   }
```

6.53.4.11 toStdString()

```
std::string minerule::FSTreeSequence::toStdString ( )
```

Returns

astd::string that contains the items in this FSTreeSequence

Definition at line 76 of file FSTreeSequence.cpp.

```
00076                                        {
00077     ItemVector::const_iterator it;
00078    std::string ris="";
00079     for (it=seq->begin();it!=seq->end();it++)
00080       ris = ris+" "+(*it).getSQLData();
00081
00082     /*   std::string ris="";
00083     if (seq->size()>0)
00084       ris=(*seq)[0];
00085     for (size_t i=1;i<seq->size();i++)
00086     ris=ris+"->"+(*seq)[i]; */
00087     return ris;
00088   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeSequence.hpp
- /Users/esposito/Software/minerule/src/Algorithms/FSTreeSequence.cpp

6.54 minerule::GAQueryCombinator Class Reference

`#include <GAQueryCombinator.hpp>`

Public Types

- typedef std::vector< size_t > QueryAndList
- typedef std::vector< QueryAndList > QueryOrList

Public Member Functions

- GAQueryCombinator (Predicate &_mr_target, std::vector< Predicate > &_mr_candidates)
- virtual ~GAQueryCombinator ()
- void evolve ()
- const QueryOrList & getResult () const
- float getTimeOutThreshold () const
- size_t getMaxDisjuncts () const
- size_t getMaxQueries () const
- size_t getMaxDistinctPredicates () const
- void setMaxDisjuncts (size_t m)
- void setMaxQueries (size_t m)
- void setMaxDistinctPredicates (size_t m)
- void setTimeOutThreshold (float to)

Static Public Member Functions

- static float evaluator (GAGenome &g)
- static GABoolean TerminationCriterion (GAGeneticAlgorithm &ga)

6.54.1 Detailed Description

This class implements the GA logic. It requires a vector V of predicates to be specified along with a target predicate Q. The predicates in the vector are and'ed and or'ed until a formula equivalent to predicate Q is found. The result R is returned as a vector of int vectors. R represents a disjunctive formula, i.e. R[0] represent the first conjunct, R[1] represent the second conjunct and so on. Thus R[0] or R[1] or ... or R[R.size()-1] represent the result of the algorithm.

Each int vector V represent a conjunction, element j being set to the index of the predicate that is used in the conjunction. The indexes refer to the predicates in mr_candidates (C in the following), and hence the resulting predicate can be written as: C(R[0][0]) and C(R[0][1]) ... and C(R[0][R[0].size()-1]) or C(R[1][0]) and C(R[1][1]) ... and C(R[1][R[1].size()-1]) or ... C(R[R.size()-1][0]) and C(R[R.size()-1][1]) ... and C(R[R.size()-1][R[R.size()-1].size()-1])

Definition at line 72 of file GAQueryCombinator.hpp.

6.54.2 Member Typedef Documentation

6.54.2.1 QueryAndList

```
typedef std::vector<size_t> minerule::GAQueryCombinator::QueryAndList
```

And of queries.

Definition at line 79 of file GAQueryCombinator.hpp.

6.54.2.2 QueryOrList

```
typedef std::vector<QueryAndList> minerule::GAQueryCombinator::QueryOrList
```

Or of queries. It is the type used to return the result to the user.

Definition at line 83 of file GAQueryCombinator.hpp.

6.54.3 Constructor & Destructor Documentation

6.54.3.1 GAQueryCombinator()

```
minerule::GAQueryCombinator::GAQueryCombinator (
            Predicate & _mr_target,
            std::vector< Predicate > & _mr_candidates )  [inline]
```

GAQueryCombinator constructor. Notice that a reference to _mr_target and one to _mr_candidates are stored into the class. Hence, the user must be sure to not delete those object before this class has been destroyed.

The constructor sets the following default values for the search parameters: timeOut=4 maxDisjuncts=3 max↩ Queries=5 maxDistinctPredicates=10

Definition at line 172 of file GAQueryCombinator.hpp.

```
00173       : mr_target(_mr_target),
00174         mr_candidates(_mr_candidates),
00175         timeOut(4.0),
00176         maxDisjuncts(3),
00177         maxQueries(5),
00178         maxDistinctPredicates(10) {}
```

6.54.3.2 ~GAQueryCombinator()

```
virtual minerule::GAQueryCombinator::~GAQueryCombinator ( )  [inline], [virtual]
```

Definition at line 180 of file GAQueryCombinator.hpp.

```
00180 {}
```

6.54.4 Member Function Documentation

6.54.4.1 evaluator()

```
float minerule::GAQueryCombinator::evaluator (
            GAGenome & g )  [static]
```

This is the fitness function.

Parameters

g	the GA1DBinaryStringGenome object representing the genome that need to be evaluated

Returns

the score of this genome (in the current implementation the best score is zero, being all the others negative numbers.

Definition at line 74 of file GAQueryCombinator.cpp.

```
00074                                                   {
00075                 GAQueryCombinator* qc = (GAQueryCombinator*) g.userData();
00076                 assert(qc!=NULL);
00077
00078                 if((clock()-qc->startingTime)/CLOCKS_PER_SEC > qc->timeOut )
00079                         throw TimeOutException();
00080
00081                 size_t numUsedDisjuncts=0;
00082                 size_t numUsedQueries=0;
00083                 size_t numAssertedBits=0;
00084
00085                 Predicate p2;
00086                 qc->buildAssociatedPredicate(g,p2, numUsedDisjuncts, numUsedQueries, numAssertedBits);
00087
00088                 if(numUsedDisjuncts > qc->maxDisjuncts ||       numUsedQueries > qc->maxQueries)
00089                                 return 0;
00090
00091                 Predicate& p1=qc->mr_target;
00092
00093     // The following two lines assign unique variables id
00094     // to elements in p1,p2. It do so setting the variable id
00095     // for each element of the union (through ci) to the position
00096     // in which the predicate appear in the union itself. Note
00097     // that this work only because two equivalent predicates are
00098     // indeed two references to the same object (look at SimplePredicate
00099     // class to understand why this is true).
00100                 size_t counter=0;
00101                 list_AND_node* allPreds=NULL;
00102
00103                 CountingIterator ci(counter,allPreds);
00104                 std::set<SimplePredicate*, PtrSimplePredComp>& sp1 = p1.getPredicateList();
00105                 std::set<SimplePredicate*, PtrSimplePredComp>& sp2 = p2.getPredicateList();
00106
00107                 std::set_union( sp1.begin(), sp1.end(),
00108                         sp2.begin(), sp2.end(), ci );
00109
00110
00111                 p1.setNumVariables(counter);
00112                 p2.setNumVariables(counter);
00113
00114                 InvalidConfigurationFilter filter( qc->tab_source, allPreds );
00115
00116                 ExpressionNFCoder e1(filter);
00117                 ExpressionNFCoder e2(filter);
00118
00119                 float ham_distance=EncodedNF::computeHammingDistance(e1.encode(p1),e2.encode(p2));
00120
00121                 CountingIterator::delete_list_AND_node(allPreds);
00122
00123                 if( ham_distance==0 )
00124                         return HUGE_VAL;
00125
00126                 return ( (1 << qc->maxDistinctPredicates)* HAM_LOSS
00127                         + qc->mr_candidates.size()*qc->maxDisjuncts - ham_distance*HAM_LOSS -
      numAssertedBits);
00128
00129         }
```

6.54.4.2 evolve()

```
void minerule::GAQueryCombinator::evolve ( )
```

The main algorithm. It starts the genetic algorithm and fills the result object.

Definition at line 142 of file GAQueryCombinator.cpp.

```
00142                                       {
00143     // create a genome
00144                 GA1DBinaryStringGenome genome(mr_candidates.size()*maxDisjuncts,
      GAQueryCombinator::evaluator, this);
00145     // create the genetic algorithm
00146                 GASimpleGA ga(genome);
00147                 ga.populationSize(10);
00148                 ga.nGenerations(100);
00149                 ga.pCrossover(0.8);
00150                 ga.pMutation(0.05);
00151                 ga.terminator(TerminationCriterion);
00152     // do the evolution
00153
00154                 startingTime = clock();
00155                 ga.evolve();
00156
00157                 buildResult(ga);
00158         }
```

6.54.4.3 getMaxDisjuncts()

```
size_t minerule::GAQueryCombinator::getMaxDisjuncts ( ) const  [inline]
```

Definition at line 214 of file GAQueryCombinator.hpp.

```
00214 {return maxDisjuncts;}
```

6.54.4.4 getMaxDistinctPredicates()

```
size_t minerule::GAQueryCombinator::getMaxDistinctPredicates ( ) const  [inline]
```

Definition at line 216 of file GAQueryCombinator.hpp.

```
00216 {return maxDistinctPredicates;}
```

6.54.4.5 getMaxQueries()

```
size_t minerule::GAQueryCombinator::getMaxQueries ( ) const  [inline]
```

Definition at line 215 of file GAQueryCombinator.hpp.

```
00215 {return maxQueries;}
```

6.54.4.6 getResult()

```
const GAQueryCombinator::QueryOrList & minerule::GAQueryCombinator::getResult ( ) const
```

Returns the result of the GA algorithm.

Definition at line 184 of file GAQueryCombinator.cpp.

```
00184                                                                   {
00185                 return result;
00186         }
```

6.54.4.7 getTimeOutThreshold()

```
float minerule::GAQueryCombinator::getTimeOutThreshold ( ) const  [inline]
```

Definition at line 213 of file GAQueryCombinator.hpp.

```
00213 { return timeOut; }
```

6.54.4.8 setMaxDisjuncts()

```
void minerule::GAQueryCombinator::setMaxDisjuncts (
            size_t m )  [inline]
```

Definition at line 218 of file GAQueryCombinator.hpp.

```
00218 { maxDisjuncts=m; }
```

6.54.4.9 setMaxDistinctPredicates()

```
void minerule::GAQueryCombinator::setMaxDistinctPredicates (
            size_t m )  [inline]
```

Definition at line 220 of file GAQueryCombinator.hpp.

```
00220 { maxDistinctPredicates=m; }
```

6.54.4.10 setMaxQueries()

```
void minerule::GAQueryCombinator::setMaxQueries (
            size_t m )  [inline]
```

Definition at line 219 of file GAQueryCombinator.hpp.

```
00219 { maxQueries=m; }
```

6.54.4.11 setTimeOutThreshold()

```
void minerule::GAQueryCombinator::setTimeOutThreshold (
            float to )  [inline]
```

Definition at line 221 of file GAQueryCombinator.hpp.

```
00221 { timeOut=to; }
```

6.54.4.12 TerminationCriterion()

```
GABoolean minerule::GAQueryCombinator::TerminationCriterion (
            GAGeneticAlgorithm & ga )  [static]
```

The termination criterion for the GA.

Definition at line 133 of file GAQueryCombinator.cpp.

```
00133                                                                   {
00134                 if(ga.statistics().current( GAStatistics::Maximum )==HUGE_VAL)
00135                         return gaTrue;
00136                 else
00137                         return (ga.generation() < ga.nGenerations() ? gaFalse : gaTrue);
00138         }
```

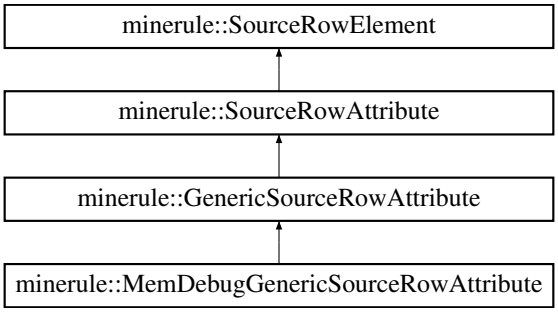
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/GAQueryCombinator.hpp
- /Users/esposito/Software/minerule/src/Optimizer/GAQueryCombinator.cpp

6.55 minerule::GenericSourceRowAttribute Class Reference

```
#include <SourceRowAttribute.hpp>
```

Inheritance diagram for minerule::GenericSourceRowAttribute:



Public Types

- typedef char ElementType

Public Member Functions

- GenericSourceRowAttribute (mrdb::ResultSet *_rs, int _elem, mrdb::Types::SQLType _type)
- GenericSourceRowAttribute ()
- GenericSourceRowAttribute (const GenericSourceRowAttribute &rhs)
- virtual ~GenericSourceRowAttribute ()
- virtual SourceRowElement * copy () const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const
- virtual mrdb::Types::SQLType getType () const
- virtual void setValue (mrdb::ResultSet *, int)
- virtual void setValue (const std::string &)
- virtual int compareTo (const SourceRowAttribute &) const

- virtual std::string asString (const std::string &sep=",") const
- virtual std::string getSQLData () const
- virtual SourceRowElement & operator= (const SourceRowElement &_rhs)
- virtual ElementType getElementType () const
- virtual void serialize (std::ostream &os) const
- virtual void deserialize (std::istream &is)
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &s1) const
- virtual bool empty () const

  return true if this attribute is empty
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual std::string getFullElementType () const

Static Public Member Functions

- static SourceRowAttribute * createAttribute (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, int elem)
- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

Friends

- std::ostream & operator<< (std::ostream &os, const SourceRowAttribute &ia)

6.55.1 Detailed Description

This is a generic attribute class, i.e. it can contain any attribute type. It should be used when more specific classes are not available.

Definition at line 141 of file SourceRowAttribute.hpp.

6.55.2 Member Typedef Documentation

6.55.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.55.3 Constructor & Destructor Documentation

6.55.3.1 GenericSourceRowAttribute() [1/3]

```
minerule::GenericSourceRowAttribute::GenericSourceRowAttribute (
            mrdb::ResultSet * _rs,
            int _elem,
            mrdb::Types::SQLType _type )  [inline]
```

Definition at line 149 of file SourceRowAttribute.hpp.

```
00151       : value(_rs->getString(_elem)), type(_type) {}
```

6.55.3.2 GenericSourceRowAttribute() [2/3]

```
minerule::GenericSourceRowAttribute::GenericSourceRowAttribute ( )  [inline]
```

Definition at line 153 of file SourceRowAttribute.hpp.

```
00154       : value(""), type((mrdb::Types::SQLType)0)
00155         /*, colId(-1) */ {};
```

6.55.3.3 GenericSourceRowAttribute() [3/3]

```
minerule::GenericSourceRowAttribute::GenericSourceRowAttribute (
            const GenericSourceRowAttribute & rhs )  [inline]
```

Definition at line 158 of file SourceRowAttribute.hpp.

```
00159       : value(rhs.value), type(rhs.type) /*, colId(rhs.colId)*/ {}
```

6.55.3.4 ~GenericSourceRowAttribute()

```
virtual minerule::GenericSourceRowAttribute::~GenericSourceRowAttribute ( )  [inline], [virtual]
```

Definition at line 161 of file SourceRowAttribute.hpp.

```
00161                                       {
00162     //      std::cout « "Generic... destructor" « std::endl;
00163   }
```

6.55.4 Member Function Documentation

6.55.4.1 asString()

```
std::string minerule::GenericSourceRowAttribute::asString (
            const std::string & sep = "," ) const  [virtual]
```

Convert this attribute to a string.

Parameters

sep	a separator string. Composite attributes should use this value to separate fields.

Returns

the std::string representation of the value of this attribute.

Implements minerule::SourceRowAttribute.

Definition at line 84 of file SourceRowAttribute.cpp.

```
00084                                                                   {
00085                 if(value == "")
00086                         return "(not set)";
00087                 else
00088                         return value;
00089         }
```

6.55.4.2 compareTo()

```
int minerule::GenericSourceRowAttribute::compareTo (
            const SourceRowAttribute &  ) const  [virtual]
```

Compares the present attribute value with the one of the given attribute.

Returns

-1 if the present attribute is less than the argument, 0 if they are equal +1 otherwise

Implements minerule::SourceRowAttribute.

Definition at line 74 of file SourceRowAttribute.cpp.

```
00074                                                                   {
00075                 return value.compare(dynamic_cast<const GenericSourceRowAttribute&>(rhs).value);
00076         }
```

6.55.4.3 copy()

```
virtual SourceRowElement * minerule::GenericSourceRowAttribute::copy ( ) const  [inline], [virtual]
```

Returns

a fresh allocated copy of the current object

Implements minerule::SourceRowAttribute.

Reimplemented in minerule::MemDebugGenericSourceRowAttribute.

Definition at line 167 of file SourceRowAttribute.hpp.

```
00167                                           {
00168     SourceRowAttribute *newRow = new GenericSourceRowAttribute(*this);
00169     return newRow;
00170   }
```

6.55.4.4 createAttribute()

```
SourceRowAttribute * minerule::SourceRowAttribute::createAttribute (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            int elem )  [static], [inherited]
```

Factory method.

Returns

an attribute capable of handling types of the given type

Factory method - returns an attribute capable of handling types of the given type

Definition at line 34 of file SourceRowAttribute.cpp.

```
00037                   {
00038                 mrdb::Types::SQLType colType = (mrdb::Types::SQLType) rsmd->getColumnType(elem);
00039 #ifdef DEBUG
00040 #warning MemDebugGeneric...
00041                 return new MemDebugGenericSourceRowAttribute(rs,elem,colType);
00042 #else
00043                 if(colType==mrdb::Types::INTEGER || colType==mrdb::Types::BIGINT)
00044                         return new NumericSourceRowAttribute(rs,elem);
00045                 else
00046                         return new GenericSourceRowAttribute(rs,elem,colType);
00047 #endif
00048         }
```

6.55.4.5 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.55.4.6 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.55.4.7 deserialize()

```
void minerule::GenericSourceRowAttribute::deserialize (
            std::istream & is )  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.cpp.

```
00097                                                                   {
00098     // In order to avoid missing spaces etc. we resort to use istream::get method
00099     // instead of » operator. This means that we need to read one char at a time
00100     // and implement a simple automata in order to parse the regular expression / @[ .* @] /
00101     // Pro: quite general
00102     // Cons: Nobody ensures that the strings @[ and @] does not appear in the value itself
00103     //       When this happens everything break
00104     // Solution: We should escape any '@' character in thestd::string before inserting it
00105     // into the database. (Cons: "The insertion process becomes more costly...")
00106
00107
00108                 typedef enum {
00109                         AS_NORMAL, // normal automa state
00110                                 AS_EXITING // The state in which the automa will be when ? has been
      read as the last char
00111                                         } AutomaState;
00112
00113                         std::string buf;
00114                         char ch;
00115
00116                         assert(is);
00117                         is » buf;
00118                         assert(is);
00119
00120                         if(buf!="@[")
00121                                 throw MineruleException(MR_ERROR_INTERNAL, "Expected `@[', but:
      `"+buf+"' found!");
00122
00123                         is.get(); // throwing away spurious space.
00124
00125                         bool finished = false;
00126                         value = "";
00127
00128                         AutomaState state = AS_NORMAL;
00129                         while(is.good() && !finished) {
00130                                 ch=is.get();
00131
00132                                 if( ch==']' && state==AS_EXITING )
00133                                         finished=true;
```

```
00134                                   else {
00135                                           switch( ch ) {
00136                                           case '@':
00137                                                   state = AS_EXITING;
00138                                                   break;
00139
00140                                           default:
00141                                                   if(state==AS_EXITING)
00142                                                           value += '@';
00143
00144                                                   value += ch;
00145                                                   break;
00146                                           }
00147                                   }
00148                           }
00149
00150                           if(!is)
00151                                   throw MineruleException(MR_ERROR_INTERNAL,
00152                                           "Unfinished Generic value, value read 'till now:"+value);
00153                   }
```

6.55.4.8 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                   {
00068                   mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                   size_t numColumns = rsmd->getColumnCount();
00070                   std::vector<int> descr;
00071
00072                   for(size_t i=start_index; i<=numColumns; ++i)
00073                           descr.push_back(i);
00074
00075                   SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                   return elem;
00077           }
```

6.55.4.9 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                                   {
00082                   std::istringstream sstream(strRepr);
00083                   assert(sstream);
00084                   ElementType elType;
00085                   sstream>>elType;
00086                   assert(sstream);
00087
00088                   SourceRowElement* elem = createElementFromType(elType);
00089                   elem->deserialize(sstream);
00090                   return elem;
00091           }
```

6.55.4.10 empty()

```
virtual bool minerule::SourceRowAttribute::empty ( ) const  [inline], [virtual], [inherited]
```

return true if this attribute is empty

Implements minerule::SourceRowElement.

Definition at line 113 of file SourceRowAttribute.hpp.

```
00113 { return false; }
```

6.55.4.11 getElementType()

```
virtual ElementType minerule::GenericSourceRowAttribute::getElementType ( ) const  [inline],
[virtual]
```

Serialization methods

Reimplemented from minerule::SourceRowElement.

Definition at line 199 of file SourceRowAttribute.hpp.

```
00199 { return 'g'; }
```

6.55.4.12 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                           {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.55.4.13 getSQLData()

```
std::string minerule::GenericSourceRowAttribute::getSQLData ( ) const  [virtual]
```

Returns

the std::string representation for the attribute in a format suitable to be used in SQL queries (tipically it is implemented as return "'"+this->asString()+"'"

Implements minerule::SourceRowAttribute.

Definition at line 156 of file SourceRowAttribute.cpp.

```
00156                                                   {
00157                         return std::string("'")+value+"'";
00158                 }
```

6.55.4.14 getType()

```
mrdb::Types::SQLType minerule::GenericSourceRowAttribute::getType ( ) const  [virtual]
```

Returns

the type of the attribute

Implements minerule::SourceRowAttribute.

Definition at line 79 of file SourceRowAttribute.cpp.

```
00079                                                {
00080                 return type;
00081         }
```

6.55.4.15 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                {
00054     return !this->operator==(el);
00055   }
```

6.55.4.16 operator()()

```
virtual bool minerule::SourceRowAttribute::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [inline], [virtual], [inherited]
```

Returns

true iff s1<s2

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.hpp.

```
00098                                                {
00099     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00100     const SourceRowAttribute &attr2 = dynamic_cast<const SourceRowAttribute &>(s2);
00101
00102     return attr1.compareTo(attr2) < 0;
00103   }
```

6.55.4.17 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                {
00058     return this->operator()(*this, el);
00059   }
```

6.55.4.18 operator<<()

```
virtual std::ostream & minerule::SourceRowAttribute::operator<< (
            std::ostream & os ) const  [inline], [virtual], [inherited]
```

Inserts the value of this attribute in the provided ostream and returns the ostream.

Parameters

os	an output stream

Returns

the output stream given in input

Implements minerule::SourceRowElement.

Definition at line 129 of file SourceRowAttribute.hpp.

```
00129                                                    {
00130     os « asString();
00131     return os;
00132   }
```

6.55.4.19 operator=()

```
virtual SourceRowElement & minerule::GenericSourceRowAttribute::operator= (
            const SourceRowElement & _rhs )  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 188 of file SourceRowAttribute.hpp.

```
00188                                                                {
00189     const GenericSourceRowAttribute &rhs =
00190         dynamic_cast<const GenericSourceRowAttribute &>(_rhs);
00191     value = rhs.value;
00192     type = rhs.type;
00193     return *this;
00194   }
```

6.55.4.20 operator==()

```
virtual bool minerule::SourceRowAttribute::operator== (
            const SourceRowElement & s1 ) const  [inline], [virtual], [inherited]
```

Returns

true iff ∗this == s1

Implements minerule::SourceRowElement.

Definition at line 106 of file SourceRowAttribute.hpp.

```
00106                                                    {
00107     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00108
00109     return this->compareTo(attr1) == 0;
00110   }
```

6.55.4.21 serialize()

```
void minerule::GenericSourceRowAttribute::serialize (
            std::ostream & os ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 92 of file SourceRowAttribute.cpp.

```
00092                                                                    {
00093                 os « " @[ " « value « "@] ";
00094         }
```

6.55.4.22 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
        {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.55.4.23 setPreparedStatementParameters()

```
virtual void minerule::GenericSourceRowAttribute::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 172 of file SourceRowAttribute.hpp.

```
00173                                                                    {
00174     state->setString(start_index, value);
00175   }
```

6.55.4.24 setValue() [1/2]

```
void minerule::GenericSourceRowAttribute::setValue (
            const std::string & value )  [virtual]
```

Sets the value of the attribute according to the given string (converting the value to the proper type when necessary).

Parameters

value	a string representation of the value to be set

Implements minerule::SourceRowAttribute.

Definition at line 66 of file SourceRowAttribute.cpp.

```
00066                                                                   {
00067
00068                 type = (mrdb::Types::SQLType) 4;
00069                 value = inStr;
00070         }
```

6.55.4.25 setValue() [2/2]

```
void minerule::GenericSourceRowAttribute::setValue (
            mrdb::ResultSet * rs,
            int col )  [virtual]
```

Sets the value of the attribute accordingly to the value of the specified column of the current row of the given result set

Parameters

rs	the result set source of the value
col	the index (starting from 1) of the column in the result set

Implements minerule::SourceRowAttribute.

Definition at line 57 of file SourceRowAttribute.cpp.

```
00057                                                                   {
00058                 assert(rs!=NULL);
00059                 assert(col<=rs->getMetaData()->getColumnCount());
00060
00061                 type = (mrdb::Types::SQLType) rs->getMetaData()->getColumnType(col);
00062                 value = rs->getString(col);
00063         }
```

6.55.5 Friends And Related Function Documentation

6.55.5.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const SourceRowAttribute & ia )  [friend]
```

Definition at line 313 of file SourceRowAttribute.hpp.

```
00314                                                                   {
00315   attr.operator«(os);
00316   return os;
00317 }
```

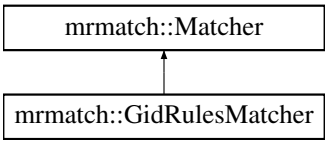
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp

6.56 mrmatch::GidRulesMatcher Class Reference

`#include <GidRulesMatcher.hpp>`

Inheritance diagram for mrmatch::GidRulesMatcher:



Public Member Functions

- GidRulesMatcher ()
- virtual ~GidRulesMatcher ()
- virtual minerule::Rule & addRule ()
- virtual void printMatches () const
- virtual void match (minerule::SourceTable &st)
- virtual void matchWithCrossProduct (minerule::SourceTable &st)
- virtual void matchWithoutCrossProduct (minerule::SourceTable &st)

Static Public Member Functions

- static Matcher * newMatcher (const Options &opts, const minerule::ParsedMinerule &p)

Protected Member Functions

- virtual void matchItemTransaction (const minerule::ItemType &gid, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &bodies, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &heads)
- virtual void matchRuleTransaction (const minerule::ItemType &gid, const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > &transaction)

6.56.1 Detailed Description

Definition at line 23 of file GidRulesMatcher.hpp.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 GidRulesMatcher()

`mrmatch::GidRulesMatcher::GidRulesMatcher ( ) [inline]`

Definition at line 39 of file GidRulesMatcher.hpp.

```
00039 {};
```

6.56.2.2 ~GidRulesMatcher()

```
virtual mrmatch::GidRulesMatcher::~GidRulesMatcher ( )  [inline], [virtual]
```

Definition at line 40 of file GidRulesMatcher.hpp.

```
00040 {};
```

6.56.3 Member Function Documentation

6.56.3.1 addRule()

```
Rule & mrmatch::GidRulesMatcher::addRule ( )  [virtual]
```

Implements mrmatch::Matcher.

Definition at line 23 of file GidRulesMatcher.cpp.

```
00023                                       {
00024                 rules.push_back(Rule());
00025                 return rules.back();
00026         }
```

6.56.3.2 match()

```
virtual void mrmatch::Matcher::match (
            minerule::SourceTable & st )  [inline], [virtual], [inherited]
```

Definition at line 39 of file Matcher.hpp.

```
00039                                                       {
00040                         st.usesCrossProduct() ? matchWithCrossProduct(st) :
      matchWithoutCrossProduct(st);
00041                 }
```

6.56.3.3 matchItemTransaction()

```
void mrmatch::GidRulesMatcher::matchItemTransaction (
            const minerule::ItemType & gid,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & bodies,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & heads )
[protected], [virtual]
```

Implements mrmatch::Matcher.

Definition at line 28 of file GidRulesMatcher.cpp.

```
00028                                                                                   {
00029                 _gidRulesVector.push_back( GidRules() );
00030                 GidRules& gidRules = _gidRulesVector.back();
00031                 gidRules.first = gid;
00032
00033                 for(RuleVector::iterator ruleIt = rules.begin(); ruleIt!=rules.end(); ++ruleIt) {

00034                         if( RulesMatcher::match( *ruleIt, bodies, heads ) ) {
00035                                 gidRules.second.push_back(& (*ruleIt) );
00036                         }
00037                 }
00038         }
```

6.56.3.4 matchRuleTransaction()

```
void mrmatch::GidRulesMatcher::matchRuleTransaction (
            const minerule::ItemType & gid,
            const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > & transaction
)  [protected], [virtual]
```

Implements mrmatch::Matcher.

Definition at line 40 of file GidRulesMatcher.cpp.

```
00040                                       {
00041                 _gidRulesVector.push_back( GidRules() );
00042                 GidRules& gidRules = _gidRulesVector.back();
00043                 gidRules.first = gid;
00044 
00045                 for(RuleVector::iterator ruleIt = rules.begin(); ruleIt!=rules.end(); ++ruleIt) {

00046                         if( RulesMatcher::match( *ruleIt, transaction ) ) {
00047                                 gidRules.second.push_back(& (*ruleIt) );
00048                         }
00049                 }
00050         }
```

6.56.3.5 matchWithCrossProduct()

```
void mrmatch::Matcher::matchWithCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 38 of file Matcher.cpp.

```
00038                                                       {
00039                 SourceTable::Iterator it = st.newIterator(SourceTable::FullIterator);
00040 
00041                 while(!it.isAfterLast()) {
00042                         ItemType gid = it->getGroup();
00043 
00044                         RuleTransaction<RulesMatcher::RuleSetType> transaction;
00045                         transaction.load(gid, it);
00046 
00047                         matchRuleTransaction(gid, transaction);
00048                 }
00049         }
```

6.56.3.6 matchWithoutCrossProduct()

```
void mrmatch::Matcher::matchWithoutCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 51 of file Matcher.cpp.

```
00051                                                          {
00052                 SourceTable::Iterator bodyIt = st.newIterator(SourceTable::BodyIterator);
00053                 SourceTable::Iterator headIt = st.newIterator(SourceTable::HeadIterator);
00054 
00055                 while(!bodyIt.isAfterLast()) {
00056                         ItemType gid = bodyIt->getGroup();
00057 
00058                         ItemTransaction<RulesMatcher::ItemSetType> bodies;
00059                         ItemTransaction<RulesMatcher::ItemSetType> heads;
00060 
00061                         bodies.loadBody(gid, bodyIt);                   // this advances the body
      iterator
00062 
00063                         if( !TransactionBase<RulesMatcher::ItemSetType>::findGid(gid, headIt) ) {//
      positioning the head iterator
```

```
00064                                   break;                                                                  // no
      more heads to load
00065                           }
00066
00067                           heads.loadHead(gid, headIt);                    // loading the heads
00068
00069                           matchItemTransaction(gid, bodies, heads);
00070                   } // while
00071
00072           } // matchWithoutCrossProduct
```

6.56.3.7 newMatcher()

```
Matcher * mrmatch::Matcher::newMatcher (
            const Options & opts,
            const minerule::ParsedMinerule & p )  [static], [inherited]
```

Definition at line 25 of file Matcher.cpp.

```
00025                                                                                                           {
00026                   MatcherSpecs specs = opts.matcherSpecs();
00027                   if((specs & MatchOutputMask) == OutOnDB) {
00028                           return new RuleGidsDBMatcher(opts.getMatchOutputTableName(), minerule);
00029                   }
00030
00031                   switch(specs & MatchKindMask) {
00032                           case RuleGids: return new RuleGidsMatcher();
00033                           case GidRules: return new GidRulesMatcher();
00034                           default: throw MineruleException( MR_ERROR_INTERNAL, "Unknown or unsupported
      matcher kind");
00035                   }
00036           }
```

6.56.3.8 printMatches()

```
void mrmatch::GidRulesMatcher::printMatches ( ) const  [virtual]
```

Implements mrmatch::Matcher.

Definition at line 52 of file GidRulesMatcher.cpp.

```
00052                                                   {
00053                   SimpleRuleFormatter sf;
00054                   sf.setFieldWidths( SimpleRuleFormatter::FieldWidths(1,1,1,1) );
00055
00056                   for( GidRulesVector::const_iterator it= _gidRulesVector.begin();
      it!=_gidRulesVector.end(); ++it ) {
00057                                   MRLog() « StringUtils::toBold("Gid: ") « it->first « std::endl;
00058                                   const RulePtrVector& matchedRules = it->second;
00059
00060                                   for( RulePtrVector::const_iterator rIt=matchedRules.begin();
      rIt!=matchedRules.end(); ++rIt) {
00061                                           MRLog() « StringUtils::toBold("\tRule: ") « sf.formatRule(
      **rIt ) « std::endl;
00062                                   }
00063                   }
00064           }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.hpp
- /Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.cpp

6.57 minerule::Head Class Reference

`#include <IncrAlgoClasses.hpp>`

Public Types

- typedef std::map< ItemType, NodeRow * > RowContainer

Public Member Functions

- Head ()
- ~Head ()
- void setAncestor (ItemType h)
- ItemType getAncestor ()
- void insertItemSetH (ItemSet &SREV, double supp)
- void insertItemSetH (ItemSet::iterator ite, ItemSet::iterator itend, double s)
- void extractRules (const std::vector< ItemType > &body, std::vector< ItemType > &head, double thrR, double thrB, double suppB, int ngroups, Connection *pconnection)
- void findHead (ItemSet *head)
- void findHead2bis (ItemSet *h1nh2, ItemSet *h1)
- void findHead2 (ItemSet *h1)

Data Fields

- RowContainer * NR

Static Public Attributes

- static size_t counth =0

6.57.1 Detailed Description

Definition at line 68 of file IncrAlgoClasses.hpp.

6.57.2 Member Typedef Documentation

6.57.2.1 RowContainer

`typedef std::map<ItemType,NodeRow*> minerule::Head::RowContainer`

Definition at line 70 of file IncrAlgoClasses.hpp.

6.57.3 Constructor & Destructor Documentation

6.57.3.1 Head()

```
minerule::Head::Head ( )
```

Definition at line 49 of file IncrAlgoClasses.cpp.

```
00049           {
00050   NR=new RowContainer;
00051   counth++;
00052  }
```

6.57.3.2 ~Head()

```
minerule::Head::~Head ( )
```

Definition at line 54 of file IncrAlgoClasses.cpp.

```
00054            {
00055     counth--;
00056     RowContainer::iterator it;
00057     it=NR->begin();
00058     while(it!=NR->end()){
00059       delete (it->second);
00060       it++;
00061     }
00062     delete NR;
00063   }
```

6.57.4 Member Function Documentation

6.57.4.1 extractRules()

```
void minerule::Head::extractRules (
            const std::vector< ItemType > & body,
            std::vector< ItemType > & head,
            double thrR,
            double thrB,
            double suppB,
            int ngroups,
            Connection * pconnection )
```

Definition at line 390 of file IncrAlgoClasses.cpp.

```
00393                                                {
00394  typedef std::map<ItemType,NodeRow*> RowContainer;
00395
00396  RowContainer::iterator it;
00397  Head* htmp;
00398  double s;
00399  double tmp;
00400  it=this->NR->begin();
00401  while (it!=this->NR->end()){
00402    htmp=it->second->getChild();
00403    //se ha figli richiamo la funzione
```

```
00404    if (htmp!=NULL){
00405       head.push_back(it->first);
00406       htmp->extractRules(body,head,thrR,thrB,suppB,ngroups,pconnection);
00407       s=it->second->getSupp();
00408       double sup=s/ngroups;
00409       tmp=(s/suppB);
00410       if((sup>=thrR)/*&&(tmp>=thrB)*/){
00411         /*        std::cout«"body: "«std::endl;
00412         std::vector<ItemType>::const_iterator itv=body.begin();
00413         while (itv!=body.end()){
00414           std::cout«" "«(*itv)«std::endl;
00415           itv++;
00416           }
00417         std::cout«"suppb: "«tmp«std::endl;
00418         std::cout«"       head: "«std::endl;
00419         itv=head.begin();
00420         while (itv!=head.end()){
00421           std::cout«"               "«(*itv)«std::endl;
00422           itv++;
00423           }
00424         std::cout«"suppr: "«sup«std::endl;
00425         */
00426         pconnection->insert(body,head,sup,tmp);
00427       }
00428    } else { //se invece non ne ha stampo la regola se conf e supp sono sufficienti
00429      head.push_back(it->first);
00430      s=it->second->getSupp();
00431      double sup=s/ngroups;
00432      tmp=(s/suppB);
00433      if((sup>=thrR)/*&&(tmp>thrB)*/){
00434        // std::cout«"body: "«std::endl;
00435        // std::vector<ItemType>::const_iterator itv=body.begin();
00436        //while (itv!=body.end()){
00437         // std::cout«" "«(*itv)«std::endl;
00438         // itv++;
00439       // }
00440       // std::cout«"suppb: "«tmp«std::endl;
00441       // std::cout«"       head: "«std::endl;
00442       // itv=head.begin();
00443      //  while (itv!=head.end()){
00444         // std::cout«"               "«(*itv)«std::endl;
00445         // itv++;
00446       // }
00447       // std::cout«"suppr: "«sup«std::endl;
00448        pconnection->insert(body,head,sup,tmp);
00449      }
00450    }
00451    head.pop_back();
00452    it++;
00453  }
00454
00455 }
```

6.57.4.2 findHead()

```
void minerule::Head::findHead (
              ItemSet * head )
```

Definition at line 198 of file IncrAlgoClasses.cpp.

```
00198                                  {
00199   RowContainer::iterator ite=NR->begin();
00200   ItemSet::iterator itfind;
00201   //  ItemSet::iterator prova=head->begin();
00202   /*  std::cout«"elementi in cui faccio ricerca:"«std::endl;
00203   while(prova!=head->end()){
00204   std::cout«(*prova)«", ";
00205   prova++;
00206   }
00207   std::cout«std::endl;*/
00208
00209   while(ite!=NR->end()){
00210     //    std::cout«"ora cerco "«ite->first«std::endl;
00211     itfind=find(head->begin(),head->end(),ite->first);
00212     if(itfind!=head->end()) {//se l'ho trovato
00213       ite->second->incremSupp();
00214       //cout«"ho increm supp di rule"«std::endl;
00215       Head* c=ite->second->getChild();
00216       //se ha figli
```

```
00217       if (c!=NULL) {
00218         c->findHead(head);
00219       }
00220     }//else std::cout«"non trovato"«std::endl;
00221   ite++;
00222   }
00223  }
```

6.57.4.3 findHead2()

```
void minerule::Head::findHead2 (
             ItemSet * h1 )
```

Definition at line 228 of file IncrAlgoClasses.cpp.

```
00228                                {
00229   RowContainer::iterator ite=NR->begin();
00230   ItemSet::iterator itfind;
00231   while(ite!=NR->end()){
00232     //cout«"esamino head "«ite->first«std::endl;
00233     itfind=find(h1->begin(),h1->end(),ite->first);
00234     if(itfind!=h1->end()) {//se l'ho trovato
00235       ite->second->decremSupp();
00236       //cout«"decremento head "«ite->first«std::endl;
00237       Head* c=ite->second->getChild();
00238       //se ha figli
00239       if (c!=NULL) {
00240         c->findHead2(h1);
00241       }
00242     }//else std::cout«"non trovato"«std::endl;
00243   ite++;
00244   }
00245  }
```

6.57.4.4 findHead2bis()

```
void minerule::Head::findHead2bis (
             ItemSet * h1nh2,
             ItemSet * h1 )
```

Definition at line 249 of file IncrAlgoClasses.cpp.

```
00249                                                {
00250   RowContainer::iterator ite=NR->begin();
00251   ItemSet::iterator itfind;
00252   while(ite!=NR->end()){
00253     //cout«"esamino head "«ite->first«std::endl;
00254     itfind=find(h1nh2->begin(),h1nh2->end(),ite->first);
00255     if(itfind!=h1nh2->end()) {//se l'ho trovato
00256       ite->second->decremSupp();
00257       //cout«"decremento head "«ite->first«std::endl;
00258       Head* c=ite->second->getChild();
00259       //se ha figli
00260       if (c!=NULL) {
00261         c->findHead2(h1);
00262       }
00263     }else {
00264       itfind=find(h1->begin(),h1->end(),ite->first);
00265       if(itfind!=h1->end()) {//se l'ho trovato
00266        Head* c=ite->second->getChild();
00267        //se ha figli
00268        if (c!=NULL) {
00269          c->findHead2bis(h1nh2,h1);
00270        }
00271       }
00272      }
00273   ite++;
00274   }
00275  }
```

6.57.4.5 getAncestor()

```
ItemType minerule::Head::getAncestor ( )  [inline]
```

Definition at line 82 of file IncrAlgoClasses.hpp.

```
00082 {return ancestor;}
```

6.57.4.6 insertItemSetH() [1/2]

```
void minerule::Head::insertItemSetH (
            ItemSet & SREV,
            double supp )
```

Definition at line 169 of file IncrAlgoClasses.cpp.

```
00169                                                   {
00170   ItemSet:: iterator ite=SREV.begin();
00171   ItemSet:: iterator itend=SREV.end();
00172   insertItemSetH(ite,itend,supp);
00173  }
```

6.57.4.7 insertItemSetH() [2/2]

```
void minerule::Head::insertItemSetH (
            ItemSet::iterator ite,
            ItemSet::iterator itend,
            double s )
```

Definition at line 129 of file IncrAlgoClasses.cpp.

```
00130                                                                {
00131   RowContainer::iterator itr;
00132   itr=NR->find(*ite);
00133
00134   //se non c'e' lo inserisco
00135   if(itr==NR->end()){
00136    (*NR)[*ite]=new NodeRow();
00137    itr=NR->find(*ite);
00138    itr->second->setSupp(supp);
00139    //cout«"                head "«itr->first«std::endl;
00140    //se ci sono ancora elementi nel vettore, vuol dire che devo inserire i figli
00141    ite++;
00142    if(ite!=itend){
00143      Head* tmph=itr->second-> setChild();
00144      tmph->setAncestor(itr->first);
00145      tmph->insertItemSetH(ite,itend,supp);
00146      }
00147   }
00148
00149   //se l'ho trovato
00150   else{
00151    //cout«"               head "«itr->first«std::endl;
00152    //se ci sono ancora figli da inserire
00153    ite++;
00154    if(ite!=itend){
00155     Head* child=itr->second->getChild();
00156      //se ha gia' figli
00157     if(child!=NULL)
00158      child->insertItemSetH(ite,itend,supp);
00159     //se non ha figli
00160     else{
00161      Head* child=itr->second->setChild();
00162      child->insertItemSetH(ite,itend,supp);
00163      }
00164    }else itr->second->setSupp(supp);
00165   }
00166   //itr->second->setSupp(supp);
00167 }
```

6.57.4.8 setAncestor()

```
void minerule::Head::setAncestor (
            ItemType h )  [inline]
```

Definition at line 81 of file IncrAlgoClasses.hpp.
```
00081 {ancestor=h;}
```

6.57.5 Field Documentation

6.57.5.1 counth

```
size_t minerule::Head::counth =0  [static]
```

Definition at line 76 of file IncrAlgoClasses.hpp.

6.57.5.2 NR

```
RowContainer* minerule::Head::NR
```

Definition at line 80 of file IncrAlgoClasses.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp
- /Users/esposito/Software/minerule/src/Algorithms/IncrAlgoClasses.cpp

6.58 minerule::HeadBodyPredicatesSeparator Class Reference

```
#include <HeadBodyPredicatesSeparator.hpp>
```

Static Public Member Functions

- static void separate (list_AND_node *l_and, std::string &bodyConstraints, std::string &headConstraints)

6.58.1 Detailed Description

This class exports the separate static function which takes a list_AND_node and returns the BODY part and the HEAD part of the predicates as std::strings. It throws a MineruleException if any cross predicate is found. Notice: at the present time the HEAD and BODY part of each attribute name is removed before inserted in the result.

Definition at line 37 of file HeadBodyPredicatesSeparator.hpp.

6.58.2 Member Function Documentation

6.58.2.1 separate()

```
void minerule::HeadBodyPredicatesSeparator::separate (
            list_AND_node * l_and,
            std::string & bodyConstraints,
            std::string & headConstraints )  [static]
```

Definition at line 59 of file HeadBodyPredicatesSeparator.cpp.

```
00061                                   {
00062                 if( l_and == NULL ) {
00063                         bodyConstraints = "1=1";
00064                         headConstraints = "1=1";
00065                         return;
00066                 }
00067
00068                 while(l_and!=NULL) {
00069                         ConstraintClass state = NO_INFO;
00070                         std::string v1 = l_and->sp->val1;
00071                         std::string v2 = l_and->sp->val2;
00072
00073                         state = setStateAccordinglyToString(v1, state);
00074                         state = setStateAccordinglyToString(v2, state);
00075
00076                         switch(state) {
00077                                 case BODY_CONSTR:
00078                                         updateConstraint(bodyConstraints, v1, l_and->sp->op, v2);
00079                                         break;
00080                                 case HEAD_CONSTR:
00081                                         updateConstraint(headConstraints, v1, l_and->sp->op, v2);
00082                                         break;
00083                                 default:
00084                                         throw MineruleException(MR_ERROR_INTERNAL,
00085                                                 "The optimizer found a constraint defined neither on
       BODY nor on HEAD!");
00086                         };
00087
00088                         l_and=l_and->next;
00089                 }
00090         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/HeadBodyPredicatesSeparator.cpp

6.59 minerule::HeaderQuery Class Reference

`#include <HeaderQuery.hpp>`

Public Member Functions

- HeaderQuery (std::string &na, MinMaxPair *cB, MinMaxPair *cH, ParsedMinerule::AttrVector *aB, ParsedMinerule::AttrVector *aH)
- HeaderQuery (std::string &na, MinMaxPair *cB, ParsedMinerule::AttrVector *aB)
- std::string getName ()
- MinMaxPair * getCardBody ()
- MinMaxPair * getCardHead ()
- ParsedMinerule::AttrVector * getAttrBody ()
- ParsedMinerule::AttrVector * getAttrHead ()

6.59.1 Detailed Description

This class is used by the parser to encapsulate the header data in order to distinguish between itemsets and rules

Definition at line 12 of file HeaderQuery.hpp.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 HeaderQuery() [1/2]

```
minerule::HeaderQuery::HeaderQuery (
            std::string & na,
            MinMaxPair * cB,
            MinMaxPair * cH,
            ParsedMinerule::AttrVector * aB,
            ParsedMinerule::AttrVector * aH )  [inline]
```

- this constructor build an object for a mine rule query

Definition at line 26 of file HeaderQuery.hpp.

```
00030                                                :
00031                     name(na),
00032                     cardB(cB),
00033                     cardH(cH),
00034                     attrB(aB),
00035                     attrH (aH) {};
```

6.59.2.2 HeaderQuery() [2/2]

```
minerule::HeaderQuery::HeaderQuery (
            std::string & na,
            MinMaxPair * cB,
            ParsedMinerule::AttrVector * aB )  [inline]
```

- this constructor build an object for a mine itemset query

Definition at line 40 of file HeaderQuery.hpp.

```
00042                                                :
00043       name(na),
00044       cardB(cB),
00045       cardH(NULL),
00046       attrB(aB),
00047       attrH (NULL) {};
```

6.59.3 Member Function Documentation

6.59.3.1 getAttrBody()

```
ParsedMinerule::AttrVector * minerule::HeaderQuery::getAttrBody ( )  [inline]
```

Definition at line 66 of file HeaderQuery.hpp.

```
00066                                       {
00067       return attrB;
00068     }
```

6.59.3.2 getAttrHead()

```
ParsedMinerule::AttrVector * minerule::HeaderQuery::getAttrHead ( )  [inline]
```

Definition at line 70 of file HeaderQuery.hpp.

```
00070                                       {
00071       return attrH;
00072     }
```

6.59.3.3 getCardBody()

```
MinMaxPair * minerule::HeaderQuery::getCardBody ( )  [inline]
```

Definition at line 58 of file HeaderQuery.hpp.

```
00058                                {
00059       return cardB;
00060     }
```

6.59.3.4 getCardHead()

```
MinMaxPair * minerule::HeaderQuery::getCardHead ( )  [inline]
```

Definition at line 62 of file HeaderQuery.hpp.

```
00062                                {
00063       return cardH;
00064     }
```

6.59.3.5 getName()

```
std::string minerule::HeaderQuery::getName ( )  [inline]
```

- the following methods are used for take the value of this object

Definition at line 54 of file HeaderQuery.hpp.

```
00054                          {
00055       return name;
00056     }
```

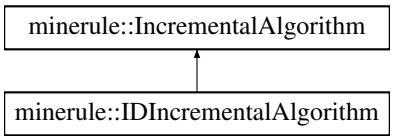
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Parsers/HeaderQuery.hpp

6.60 minerule::IDIncrementalAlgorithm Class Reference

`#include <IDIncrementalAlgorithm.hpp>`

Inheritance diagram for minerule::IDIncrementalAlgorithm:



Public Member Functions

- IDIncrementalAlgorithm (const OptimizedMinerule &mr)
- virtual ~IDIncrementalAlgorithm ()
- virtual void execute ()

Static Public Member Functions

- static IncrementalAlgorithm * newIncrementalAlgorithm (const OptimizedMinerule &mr)

Protected Member Functions

- std::set< ItemType > * fillValidItems (const std::string &constraints) const
- void getItemInfos (std::string &itemDescr, SourceRowColumnIds &hbsr) const
- bool checkInclusion (const std::set< ItemType > &validOnes, const ItemSet &foundOnes) const
- bool checkInValidRules (const ValidRules &validRules, Rule &r) const
- void filterQueries (const ValidRules &validRules)

Protected Attributes

- const ParsedMinerule::AttrVector * attrList
- const OptimizedMinerule * minerule

6.60.1 Detailed Description

This class implements a very simple incremental algorithm which can be used every time the predicates in the new query are ALL item dependent ones. In such a situation, we can easily retrieve the new result, by filtering out those itemset that does not satisfy the new constraints from the past result. In particular, we can retrieve the list of all items that do satisfy the constraint and use this list to prune the old result set.

Definition at line 39 of file IDIncrementalAlgorithm.hpp.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 IDIncrementalAlgorithm()

```
minerule::IDIncrementalAlgorithm::IDIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 54 of file IDIncrementalAlgorithm.hpp.
```
00054 : IncrementalAlgorithm(mr), attrList(NULL) { }
```

6.60.2.2 ~IDIncrementalAlgorithm()

```
virtual minerule::IDIncrementalAlgorithm::~IDIncrementalAlgorithm ( )  [inline], [virtual]
```

Definition at line 55 of file IDIncrementalAlgorithm.hpp.
```
00055 {}
```

6.60.3 Member Function Documentation

6.60.3.1 checkInclusion()

```
bool minerule::IDIncrementalAlgorithm::checkInclusion (
            const std::set< ItemType > & validOnes,
            const ItemSet & foundOnes ) const  [protected]
```

Notice that we implement our checkInclusion mechanism instead of using STL ``includes'' function. There is a performance reason to do that: the STL includes function, does not make assumption on the internal representation of the two sets, i.e. it does not exploit the fact that one can search one element in one of the sets in O(log(N)) time, moreover we also know that one of the two sets is *much* smaller than the other one. The total complexity of the STL algorithm is O(N+M) where N and M are the sizes of the two sets. This version is O(N∗log(M)), however since M is usually much bigger than N this strategy should pay in our case (e.g., a tipical N size may something as 3, while M is usually in the order of 1000, the linear algorithm takes about 1000 operations, while the one implemented here takes 3∗log(1000) ∼= 30 operations.

Definition at line 97 of file IDIncrementalAlgorithm.cpp.
```
00097
          {
00098                 ItemSet::const_iterator it;
00099                 for(it=foundOnes.begin();it!=foundOnes.end();it++) {
00100                         if(validOnes.find(*it)==validOnes.end())
00101                                 return false;
00102                 }
00103
00104                 return true;
00105         }
```

6.60.3.2 checkInValidRules()

```
bool minerule::IDIncrementalAlgorithm::checkInValidRules (
            const ValidRules & validRules,
            Rule & r ) const  [protected]
```

Definition at line 108 of file IDIncrementalAlgorithm.cpp.

```
00108                                                                                                   {
00109                 ValidRules::const_iterator it = validRules.begin();
00110                 bool found = false;
00111                 while(it!=validRules.end() && !found) {
00112                         found=
00113                                 (it->first==NULL || checkInclusion(*it->first, r.getBody())) &&
00114                                         (it->second==NULL|| checkInclusion(*it->second, r.getHead()));
00115
00116                         it++;
00117                 }
00118
00119                 return found;
00120         }
```

6.60.3.3 execute()

```
void minerule::IDIncrementalAlgorithm::execute ( )  [virtual]
```

Implements minerule::IncrementalAlgorithm.

Definition at line 157 of file IDIncrementalAlgorithm.cpp.

```
00157                                           {
00158                 assert(minerule->getParsedMinerule().mc!=NULL);
00159
00160                 MRLogPush("This is IDIncrementalAlgorithm. Starting...");
00161
00162                 ValidRules validRules;
00163                 list_OR_node* it;
00164                 MRLogPush("Building the list of items satisfying the constraints");
00165                 size_t conjNum=0;
00166                 for(it=minerule->getParsedMinerule().mc; it!=NULL; it=it->next) {
00167                         MRLog() « "Conjunct number:" « ++conjNum « std::endl;
00168                         std::string bodyConstraints;
00169                         std::string headConstraints;
00170                         HeadBodyPredicatesSeparator::separate(it->l_and, bodyConstraints,
      headConstraints);
00171
00172     // a little bit ugly, but useful. The fillValidItems will use the attrList
00173     // in order to select the correct portion of the database needed to build
00174     // the items in the head/body part of the rule. We need to set it to the
00175     // correct list before calling fillValidItems
00176                         attrList = &minerule->getParsedMinerule().ba;
00177                         std::set<ItemType>* validBodies = fillValidItems( bodyConstraints );
00178                         attrList = &minerule->getParsedMinerule().ha;
00179                         std::set<ItemType>* validHeads  = fillValidItems( headConstraints );
00180                         attrList = NULL;
00181                         validRules.push_back(ValidRule(validBodies,validHeads));
00182                 }
00183
00184                 MRLogPop(); // Building the list of items satisfying the constraints
00185
00186                 MRLogPush("Filtering past results...");
00187                 filterQueries(validRules);
00188                 MRLogPop(); // Filtering past results...
00189
00190   //trashing the trashable
00191                 MRLogPush("Cleaning up...");
00192                 ValidRules::const_iterator rule_it;
00193                 for(rule_it=validRules.begin();rule_it!=validRules.end();rule_it++) {
00194                         if(rule_it->first!=NULL)
00195                                 delete rule_it->first;
00196
00197                         if(rule_it->second!=NULL)
00198                                 delete rule_it->second;
00199                 }
00200                 MRLogPop(); // Cleaning up...
00201
00202
00203                 MRLogPop();  // This is IDIncrementalAlgorithm. Starting...
00204         }
```

6.60.3.4 fillValidItems()

```
std::set< ItemType > * minerule::IDIncrementalAlgorithm::fillValidItems (
            const std::string & constraints ) const  [protected]
```

Definition at line 44 of file IDIncrementalAlgorithm.cpp.

```
00044                                                                                {
00045                 if(constraints=="")
00046                         return NULL;
00047
00048                 std::set<ItemType>* result = new std::set<ItemType>;
00049
00050 #ifdef MRUSERWARNING
00051 #warning In the future we will assume to have a normalized database. \
00052                 Then we will need to make the following selection over the correct \
00053                 relation of the db instead of aover the current source table (which \
00054                 is needingly larger).
00055 #endif
00056                 std::string itemDescr;
00057                 SourceRowColumnIds hbsr;
00058
00059                 getItemInfos(itemDescr,hbsr);
00060
00061                 std::string query =
00062                         "SELECT DISTINCT "+itemDescr+" "+
00063                         "FROM "+minerule->getParsedMinerule().tab_source+" "+
00064                         (constraints.size()>0 ? "WHERE "+constraints : "");
00065
00066                 mrdb::Connection* con =
00067                         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00068
00069                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00070                 std::auto_ptr<mrdb::ResultSet> rs(state->executeQuery(query.c_str()));
00071
00072                 while(rs->next()) {
00073                         SourceRow sr(rs.get(), hbsr);
00074                         result->insert(sr.getBody());
00075                 };
00076
00077                 return result;
00078         }
```

6.60.3.5 filterQueries()

```
void minerule::IDIncrementalAlgorithm::filterQueries (
            const ValidRules & validRules )  [protected]
```

Definition at line 123 of file IDIncrementalAlgorithm.cpp.

```
00123                                                                                {
00124                 OptimizerCatalogue& cat =
      MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00125
00126                 std::string resName =
      cat.getResultsetName(minerule->getOptimizationInfo().minerule.tab_result );
00127
00128                 QueryResult::Iterator qri;
00129                 qri.init(resName, minerule->getOptimizationInfo().minerule.sup,
      minerule->getOptimizationInfo().minerule.conf);
00130
00131                 MRLog() << "tab results:" << resName << std::endl;
00132                 MRLog() << "Orig supp:"   << minerule->getOptimizationInfo().minerule.sup << std::endl;
00133                 MRLog() << "Orig conf:"   << minerule->getOptimizationInfo().minerule.conf << std::endl;
00134
00135                 MRLog() << "Connection ..."<<std::endl;
00136                 Connection connection;
00137                 connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00138                 connection.setBodyCardinalities(minerule->getParsedMinerule().bodyCardinalities);
00139                 connection.setHeadCardinalities(minerule->getParsedMinerule().headCardinalities);
00140                 connection.useMRDBConnection(
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00141                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule->getParsedMinerule()));
00142
00143                 size_t rcount = 0;
00144                 while(qri.next()) {
```

```
00145                           Rule r;
00146                           qri.getRule(r);
00147                           if(checkInValidRules(validRules, r))
00148                                   connection.insert(r.getBody(), r.getHead(), r.getSupport(),
      r.getConfidence());
00149
00150                           rcount++;
00151                   }
00152                   MRLog() « "Processed #" « rcount « " rules."«std::endl;
00153           }
```

6.60.3.6 getItemInfos()

```
void minerule::IDIncrementalAlgorithm::getItemInfos (
            std::string & itemDescr,
            SourceRowColumnIds & hbsr ) const  [protected]
```

Definition at line 29 of file IDIncrementalAlgorithm.cpp.

```
00029                                                                                                   {
00030                   assert(attrList!=NULL);
00031                   ParsedMinerule::AttrVector::const_iterator it;
00032                   int n;
00033                   for(it=attrList->begin(), n=1; it!=attrList->end(); it++,n++) {
00034                           if(itemDescr!="")
00035                                   itemDescr+=",";
00036
00037                           itemDescr+=*it;
00038                           hbsr.bodyElems.push_back(n);
00039                   }
00040           }
```

6.60.3.7 newIncrementalAlgorithm()

```
IncrementalAlgorithm * minerule::IncrementalAlgorithm::newIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.cpp.

```
00025                                                                                                   {
00026     // if mr has only ItemDependent constraints
00027                   MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028
00029                   if( mr.hasIDConstraints() ) {
00030                           MRLog() « "The minerule is item dependent!" « std::endl;
00031                           if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                   return new ResultCombinator(mr);
00033                           else
00034                                   return new IDIncrementalAlgorithm(mr);
00035                   }
00036
00037                   MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038
00039                   if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040
00041                           IncrementalAlgorithm* incrAlgo = NULL;
00042
00043                           switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                   case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                           incrAlgo = new ConstrTree(mr);
00046                                           break;
00047                                   case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                           incrAlgo = new DestrTree(mr);
00049                                           break;
00050                                   case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                           incrAlgo = new ConstrTree(mr);
00052                                           break;
00053                           }
00054
00055                           return incrAlgo;
00056                   } else {
00057                           MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                   « " to the system yet." « std::endl;
00059                           return NULL;
00060                   }
00061           }
```

6.60.4 Field Documentation

6.60.4.1 attrList

```
const ParsedMinerule::AttrVector* minerule::IDIncrementalAlgorithm::attrList  [protected]
```

Definition at line 43 of file IDIncrementalAlgorithm.hpp.

6.60.4.2 minerule

```
const OptimizedMinerule* minerule::IncrementalAlgorithm::minerule  [protected], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.hpp.

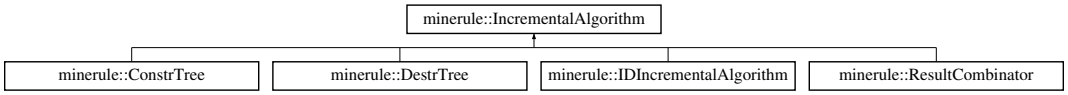
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IDIncrementalAlgorithm.hpp
- /Users/esposito/Software/minerule/src/Algorithms/IDIncrementalAlgorithm.cpp

6.61 minerule::IncrementalAlgorithm Class Reference

```
#include <IncrementalAlgorithm.hpp>
```

Inheritance diagram for minerule::IncrementalAlgorithm:



Public Member Functions

- IncrementalAlgorithm (const OptimizedMinerule &mr)
- virtual ~IncrementalAlgorithm ()
- virtual void execute ()=0

Static Public Member Functions

- static IncrementalAlgorithm * newIncrementalAlgorithm (const OptimizedMinerule &mr)

Protected Attributes

- const OptimizedMinerule * minerule

6.61.1 Detailed Description

Definition at line 23 of file IncrementalAlgorithm.hpp.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 IncrementalAlgorithm()

```
minerule::IncrementalAlgorithm::IncrementalAlgorithm (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 27 of file IncrementalAlgorithm.hpp.

```
00027                                                 :
00028     minerule(&mr) {}
```

6.61.2.2 ~IncrementalAlgorithm()

```
virtual minerule::IncrementalAlgorithm::~IncrementalAlgorithm ( )  [inline], [virtual]
```

Definition at line 30 of file IncrementalAlgorithm.hpp.

```
00030 {}
```

6.61.3 Member Function Documentation

6.61.3.1 execute()

```
virtual void minerule::IncrementalAlgorithm::execute ( )  [pure virtual]
```

Implemented in minerule::ConstrTree, minerule::DestrTree, minerule::IDIncrementalAlgorithm, and minerule::ResultCombinator.

6.61.3.2 newIncrementalAlgorithm()

```
IncrementalAlgorithm * minerule::IncrementalAlgorithm::newIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [static]
```

Definition at line 25 of file IncrementalAlgorithm.cpp.

```
00025                                                                                {
00026     // if mr has only ItemDependent constraints
00027                 MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028
00029                 if( mr.hasIDConstraints() ) {
00030                         MRLog() « "The minerule is item dependent!" « std::endl;
00031                         if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                 return new ResultCombinator(mr);
00033                         else
00034                                 return new IDIncrementalAlgorithm(mr);
00035                 }
00036
00037                 MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038
00039                 if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040
00041                         IncrementalAlgorithm* incrAlgo = NULL;
00042
00043                         switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                 case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                         incrAlgo = new ConstrTree(mr);
00046                                         break;
00047                                 case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                         incrAlgo = new DestrTree(mr);
00049                                         break;
00050                                 case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                         incrAlgo = new ConstrTree(mr);
00052                                         break;
00053                         }
00054
00055                         return incrAlgo;
00056                 } else {
00057                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                 « " to the system yet." « std::endl;
00059                         return NULL;
00060                 }
00061         }
```

6.61.4 Field Documentation

6.61.4.1 minerule

```
const OptimizedMinerule* minerule::IncrementalAlgorithm::minerule  [protected]
```

Definition at line 25 of file IncrementalAlgorithm.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrementalAlgorithm.hpp
- /Users/esposito/Software/minerule/src/Algorithms/IncrementalAlgorithm.cpp

6.62 minerule::Interval Class Reference

```
#include <Interval.hpp>
```

Public Types

- enum Operator {
  Less , LessEq , Grt , GrtEq ,
  Eq , NotEq }

Public Member Functions

- Interval ()
- Interval (const Interval &rhs)
- Interval (Operator op, const char ∗val, SQLUtils::Type t)
- void intersect (Interval &i)
- void update (const list_AND_node ∗l, bool negateIt)
- void setType (SQLUtils::Type t)
- bool isEmpty ()

Static Public Member Functions

- static const char ∗ getValue (const char ∗val1, const char ∗val2)
- static const char ∗ getAttribute (const char ∗val1, const char ∗val2)

Protected Member Functions

- Operator getOperator (const char ∗op, bool negateIt) const
- int compareValues (const char ∗val1, const char ∗val2) const
- const char ∗ getMaxLwr (Interval &lhs, Interval &rhs, bool &maxIsOpen) const
- const char ∗ getMinUpp (Interval &lhs, Interval &rhs, bool &minIsOpen) const

Protected Attributes

- const char ∗ lwr
- const char ∗ upp
- bool lwrOpen
- bool uppOpen
- SQLUtils::Type type
- bool typeOk
- std::set< const char ∗, StringCompare > openPoints

Static Protected Attributes

- static const char ∗ POSINFTY ="+"
- static const char ∗ NEGINFTY ="-"

Friends

- std::ostream & operator<< (std::ostream &, const Interval &)

6.62.1 Detailed Description

Definition at line 48 of file Interval.hpp.

6.62.2 Member Enumeration Documentation

6.62.2.1 Operator

enum minerule::Interval::Operator

An enumerated type corresponding to the operators <,<=,>,>=,=,<>

Enumerator

Less	
LessEq	
Grt	
GrtEq	
Eq	
NotEq	

Definition at line 54 of file Interval.hpp.

```
00054                 {
00055       Less,
00056       LessEq,
00057       Grt,
00058       GrtEq,
00059       Eq,
00060       NotEq
00061     } Operator;
```

6.62.3 Constructor & Destructor Documentation

6.62.3.1 Interval() [1/3]

minerule::Interval::Interval ( ) [inline]

Definition at line 164 of file Interval.hpp.

```
00164 : lwr(NEGINFTY), upp(POSINFTY), lwrOpen(true), uppOpen(true), typeOk(false) { }
```

6.62.3.2 Interval() [2/3]

```
minerule::Interval::Interval (
            const Interval & rhs )  [inline]
```

Definition at line 165 of file Interval.hpp.

```
00165                                   :
00166       lwr(rhs.lwr),
00167       upp(rhs.upp),
00168       lwrOpen(rhs.lwrOpen),
00169       uppOpen(rhs.uppOpen),
00170       type(rhs.type),
00171       typeOk(rhs.typeOk) {}
```

6.62.3.3 Interval() [3/3]

```
minerule::Interval::Interval (
            Operator op,
            const char * val,
            SQLUtils::Type t )
```

Definition at line 39 of file Interval.cpp.

```
00039                                                                   {
00040     upp=POSINFTY;
00041     lwr=NEGINFTY;
00042     uppOpen=lwrOpen=true;
00043     setType(t);
00044
00045     switch(op) {
00046     case Less:
00047       upp=val;
00048       break;
00049     case LessEq:
00050       upp=val;
00051       uppOpen=false;
00052       break;
00053     case Grt:
00054       lwr=val;
00055       break;
00056     case GrtEq:
00057       lwr=val;
00058       lwrOpen=false;
00059       break;
00060     case Eq:
00061       lwr=val;
00062       upp=val;
00063       lwrOpen=false;
00064       uppOpen=false;
00065       break;
00066     case NotEq:
00067       openPoints.insert(val);
00068       break;
00069     }
00070   }
```

6.62.4 Member Function Documentation

6.62.4.1 compareValues()

```
int minerule::Interval::compareValues (
            const char * val1,
            const char * val2 ) const  [protected]
```

Returns -1 if val1<val2 0 if val1=val2 and 1 if val1>val2. The comparison is taken w.r.t. the type reported by the "type" member variable.

Definition at line 102 of file Interval.cpp.

```
00102                                                                    {
00103     assert(typeOk);
00104
00105     if(val1==NEGINFTY) {
00106       if(val2==NEGINFTY)
00107         return 0;
00108       else
00109         return -1;
00110     }
00111     if(val1==POSINFTY) {
00112       if(val2==POSINFTY)
00113         return 0;
00114       else
00115         return 1;
00116     }
00117     if(val2==NEGINFTY)
00118       return 1;
00119     if(val2==POSINFTY)
00120       return -1;
00121
00122     if( type==SQLUtils::String) {
00123       return strcmp(val1,val2);
00124     } else if( type==SQLUtils::Numeric ) {
00125       return int(Converter(val1).toDouble()-Converter(val2).toDouble());
00126     } else {
00127       throw MineruleException(MR_ERROR_INTERNAL,
00128                               "SimplePredAnalyzer still does not support SQL types"
00129                               " which differ fromstd::string and numeric");
00130     }
00131   }
```

6.62.4.2 getAttribute()

```
const char * minerule::Interval::getAttribute (
            const char * val1,
            const char * val2 )  [static]
```

Parameters

val1	
val2	

Returns one among val1 and val2. In particular it returns the first of the two which is found to be an attribute identifier. For instance, if val1=="A" val2=="5" the function returns "A".

Definition at line 174 of file Interval.cpp.

```
00174                                                                    {
00175     if( SQLUtils::isAttribute(val1) )
00176       return val1;
00177
00178     if( SQLUtils::isAttribute(val2) )
00179       return val2;
00180
00181     return NULL;
00182   }
```

6.62.4.3 getMaxLwr()

```
const char * minerule::Interval::getMaxLwr (
             Interval & lhs,
             Interval & rhs,
             bool & maxIsOpen ) const  [protected]
```

Parameters

lhs	an Interval
rhs	an Interval
maxIsOpen	a boolean used as an additional return value

Returns the highest between lhs.lwr and rhs.lwr, in addition it sets maxIsOpen accordingly to whether the corresponding interval is open on the left or not. Example: let us denote an Interval with the usual mathematical notation. getMinUpp((10,20], [11,15)) returns 11 since 11 is the highest lwr value, and sets maxIsOpen=false since [11,15) is closed on the left side.

Definition at line 133 of file Interval.cpp.

```
00133                                                                    {
00134     int order = compareValues(lhs.lwr, rhs.lwr);
00135
00136     if( order<0 ) {
00137       maxIsOpen = rhs.lwrOpen;
00138       return rhs.lwr;
00139     } else if(order>0) {
00140       maxIsOpen = lhs.lwrOpen;
00141       return lhs.lwr;
00142     } else {
00143       maxIsOpen = lhs.lwrOpen || rhs.lwrOpen;
00144       return lhs.lwr;
00145     }
00146   }
```

6.62.4.4 getMinUpp()

```
const char * minerule::Interval::getMinUpp (
             Interval & lhs,
             Interval & rhs,
             bool & minIsOpen ) const  [protected]
```

Parameters

lhs	an Interval
rhs	an Interval
minIsOpen	a boolean used as an additional return value

Returns the lowest between lhs.upp and rhs.upp, in addition it sets minIsOpen accordingly to whether the corresponding interval is open on the right or not. Example: let us denote an Interval with the usual mathematical notation. getMinUpp((10,20], [11,15)) returns 15 since 15 is the lowest upp value, and sets minIsOpen=true since [11,15) is open on the right side.

Definition at line 148 of file Interval.cpp.

```
00148                                                                    {
00149     int order = compareValues(lhs.upp, rhs.upp);
00150
```

```
00151     if( order<0 ) {
00152       minIsOpen = lhs.uppOpen;
00153       return lhs.upp;
00154     } else if( order>0 ){
00155       minIsOpen = rhs.uppOpen;
00156       return rhs.upp;
00157     } else {
00158       minIsOpen = lhs.uppOpen || rhs.uppOpen;
00159       return lhs.upp;
00160     }
00161   }
```

6.62.4.5 getOperator()

```
Interval::Operator minerule::Interval::getOperator (
            const char * op,
            bool negateIt ) const  [protected]
```

Parameters

op	Astd::string representing a supported operator (that is, one of "<,<=,>,>=,=,<>")
negateIt	If true, the function returns the symbolic representation of the negated operator

Returns the symbolic representation of the operator described by op. If negateIt is true, the results corresponds to the negation of the operator, e.g., getOperator("<=",false) returns LessEq; getOperator("<=",true) returns Grt;

Definition at line 72 of file Interval.cpp.

```
00072                                                                    {
00073     static const char* errMsg = "Do not know how to handle operator: ";
00074     bool twoCharOp = op[1]!='\0';
00075     if(op[0]=='<') {
00076       if(!twoCharOp)
00077         return negateIt?GrtEq:Less;
00078       else if(op[1]=='>')
00079         return negateIt?Eq:NotEq;
00080       else if(op[1]=='=')
00081         return negateIt?Grt:LessEq;
00082       else throw MineruleException( MR_ERROR_OPTIMIZER_ERROR,
00083                                     errMsg+ std::string(op) );
00084     } else if( op[0]=='>' ) {
00085       if(!twoCharOp)
00086         return negateIt ? LessEq : Grt;
00087       else if(op[1]=='=')
00088         return negateIt ? Less : GrtEq;
00089       else throw MineruleException(  MR_ERROR_OPTIMIZER_ERROR,
00090                                      errMsg+ std::string(op) );
00091
00092     }   else if( op[0]=='=' ) {
00093       if(twoCharOp)
00094         throw  MineruleException(  MR_ERROR_OPTIMIZER_ERROR,
00095                                    errMsg+ std::string(op) );
00096       else
00097         return negateIt ? NotEq : Eq;
00098     } else throw MineruleException( MR_ERROR_OPTIMIZER_ERROR,
00099                                     errMsg+ std::string(op) );
00100   }
```

6.62.4.6 getValue()

```
const char * minerule::Interval::getValue (
            const char * val1,
            const char * val2 )  [static]
```

Parameters

val1	
val2	

Returns one among val1 and val2. In particular it returns the first of the two which is found not to be an attribute identifier. For instance, if val1=="A" val2=="5" the function returns "5".

Definition at line 164 of file Interval.cpp.

```
00164                                                                    {
00165     if( !SQLUtils::isAttribute(val1) )
00166       return val1;
00167
00168     if( !SQLUtils::isAttribute(val2) )
00169       return val2;
00170
00171     return NULL;
00172   }
```

6.62.4.7 intersect()

```
void minerule::Interval::intersect (
            Interval & i )  [inline]
```

Parameters

i	An Interval Intersects the current interval with the i.

Definition at line 179 of file Interval.hpp.

```
00179                                {
00180       lwr = getMaxLwr(*this,i,lwrOpen);
00181       upp = getMinUpp(*this,i,uppOpen);
00182       openPoints.insert( i.openPoints.begin(), i.openPoints.end() );
00183     }
```

6.62.4.8 isEmpty()

```
bool minerule::Interval::isEmpty ( )  [inline]
```

Returns true if the current interval is an empty one.

Definition at line 204 of file Interval.hpp.

```
00204                     {
00205       int order = compareValues(lwr,upp);
00206       if( order < 0 )  return false;
00207       if( order == 0 ) return lwrOpen || uppOpen || (openPoints.find(lwr)!=openPoints.end());
00208       return true;
00209     }
```

6.62.4.9 setType()

```
void minerule::Interval::setType (
            SQLUtils::Type t )  [inline]
```

It sets type = t and typeOk to true;

Definition at line 195 of file Interval.hpp.

```
00195                                   {
00196       type=t;
00197       typeOk = true;
00198     }
```

6.62.4.10 update()

```
void minerule::Interval::update (
            const list_AND_node * l,
            bool negateIt )
```

Intersects the current interval with the one built from l->sp

Definition at line 186 of file Interval.cpp.

```
00186                                                                {
00187     assert(l!=NULL && l->sp!=NULL &&
00188            l->sp->val1!=NULL && l->sp->op!=NULL && l->sp->val2!=NULL);
00189
00190     const char* value = getValue(l->sp->val1, l->sp->val2);
00191     if(value==NULL) // this is a cross condition, we just skip the update
00192       return;
00193
00194     Interval i( getOperator( l->sp->op, negateIt ), value, type );
00195     intersect(i);
00196   }
```

6.62.5 Friends And Related Function Documentation

6.62.5.1 operator<<

```
std::ostream & operator<< (
            std::ostream & ,
            const Interval &  )  [friend]
```

Definition at line 30 of file Interval.cpp.

```
00030                                                                {
00031     if( i.lwrOpen || i.openPoints.find(i.lwr)!=i.openPoints.end()) os « "("; else os « "[";
00032     os « i.lwr « "," « i.upp;
00033     if( i.uppOpen || i.openPoints.find(i.upp)!=i.openPoints.end()) os « ")"; else os « "]";
00034
00035     return os;
00036   }
```

6.62.6 Field Documentation

6.62.6.1 lwr

```
const char* minerule::Interval::lwr  [protected]
```

The lower part of the interval (e.g., if the object is used to represent (10,4], then lwr will contain thestd::string "10").

Definition at line 81 of file Interval.hpp.

6.62.6.2 lwrOpen

```
bool minerule::Interval::lwrOpen  [protected]
```

true if the interval is open on the left (e.g., if the object is used to represent (10,4], then lwrOpen is set to true)

Definition at line 91 of file Interval.hpp.

6.62.6.3 NEGINFTY

```
const char * minerule::Interval::NEGINFTY ="-"  [static], [protected]
```

Thestd::string used to represent negative infty. Notice that its content are not really important since, all the comparison will be made using the address of the string.

Definition at line 75 of file Interval.hpp.

6.62.6.4 openPoints

```
std::set<const char*, StringCompare> minerule::Interval::openPoints  [protected]
```

Used to keep tracks of isolated points that must not be considered to be valid ones (they corresponds to <> predicates)

Definition at line 115 of file Interval.hpp.

6.62.6.5 POSINFTY

```
const char * minerule::Interval::POSINFTY ="+"  [static], [protected]
```

Thestd::string used to represent positive infty. Notice that its content are not really important since, all the comparison will be made using the address of the string.

Definition at line 69 of file Interval.hpp.

6.62.6.6 type

```
SQLUtils::Type minerule::Interval::type  [protected]
```

The mrdb type of the attribute corresponding to this interval. It is used in order to compare values in all the the operations.

Definition at line 104 of file Interval.hpp.

6.62.6.7 typeOk

```
bool minerule::Interval::typeOk  [protected]
```

Used to keep track of whether type has been already setted or not.

Definition at line 109 of file Interval.hpp.

6.62.6.8 upp

```
const char* minerule::Interval::upp  [protected]
```

The upper part of the interval (e.g., if the object is used to represent (10,4], then upp will contain thestd::string "4").

Definition at line 86 of file Interval.hpp.

6.62.6.9 uppOpen

```
bool minerule::Interval::uppOpen  [protected]
```

true if the interval is open on the right (e.g., if the object is used to represent (10,4], then uppOpen is set to false)

Definition at line 97 of file Interval.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/Interval.cpp

6.63 minerule::IntervalChecker Class Reference

```
#include <Interval.hpp>
```

Public Member Functions

- IntervalChecker (const std::string &t)
- SQLUtils::Type typeForAttribute (const char *attr)
- bool impossibleVariableSetting (const VarSet &vset, const list_AND_node *l)

6.63.1 Detailed Description

Definition at line 244 of file Interval.hpp.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 IntervalChecker()

```
minerule::IntervalChecker::IntervalChecker (
            const std::string & t )  [inline]
```

Definition at line 249 of file Interval.hpp.

```
00249 : tab_source(t) {}
```

6.63.3 Member Function Documentation

6.63.3.1 impossibleVariableSetting()

```
bool minerule::IntervalChecker::impossibleVariableSetting (
            const VarSet & vset,
            const list_AND_node * l )
```

Parameters

l	a list of predicates

It iterates throug the list l maintaining a set of intervals corresponding to the attributes in l. It returns false as soon one of those intervals becomes empty. It returns true if at the end of the list all intervals are still non-empty.

Definition at line 223 of file Interval.cpp.

```
00223                                                                                {
00224     std::map<const char*, Interval, StringCompare> intervals;
00225     //    std::cout << std::endl;
00226
00227     while( l!=NULL ) {
00228       const char* attr = Interval::getAttribute( l->sp->val1, l->sp->val2 );
00229       if(attr!=NULL) {
00230         SimplePredicate& pred = SimplePredicate::newSimplePredicate(l->sp);
00231         Interval& i=intervals[attr];
00232         i.setType(typeForAttribute(attr));
```

```
00233         i.update(l, !vset.getVar(pred.getVarId()));
00234
00235         //      std::cout « l->sp->val1 « l->sp->op « l->sp->val2 « " ";
00236         //      std::cout « "id:" « pred.getVarId() « " v(id):" « vset.getVar(pred.getVarId()) « " ";
00237         //      std::cout « i « std::endl;
00238
00239         if(i.isEmpty())
00240           return true;
00241       };
00242
00243       l=l->next;
00244     }
00245
00246     //    std::cout « intervals["price"] « " ";
00247
00248     return false;
00249   }
```

6.63.3.2 typeForAttribute()

```
SQLUtils::Type minerule::IntervalChecker::typeForAttribute (
            const char * attr )
```

Parameters

attr	

It returns the type for attribute attr by searching the map typesMap. In case the type cannot be found in typesMap, then it try to determine it using SQLUtils and update the map accordingly. If it cannot determine the type, it throw an exception.

Definition at line 203 of file Interval.cpp.

```
00203                                                                    {
00204     std::map<const char*, SQLUtils::Type>::const_iterator it =
00205       typesMap.find(attr);
00206
00207     SQLUtils::Type t;
00208     if(it!=typesMap.end()) {
00209       t = it->second;
00210     } else {
00211       t = SQLUtils::getType( MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection(),
00212                              tab_source,
00213                              attr );
00214
00215       typesMap[attr] = t;
00216     }
00217
00218     return t;
00219   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/Interval.cpp

6.64 minerule::InvalidConfigurationFilter Class Reference

```
#include <InvalidConfigurationFilter.hpp>
```

Public Member Functions

- InvalidConfigurationFilter (const std::string &t, list_AND_node *l)
- bool operator() (const VarSet &vset)

6.64.1 Detailed Description

Definition at line 26 of file InvalidConfigurationFilter.hpp.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 InvalidConfigurationFilter()

```
minerule::InvalidConfigurationFilter::InvalidConfigurationFilter (
            const std::string & t,
            list_AND_node * l )  [inline]
```

Definition at line 31 of file InvalidConfigurationFilter.hpp.

```
00032                                                 :
00033       tab_source(t),
00034       preds(l),
00035       ic(t) {}
```

6.64.3 Member Function Documentation

6.64.3.1 operator()()

```
bool minerule::InvalidConfigurationFilter::operator() (
            const VarSet & vset )  [inline]
```

returns true if the vset configuration must be filtered out

Definition at line 40 of file InvalidConfigurationFilter.hpp.

```
00040                                       {
00041       return ic.impossibleVariableSetting( vset, preds );
00042     }
```

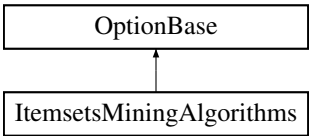
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/InvalidConfigurationFilter.hpp

6.65 ItemsetsMiningAlgorithms Class Reference

`#include <itemsetmining.hpp>`

Inheritance diagram for ItemsetsMiningAlgorithms:



Public Member Functions

- ItemsetsMiningAlgorithms ()
- ItemsetsMiningAlgorithms (const ItemsetsMiningAlgorithms &rhs)
- virtual ~ItemsetsMiningAlgorithms ()
- void setPreferredAlgorithm (AlgorithmTypes type)
- AlgorithmTypes getPreferredAlgorithm () const
- virtual std::string className () const
- virtual OptionBase & subclassForName (const std::string &subclassName)
- virtual void setOption (const std::string &name, const std::string &value)

6.65.1 Detailed Description

Definition at line 16 of file itemsetmining.hpp.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 ItemsetsMiningAlgorithms() [1/2]

```
ItemsetsMiningAlgorithms::ItemsetsMiningAlgorithms ( )  [inline]
```

Definition at line 19 of file itemsetmining.hpp.

```
00019 : preferredAlgorithm(ATNone) {  }
```

6.65.2.2 ItemsetsMiningAlgorithms() [2/2]

```
ItemsetsMiningAlgorithms::ItemsetsMiningAlgorithms (
            const ItemsetsMiningAlgorithms & rhs )  [inline]
```

Definition at line 21 of file itemsetmining.hpp.

```
00021                                                                 :
00022          preferredAlgorithm(rhs.preferredAlgorithm) {};
```

6.65.2.3 ~ItemsetsMiningAlgorithms()

```
virtual ItemsetsMiningAlgorithms::~ItemsetsMiningAlgorithms ( )  [inline], [virtual]
```

Definition at line 24 of file itemsetmining.hpp.

```
00024 {}
```

6.65.3 Member Function Documentation

6.65.3.1 className()

```
virtual std::string ItemsetsMiningAlgorithms::className ( ) const  [inline], [virtual]
```

Definition at line 34 of file itemsetmining.hpp.

```
00034                                          {
00035           return "itemsetsmining";
00036         }
```

6.65.3.2 getPreferredAlgorithm()

```
AlgorithmTypes ItemsetsMiningAlgorithms::getPreferredAlgorithm ( ) const  [inline]
```

Definition at line 30 of file itemsetmining.hpp.

```
00030                                                  {
00031           return preferredAlgorithm;
00032         }
```

6.65.3.3 setOption()

```
virtual void ItemsetsMiningAlgorithms::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.65.3.4 setPreferredAlgorithm()

```
void ItemsetsMiningAlgorithms::setPreferredAlgorithm (
            AlgorithmTypes type )  [inline]
```

Definition at line 26 of file itemsetmining.hpp.

```
00026                                                    {
00027           preferredAlgorithm = type;
00028         }
```

6.65.3.5 subclassForName()

```
virtual OptionBase & ItemsetsMiningAlgorithms::subclassForName (
            const std::string & subclassName )  [inline], [virtual]
```

Definition at line 39 of file itemsetmining.hpp.

```
00040           {
00041             return OptionBase::subclassForName(subclassName);
00042         }
```

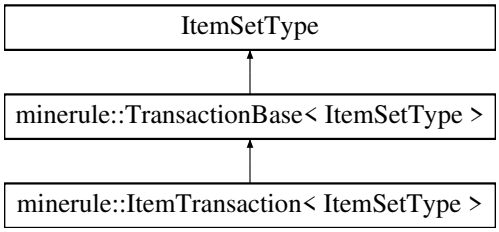
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/itemsetmining.hp

6.66 minerule::ItemTransaction< ItemSetType > Class Template Reference

`#include <Transaction.hpp>`

Inheritance diagram for minerule::ItemTransaction< ItemSetType >:



Public Member Functions

- ItemTransaction ()
- void loadBody (ItemType &gid, SourceTable::Iterator &it)
- void loadHead (ItemType &gid, SourceTable::Iterator &it)

Static Public Member Functions

- static bool findGid (ItemType &gid, SourceTable::Iterator &it)

6.66.1 Detailed Description

template<class ItemSetType>
class minerule::ItemTransaction< ItemSetType >

Definition at line 46 of file Transaction.hpp.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 ItemTransaction()

```
template<class ItemSetType >
minerule::ItemTransaction< ItemSetType >::ItemTransaction ( )  [inline]
```

Definition at line 48 of file Transaction.hpp.

```
00048 : TransactionBase<ItemSetType>() {}
```

6.66.3 Member Function Documentation

6.66.3.1 findGid()

```
static bool minerule::TransactionBase< ItemSetType >::findGid (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [static], [inherited]
```

Definition at line 32 of file Transaction.hpp.

```
00032                                                                   {
00033                         while( !it.isAfterLast() && gid > it->getGroup() ) {
00034                                 ++it;
00035                         }
00036
00037                         return !it.isAfterLast() && gid == it->getGroup();
00038                 }
```

6.66.3.2 loadBody()

```
template<class ItemSetType >
void minerule::ItemTransaction< ItemSetType >::loadBody (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline]
```

Definition at line 50 of file Transaction.hpp.

```
00050                                                                   {
00051                         while (!it.isAfterLast() && gid == it->getGroup()) {
00052
00053                                 ItemSetType::insert(it->getBody());
00054                                 ++it;
00055                         }
00056                 }
```

6.66.3.3 loadHead()

```
template<class ItemSetType >
void minerule::ItemTransaction< ItemSetType >::loadHead (
             ItemType & gid,
             SourceTable::Iterator & it )  [inline]
```

Definition at line 58 of file Transaction.hpp.

```
00058                                                                {
00059                         while (!it.isAfterLast() && gid == it->getGroup()) {
00060 
00061                                 ItemSetType::insert(it->getHead());
00062                                 ++it;
00063                         }
00064                 }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/Transaction.hpp

6.67 minerule::ItemType Class Reference

`#include <ItemType.hpp>`

Public Member Functions

- ItemType ()
- virtual ~ItemType ()
- ItemType (const ItemType &i)
- ItemType (const SourceRowElement &srel)
- ItemType (SourceRowElement *srel)
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const
- std::string getFullElementType () const
- void setSourceRowElement (SourceRowElement &srel)
- ItemType & operator= (const ItemType &i)
- ItemType & operator= (const SourceRowElement &i)
- bool operator() (const ItemType &it1, const ItemType &it2) const
- bool operator< (const ItemType &it) const
- bool operator< (const SourceRowElement &elem) const
- bool operator> (const ItemType &it) const
- bool operator> (const SourceRowElement &elem) const
- bool operator== (const ItemType &it) const
- bool operator== (const SourceRowElement &elem) const
- bool operator!= (const ItemType &it) const
- bool operator!= (const SourceRowElement &elem) const
- std::string getSQLData () const
- std::string asString () const
- minerule::SourceRowElement & getElement () const

Static Public Member Functions

- static bool lessThan (const ItemType &, const ItemType &)

Friends

- std::ostream & operator<< (std::ostream &, const ItemType &)

6.67.1 Detailed Description

Definition at line 40 of file ItemType.hpp.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 ItemType() [1/4]

```
minerule::ItemType::ItemType ( )  [inline]
```

Definition at line 44 of file ItemType.hpp.

```
00044                           : el(NULL) {
00045                 }
```

6.67.2.2 ~ItemType()

```
virtual minerule::ItemType::~ItemType ( )  [inline], [virtual]
```

Definition at line 47 of file ItemType.hpp.

```
00047                                   {
00048                         if(el!=NULL)
00049                                 delete el;
00050                 }
```

6.67.2.3 ItemType() [2/4]

```
minerule::ItemType::ItemType (
            const ItemType & i )  [inline]
```

Definition at line 52 of file ItemType.hpp.

```
00052                                           {
00053                         if( i.el==NULL)
00054                                 el=NULL;
00055                         else
00056                                 el=i.el->copy();
00057                 }
```

6.67.2.4 ItemType() [3/4]

```
minerule::ItemType::ItemType (
            const SourceRowElement & srel )  [inline]
```

Definition at line 61 of file ItemType.hpp.

```
00061                                                   {
00062                         el = srel.copy();
00063                 }
```

6.67.2.5 ItemType() [4/4]

```
minerule::ItemType::ItemType (
            SourceRowElement * srel )  [inline]
```

Definition at line 66 of file ItemType.hpp.

```
00066 : el(srel) {}
```

6.67.3 Member Function Documentation

6.67.3.1 asString()

```
std::string minerule::ItemType::asString ( ) const  [inline]
```

Definition at line 199 of file ItemType.hpp.

```
00199                                           {
00200                         assert(el!=NULL);
00201                         return el->asString();
00202                 }
```

6.67.3.2 getElement()

```
minerule::SourceRowElement & minerule::ItemType::getElement ( ) const  [inline]
```

Definition at line 204 of file ItemType.hpp.

```
00204                                                   {
00205                         if(el==NULL)
00206                                 throw MineruleException(MR_ERROR_INTERNAL, "Accessing a NULL
      element!");
00207                         return *el;
00208                 }
```

6.67.3.3 getFullElementType()

```
std::string minerule::ItemType::getFullElementType ( ) const  [inline]
```

Definition at line 72 of file ItemType.hpp.

```
00072 { return el->getFullElementType(); }
```

6.67.3.4 getSQLData()

```
std::string minerule::ItemType::getSQLData ( ) const  [inline]
```

Definition at line 194 of file ItemType.hpp.

```
00194                                           {
00195                         assert(el!=NULL);
00196                         return el->getSQLData();
00197                 }
```

6.67.3.5 lessThan()

```
bool minerule::ItemType::lessThan (
              const ItemType & it1,
              const ItemType & it2 )  [inline], [static]
```

Definition at line 225 of file ItemType.hpp.

```
00225                                                                                   {
00226                 return it1<it2;
00227         }
```

6.67.3.6 operator"!=() [1/2]

```
bool minerule::ItemType::operator!= (
              const ItemType & it ) const  [inline]
```

Definition at line 174 of file ItemType.hpp.

```
00174                                                   {
00175                         if(el==NULL)
00176                                 return it.el!=NULL;
00177
00178                         if(it.el==NULL)
00179                                 return true;
00180
00181                         return *el != *it.el;
00182                 }
```

6.67.3.7 operator"!=() [2/2]

```
bool minerule::ItemType::operator!= (
            const SourceRowElement & elem ) const  [inline]
```

Definition at line 184 of file ItemType.hpp.

```
00184                                                                    {
00185                         if(el==NULL)
00186                                 return !elem.empty();
00187
00188                         if(elem.empty())
00189                                 return true;
00190
00191                         return *el != elem;
00192                 }
```

6.67.3.8 operator()()

```
bool minerule::ItemType::operator() (
            const ItemType & it1,
            const ItemType & it2 ) const  [inline]
```

Definition at line 104 of file ItemType.hpp.

```
00104                                                                                    {
00105                         if(it1.el==NULL)
00106                                 return it2.el!=NULL;
00107
00108                         if(it2.el==NULL)
00109                                 return false;
00110
00111                         return *it1.el < *it2.el;
00112                 }
```

6.67.3.9 operator<() [1/2]

```
bool minerule::ItemType::operator< (
            const ItemType & it ) const  [inline]
```

Definition at line 114 of file ItemType.hpp.

```
00114                                                          {
00115                         if(el==NULL)
00116                                 return it.el!=NULL;
00117
00118                         if(it.el==NULL)
00119                                 return false;
00120
00121                         return *el < *it.el;
00122                 }
```

6.67.3.10 operator<() [2/2]

```
bool minerule::ItemType::operator< (
            const SourceRowElement & elem ) const  [inline]
```

Definition at line 124 of file ItemType.hpp.

```
00124                                                                    {
00125                         if(el==NULL)
00126                                 return !elem.empty();
00127
00128                         if(elem.empty())
00129                                 return false;
00130
00131                         return *el < elem;
00132                 }
```

6.67.3.11 operator=() [1/2]

```
ItemType & minerule::ItemType::operator= (
            const ItemType & i )  [inline]
```

Definition at line 82 of file ItemType.hpp.

```
00082                                                   {
00083                         if(el!=NULL)
00084                                 delete el;
00085
00086                         if(i.el!=NULL)
00087                                 el = i.el->copy();
00088                         else
00089                                 el = NULL;
00090
00091                         return *this;
00092                 }
```

6.67.3.12 operator=() [2/2]

```
ItemType & minerule::ItemType::operator= (
            const SourceRowElement & i )  [inline]
```

Definition at line 94 of file ItemType.hpp.

```
00094                                                           {
00095                         if(el!=NULL)
00096                                 delete el;
00097
00098                         el = i.copy();
00099
00100                        return *this;
00101                 }
```

6.67.3.13 operator==() [1/2]

```
bool minerule::ItemType::operator== (
            const ItemType & it ) const  [inline]
```

Definition at line 154 of file ItemType.hpp.

```
00154                                                   {
00155                         if(el==NULL)
00156                                 return it.el==NULL;
00157
00158                         if(it.el==NULL)
00159                                 return false;
00160
00161                         return *el == *it.el;
00162                 }
```

6.67.3.14 operator==() [2/2]

```
bool minerule::ItemType::operator== (
            const SourceRowElement & elem ) const  [inline]
```

Definition at line 164 of file ItemType.hpp.

```
00164                                                           {
00165                         if(el==NULL)
00166                                 return elem.empty();
00167
00168                         if(elem.empty())
00169                                 return false;
00170
00171                         return *el == elem;
00172                 }
```

6.67.3.15 operator>() [1/2]

```
bool minerule::ItemType::operator> (
            const ItemType & it ) const  [inline]
```

Definition at line 134 of file ItemType.hpp.

```
00134                                                   {
00135                         if(el==NULL)
00136                                 return false;
00137
00138                         if(it.el==NULL)
00139                                 return true;
00140
00141                         return *it.el < *el;
00142                 }
```

6.67.3.16 operator>() [2/2]

```
bool minerule::ItemType::operator> (
            const SourceRowElement & elem ) const  [inline]
```

Definition at line 144 of file ItemType.hpp.

```
00144                                                                   {
00145                         if(el==NULL)
00146                                 return false;
00147
00148                         if(elem.empty())
00149                                 return true;
00150
00151                         return elem < *el;
00152                 }
```

6.67.3.17 setPreparedStatementParameters()

```
virtual void minerule::ItemType::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [inline], [virtual]
```

Definition at line 68 of file ItemType.hpp.

```
00068
                {
00069                         el->setPreparedStatementParameters(state, start_index);
00070                 }
```

6.67.3.18 setSourceRowElement()

```
void minerule::ItemType::setSourceRowElement (
            SourceRowElement & srel )  [inline]
```

Definition at line 78 of file ItemType.hpp.

```
00078                                                                   {
00079                         el = &srel;
00080                 }
```

6.67.4 Friends And Related Function Documentation

6.67.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const ItemType & it )  [friend]
```

Definition at line 215 of file ItemType.hpp.

```
00215                                                                   {
00216                 if(it.el==NULL)
00217                         os«"NULL";
00218                 else
00219                         os « *it.el;
00220
00221                 return os ;
00222         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/ItemType.hpp

6.68 minerule::QueryResult::Iterator Class Reference

```
#include <QueryResult-header.hpp>
```

Public Member Functions

- Iterator ()
- ~Iterator ()
- void init (const std::string &rulesTable, double support, double confidence)
- bool next ()
- void getRule (Rule &r)

6.68.1 Detailed Description

Definition at line 39 of file QueryResult-header.hpp.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 Iterator()

```
minerule::QueryResult::Iterator::Iterator ( )  [inline]
```

Definition at line 48 of file QueryResult-header.hpp.

```
00048 : state(NULL), rs_rules(NULL), body_elems(NULL), head_elems(NULL) {}
```

6.68.2.2 ~Iterator()

```
minerule::QueryResult::Iterator::~Iterator ( )  [inline]
```

Definition at line 50 of file QueryResult-header.hpp.

```
00050                                    {
00051                                  if(rs_rules!=NULL)
00052                                          delete rs_rules;
00053
00054                                  if(state!=NULL)
00055                                          delete state;
00056
00057                                  if( body_elems!=NULL )
00058                                          delete body_elems;
00059
00060                                  if( head_elems!=NULL )
00061                                          delete head_elems;
00062                          }
```

6.68.3 Member Function Documentation

6.68.3.1 getRule()

```
void minerule::QueryResult::Iterator::getRule (
            Rule & r )
```

Definition at line 76 of file QueryResult.cpp.

```
00076                                                {
00077     ItemSet* body = new ItemSet();
00078     ItemSet* head = new ItemSet();
00079     size_t bid, hid;
00080
00081     bid = rs_rules->getInt(1);
00082     hid = rs_rules->getInt(2);
00083
00084     readElems( bid, *body, body_elems );
00085     readElems( hid, *head, head_elems );
00086
00087     r.setBody(body);
00088     r.setHead(head);
00089     r.setSupport(rs_rules->getDouble(3));
00090     r.setConfidence(rs_rules->getDouble(4));
00091     r.setBodyId(bid);
00092     r.setHeadId(hid);
00093   }
```

6.68.3.2 init()

```
void minerule::QueryResult::Iterator::init (
            const std::string & rulesTable,
            double support,
            double confidence )
```

Definition at line 44 of file QueryResult.cpp.

```
00046                       {
00047     mrdb::Connection* mrdb_conn = MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00048     Connection connection;
00049     connection.setOutTableName(rulesTable);
00050
00051     state=mrdb_conn->createStatement();
00052     std::string query =
```

```
00053     "SELECT bodyId, headId, supp, con "
00054     "FROM " + connection.getTableName(Connection::RulesTable) + " "
00055     "WHERE supp>=" + Converter(support).toString() + " AND "
00056     "con>="+Converter(confidence).toString();
00057
00058     body_elems = mrdb_conn->prepareStatement(
00059     "SELECT * FROM "+ connection.getTableName(Connection::BodiesTable) +
00060     " WHERE id=?");
00061
00062     head_elems = mrdb_conn->prepareStatement(
00063     "SELECT * FROM "+ connection.getTableName(Connection::HeadsTable) +
00064     " WHERE id=?");
00065
00066
00067     MRDebug() « "QueryResult::Iterator, the filter query is:" « query « std::endl;
00068
00069     rs_rules = state->executeQuery(query);
00070   }
```

6.68.3.3 next()

```
bool minerule::QueryResult::Iterator::next ( )
```

Definition at line 72 of file QueryResult.cpp.

```
00072                                  {
00073     return rs_rules->next();
00074   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp
- /Users/esposito/Software/minerule/src/Result/QueryResult.cpp

6.69 minerule::SourceTable::Iterator Class Reference

```
#include <SourceTable.hpp>
```

Public Member Functions

- Iterator ()
- Iterator (const Iterator &it)
- virtual ~Iterator ()
- bool next ()
- bool isAfterLast () const
- SourceRow * get ()
- SourceRow * operator-> ()
- Iterator & operator++ ()

Friends

- class SourceTable

6.69.1 Detailed Description

Definition at line 28 of file SourceTable.hpp.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 Iterator() [1/2]

```
minerule::SourceTable::Iterator::Iterator ( )  [inline]
```

Definition at line 48 of file SourceTable.hpp.

```
00048 : _resultSet(NULL), _columnIds(), _sourceRow(NULL) { }
```

6.69.2.2 Iterator() [2/2]

```
minerule::SourceTable::Iterator::Iterator (
            const Iterator & it )  [inline]
```

Definition at line 49 of file SourceTable.hpp.

```
00050         : _resultSet(it._resultSet), _columnIds(it._columnIds),
00051           _sourceRow(it._sourceRow) {  }
```

6.69.2.3 ~Iterator()

```
virtual minerule::SourceTable::Iterator::~Iterator ( )  [inline], [virtual]
```

Definition at line 53 of file SourceTable.hpp.

```
00053                          {
00054     }
```

6.69.3 Member Function Documentation

6.69.3.1 get()

```
SourceRow * minerule::SourceTable::Iterator::get ( )  [inline]
```

Definition at line 59 of file SourceTable.hpp.

```
00059                        {
00060       assert(_sourceRow != NULL);
00061       return _sourceRow;
00062     }
```

6.69.3.2 isAfterLast()

```
bool minerule::SourceTable::Iterator::isAfterLast ( ) const
```

Definition at line 40 of file SourceTable.cpp.

```
00040                                                   {
00041                 assert( _resultSet!=NULL );
00042                 return _resultSet->isAfterLast();
00043         }
```

6.69.3.3 next()

```
bool minerule::SourceTable::Iterator::next ( )
```

Definition at line 28 of file SourceTable.cpp.

```
00028                                           {
00029                 ++_rowCounter;
00030                 assert( _sourceRow != NULL && _resultSet != NULL );
00031
00032                 if( _resultSet->next() ) {
00033                         _sourceRow->init(_resultSet, _columnIds);
00034                         return true;
00035                 } else {
00036                         return false;
00037                 }
00038         }
```

6.69.3.4 operator++()

```
Iterator & minerule::SourceTable::Iterator::operator++ ( )  [inline]
```

Definition at line 65 of file SourceTable.hpp.

```
00065                           {
00066       next();
00067       return *this;
00068     }
```

6.69.3.5 operator->()

```
SourceRow * minerule::SourceTable::Iterator::operator-> ( )  [inline]
```

Definition at line 63 of file SourceTable.hpp.

```
00063 { return get(); }
```

6.69.4 Friends And Related Function Documentation

6.69.4.1 SourceTable

```
friend class SourceTable  [friend]
```

Definition at line 29 of file SourceTable.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceTable.hpp
- /Users/esposito/Software/minerule/src/Database/SourceTable.cpp

6.70 minerule::FSTreeSequence::less_sequence Struct Reference

```
#include <FSTreeSequence.hpp>
```

Public Member Functions

- bool operator() (const FSTreeSequence &first, const FSTreeSequence &second) const
- bool comp (const FSTreeSequence &first, const FSTreeSequence &second) const

6.70.1 Detailed Description

this struct define the order between two FSTreeSequence

Definition at line 41 of file FSTreeSequence.hpp.

6.70.2 Member Function Documentation

6.70.2.1 comp()

```
bool minerule::FSTreeSequence::less_sequence::comp (
            const FSTreeSequence & first,
            const FSTreeSequence & second ) const  [inline]
```

Definition at line 46 of file FSTreeSequence.hpp.

```
00046                                                                    {
00047         return (first < second);
00048       }
```

6.70.2.2 operator()()

```
bool minerule::FSTreeSequence::less_sequence::operator() (
            const FSTreeSequence & first,
            const FSTreeSequence & second ) const  [inline]
```

Definition at line 42 of file FSTreeSequence.hpp.

```
00042                                                                                                {
00043                 return (first < second);
00044         }
```

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeSequence.hpp

6.71 mrc::Options::ListFormat Class Reference

```
#include <Options.hpp>
```

Public Member Functions

- ListFormat ()

Data Fields

- bool size
- bool text
- bool result

6.71.1 Detailed Description

Definition at line 28 of file Options.hpp.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 ListFormat()

```
mrc::Options::ListFormat::ListFormat ( )  [inline]
```

Definition at line 34 of file Options.hpp.

```
00034 : size(false),      text(false), result(false) {}
```

6.71.3 Field Documentation

6.71.3.1 result

```
bool mrc::Options::ListFormat::result
```

Definition at line 32 of file Options.hpp.

6.71.3.2 size

```
bool mrc::Options::ListFormat::size
```

Definition at line 30 of file Options.hpp.

6.71.3.3 text

```
bool mrc::Options::ListFormat::text
```

Definition at line 31 of file Options.hpp.

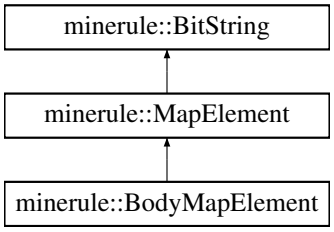
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.hpp

6.72 minerule::MapElement Class Reference

```
#include <Bodymap.hpp>
```

Inheritance diagram for minerule::MapElement:



Public Member Functions

- void insert (const int x)
- MapElement ()
- void operator= (const BitString &bs)
- boolean test (int i) const
- BitString & set ()
- BitString & set (int i, boolean value=true)
- BitString & reset ()
- BitString & reset (int i)
- BitString & clear ()
- BitString & clear (int i)
- BitString & invert ()
- BitString & invert (int i)
- BitString & operator&= (const BitString &bs)
- BitString & operator|= (const BitString &bs)
- BitString & operator^= (const BitString &bs)
- boolean operator== (const BitString &bs)
- boolean operator!= (const BitString &bs)
- BitString operator& (const BitString &bs1)
- int count (boolean what=true) const
- bool moreThan (double threshold) const
- int length () const
- int ssize () const
- int size () const
- void serialize (char *serialized, int *start)
- void unserialize (char *serialized, int *start)
- void print ()
- void print (std::ostream &out)
- void setNBit (int num)
- void * getElmA (int which)
- Bits getElm (int which)
- void insert (Bits element)
- std::vector< Bits > & gbits ()
- char operator[ ] (int which) const

Data Fields

- int counter

Static Public Attributes

- static int intersections = 0

6.72.1 Detailed Description

Definition at line 49 of file Bodymap.hpp.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 MapElement()

```
minerule::MapElement::MapElement ( )  [inline]
```

Definition at line 53 of file Bodymap.hpp.

```
00053 : counter(0) {}
```

6.72.3 Member Function Documentation

6.72.3.1 clear() [1/2]

```
BitString & minerule::BitString::clear ( )  [inherited]
```

Definition at line 121 of file Bitstring.cpp.

```
00122 {
00123 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00124 return *this;
00125 }
```

6.72.3.2 clear() [2/2]

```
BitString & minerule::BitString::clear (
            int i )  [inherited]
```

Definition at line 131 of file Bitstring.cpp.

```
00132 {
00133 return set(i, false);
00134 }
```

6.72.3.3 count()

```
int minerule::BitString::count (
            boolean what = true ) const  [inherited]
```

Definition at line 284 of file Bitstring.cpp.

```
00285 {
00286 int i,j, k=0;
00287 for (i=0; i<=Nw; i++) {
00288 //      Bits b = bits[i];
00289 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b>>=1;}
00290         for (j=0; j<sizeof(Bits); j++)
00291         k += NBITS[*(((unsigned char*)&bits[i])+j)];
00292 }
00293 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00294 return k;
00295 }
```

6.72.3.4 gbits()

```
std::vector< Bits > & minerule::BitString::gbits ( )  [inline], [inherited]
```

Definition at line 101 of file Bitstring.hpp.

```
00101 {  return bits; }
```

6.72.3.5 getElm()

```
Bits minerule::BitString::getElm (
            int which )  [inline], [inherited]
```

Definition at line 99 of file Bitstring.hpp.

```
00099 {  return bits[which]; }
```

6.72.3.6 getElmA()

```
void * minerule::BitString::getElmA (
            int which )  [inline], [inherited]
```

Definition at line 98 of file Bitstring.hpp.

```
00098 {  return &(bits[which]); }
```

6.72.3.7 insert() [1/2]

```
void minerule::BitString::insert (
            Bits element )  [inline], [inherited]
```

Definition at line 100 of file Bitstring.hpp.

```
00100 {  bits.push_back(element); }
```

6.72.3.8 insert() [2/2]

```
void minerule::MapElement::insert (
            const int x )  [inline]
```

Definition at line 52 of file Bodymap.hpp.

```
00052 { set(x,true); }
```

6.72.3.9 invert() [1/2]

```
BitString & minerule::BitString::invert ( )  [inherited]
```

Definition at line 140 of file Bitstring.cpp.

```
00141 {
00142 for (int i=0; i<=Nw; i++) bits[i] = ~bits[i];
00143 return *this;
00144 }
```

6.72.3.10 invert() [2/2]

```
BitString & minerule::BitString::invert (
            int i )  [inherited]
```

Definition at line 150 of file Bitstring.cpp.

```
00151 {
00152 if (n <= i) _Xran(i);
00153 bits[i/Nb] ^= (Bits)1 « i%Nb;
00154 return *this;
00155 }
```

6.72.3.11 length()

```
int minerule::BitString::length ( ) const  [inline], [inherited]
```

Definition at line 87 of file Bitstring.hpp.

```
00087 { return n; }
```

6.72.3.12 moreThan()

```
bool minerule::BitString::moreThan (
            double threshold ) const  [inherited]
```

Definition at line 301 of file Bitstring.cpp.

```
00302 {
00303 int i,j, k=(int)(threshold);
00304 for (i=0; i<=Nw && k >= 0; i++) {
00305 //      Bits b = bits[i];
00306 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b»=1;}
00307         for (j=0; j<sizeof(Bits); j++)
00308         k -= NBITS[*(((unsigned char*)&bits[i])+j)];
00309 }
00310 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00311 return k<0;
00312 }
```

6.72.3.13 operator"!=()

```
boolean minerule::BitString::operator!= (
            const BitString & bs )  [inherited]
```

Definition at line 263 of file Bitstring.cpp.

```
00264 {
00265 return !operator==(bs);
00266 }
```

6.72.3.14 operator&()

```
BitString minerule::BitString::operator& (
            const BitString & bs1 )  [inherited]
```

Definition at line 223 of file Bitstring.cpp.

```
00224 {
00225         BitString bs;
00226         int min = Nw < bs1.Nw ? Nw : bs1.Nw;
00227         for (int i=0; i<=min; i++) bs.bits.insert(bs.bits.end(),bits[i] & bs1.bits[i]);
00228         bs.Nw = min;
00229         bs.n = n < bs1.n ? n : bs1.n;
00230         return bs;
00231 }
```

6.72.3.15 operator&=()

```
BitString & minerule::BitString::operator&= (
            const BitString & bs )  [inherited]
```

Definition at line 195 of file Bitstring.cpp.

```
00196 {
00197 /*
00198 if (n < bs.n) {
00199         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n);
00200         n = bs.n;
00201 }
00202 if (Nw < bs.Nw) {
00203         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), ~(char)0); }
00204         n = bs.n;
00205 }
00206 for (int i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00207 */
00208 if (Nw < bs.Nw) {
00209         for (int i=0; i<=Nw; i++) bits[i] &= bs.bits[i];
00210 } else {
00211         int i;
00212         for (i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00213         for (; i<=Nw; i++) bits[i] = (Bits)0;
00214 }
00215 intersections++;
00216 return *this;
00217 }
```

6.72.3.16 operator=()

```
void minerule::MapElement::operator= (
            const BitString & bs )  [inline]
```

Definition at line 54 of file Bodymap.hpp.

```
00054 { BitString::operator=(bs); }
```

6.72.3.17 operator==()

```
boolean minerule::BitString::operator== (
            const BitString & bs )  [inherited]
```

Definition at line 251 of file Bitstring.cpp.

```
00252 {
00253 boolean eq = n == bs.n;
00254 for (int i=0; i<Nw && eq; i++) eq = bits[i] == bs.bits[i];
00255 for (int i=(Nw-1)*Nb; i<n && eq; i++) eq = test(i) == bs.test(i);
00256 return eq;
00257 }
```

6.72.3.18 operator[]()

```
char minerule::BitString::operator[] (
            int which ) const  [inline], [inherited]
```

Definition at line 103 of file Bitstring.hpp.

```
00103 { return (test(which) ? '1' : '0'); }
```

6.72.3.19 operator^=()

```
BitString & minerule::BitString::operator^= (
            const BitString & bs )  [inherited]
```

Definition at line 178 of file Bitstring.cpp.

```
00179 {
00180 if (n < bs.n) {
00181         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n,bs.test(n));
00182         n = bs.n;
00183 }
00184 if (Nw < bs.Nw) {
00185         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), bs.bits[Nw]); }
00186 }
00187 for (int i=0; i<=bs.Nw; i++) bits[i] ^= bs.bits[i];
00188 return *this;
00189 }
```

6.72.3.20 operator"|=()

```
BitString & minerule::BitString::operator|= (
            const BitString & bs )  [inherited]
```

Definition at line 161 of file Bitstring.cpp.

```
00162 {
00163 if (n < bs.n) {
00164         for (; n < (Nw+1)*Nb && n < bs.n; n++) reset(n);
00165         n = bs.n;
00166 }
00167 if (Nw < bs.Nw) {
00168         for (; Nw <= bs.Nw; ) { Nw++; bits.insert(bits.end(), (Bits)0); }
00169 }
00170 for (int i=0; i<=bs.Nw; i++) bits[i] |= bs.bits[i];
00171 return *this;
00172 }
```

6.72.3.21 print() [1/2]

```
void minerule::BitString::print ( )  [inherited]
```

Definition at line 404 of file Bitstring.cpp.

```
00405 {int i,dim;
00406
00407  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00408  {
00409  dim=(*this).length();
00410  for (i=0;i<dim;i++)
00411  {
00412   if ((i%8==0)&&(i!=0)) std::cout«"-";
00413   std::cout«test(i);
00414  }
00415  }
00416 }
```

6.72.3.22 print() [2/2]

```
void minerule::BitString::print (
            std::ostream & out )  [inherited]
```

Definition at line 390 of file Bitstring.cpp.

```
00391 {int i,dim;
00392
00393  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00394  {
00395  dim=(*this).length();
00396  for (i=0;i<dim;i++)
00397  {
00398   if ((i%8==0)&&(i!=0)) out«"-";
00399   out«test(i);
00400  }
00401  }
00402 }
```

6.72.3.23 reset() [1/2]

```
BitString & minerule::BitString::reset ( )  [inherited]
```

Definition at line 102 of file Bitstring.cpp.

```
00103 {
00104 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00105 return *this;
00106 }
```

6.72.3.24 reset() [2/2]

```
BitString & minerule::BitString::reset (
            int i )  [inherited]
```

Definition at line 112 of file Bitstring.cpp.

```
00113 {
00114 return set(i, false);
00115 }
```

6.72.3.25 serialize()

```
void minerule::BitString::serialize (
            char * serialized,
            int * start )  [inherited]
```

Definition at line 352 of file Bitstring.cpp.

```
00353 {int i;
00354
00355  memcpy(&(serialized[(*start)]),&Nw,sizeof(int));
00356         //cout<<int(serialized[(*start)])<<" "<<std::endl;
00357  //cout<<"\"";
00358  (*start) += sizeof(int);
00359  memcpy(&(serialized[(*start)]),&n,sizeof(int));
00360  (*start) += sizeof(int);
00361         for (i=0;i<=Nw;i++)
00362         {
00363          memcpy(&(serialized[(*start)]),&(bits[i]),sizeof(Bits));
00364          //cout<<(int)(serialized[(*start)])<<" ";
00365                 (*start) += sizeof(Bits);
00366         }
00367 }
```

6.72.3.26 set() [1/2]

```
BitString & minerule::BitString::set ( )  [inherited]
```

Definition at line 80 of file Bitstring.cpp.

```
00081 {
00082 for (int i=0; i<=Nw; i++) bits[i] = ~(Bits)0;
00083 return *this;
00084 }
```

6.72.3.27 set() [2/2]

```
BitString & minerule::BitString::set (
            int i,
            boolean value = true )  [inherited]
```

Definition at line 90 of file Bitstring.cpp.

```
00091 {
00092 if (n <= i) _Xran(i);
00093 if (value) bits[i/Nb] |= (Bits)1 << i%Nb;
00094 else bits[i/Nb] &= ~((Bits)1 << i%Nb);
00095 return *this;
00096 }
```

6.72.3.28 setNBit()

```
void minerule::BitString::setNBit (
            int num )  [inline], [inherited]
```

Definition at line 97 of file Bitstring.hpp.

```
00097 {  Nw = (num == 0) ? 0 : (num-1) / Nb;n=num; }
```

6.72.3.29 size()

```
int minerule::BitString::size ( ) const  [inline], [inherited]
```

Definition at line 89 of file Bitstring.hpp.

```
00089 { return n; }
```

6.72.3.30 ssize()

```
int minerule::BitString::ssize ( ) const  [inline], [inherited]
```

Definition at line 88 of file Bitstring.hpp.

```
00088 { return bits.size(); }
```

6.72.3.31 test()

```
boolean minerule::BitString::test (
            int i ) const  [inline], [inherited]
```

Definition at line 58 of file Bitstring.hpp.

```
00058                                   {
00059                         if (i<0 || n <= i) return false;
00060                         return ((bits[i/Nb] & ((Bits)1 « i%Nb)) != 0);
00061                 }
```

6.72.3.32 unserialize()

```
void minerule::BitString::unserialize (
            char * serialized,
            int * start )  [inherited]
```

Definition at line 369 of file Bitstring.cpp.

```
00370 {int i;
00371  char tmpchar;
00372
00373  memcpy(&Nw,&(serialized[(*start)]),sizeof(int));
00374         //cout«int(serialized[(*start)])«" "«std::endl;
00375  //cout«"\"";
00376  (*start) += sizeof(int);
00377  memcpy(&n,&(serialized[(*start)]),sizeof(int));
00378  (*start) += sizeof(int);
00379  for (i=bits.size();i<=Nw;i++) bits.insert(bits.end(),(Bits)0);
00380         for (i=0;i<=Nw;i++)
00381         {
00382          memcpy(&tmpchar,&serialized[(*start)],sizeof(Bits));
00383          //cout«(int)(serialized[(*start)])«" ";
00384          bits[i] = tmpchar;
00385          (*start) += sizeof(Bits);
00386         }
00387 // setNBit(numbit);
00388 }
```

6.72.4 Field Documentation

6.72.4.1 counter

```
int minerule::MapElement::counter
```

Definition at line 51 of file Bodymap.hpp.

6.72.4.2 intersections

```
int minerule::BitString::intersections = 0  [static], [inherited]
```

Definition at line 55 of file Bitstring.hpp.

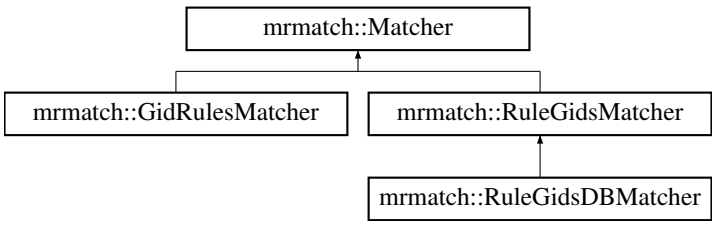
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp

6.73 mrmatch::Matcher Class Reference

```
#include <Matcher.hpp>
```

Inheritance diagram for mrmatch::Matcher:



Public Member Functions

- virtual minerule::Rule & addRule ()=0
- virtual void match (minerule::SourceTable &st)
- virtual void matchWithCrossProduct (minerule::SourceTable &st)
- virtual void matchWithoutCrossProduct (minerule::SourceTable &st)
- virtual void printMatches () const =0

Static Public Member Functions

- static Matcher * newMatcher (const Options &opts, const minerule::ParsedMinerule &p)

Protected Member Functions

- virtual void matchItemTransaction (const minerule::ItemType &gid, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &bodies, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &heads)=0
- virtual void matchRuleTransaction (const minerule::ItemType &gid, const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > &transaction)=0

6.73.1 Detailed Description

Definition at line 30 of file Matcher.hpp.

6.73.2 Member Function Documentation

6.73.2.1 addRule()

```
virtual minerule::Rule & mrmatch::Matcher::addRule ( )  [pure virtual]
```

Implemented in mrmatch::GidRulesMatcher, and mrmatch::RuleGidsMatcher.

6.73.2.2 match()

```
virtual void mrmatch::Matcher::match (
            minerule::SourceTable & st )  [inline], [virtual]
```

Definition at line 39 of file Matcher.hpp.

```
00039                                                       {
00040                       st.usesCrossProduct() ? matchWithCrossProduct(st) :
      matchWithoutCrossProduct(st);
00041                 }
```

6.73.2.3 matchItemTransaction()

```
virtual void mrmatch::Matcher::matchItemTransaction (
            const minerule::ItemType & gid,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & bodies,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & heads )
[protected], [pure virtual]
```

Implemented in mrmatch::GidRulesMatcher, and mrmatch::RuleGidsMatcher.

6.73.2.4 matchRuleTransaction()

```
virtual void mrmatch::Matcher::matchRuleTransaction (
            const minerule::ItemType & gid,
            const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > & transaction
)  [protected], [pure virtual]
```

Implemented in mrmatch::GidRulesMatcher, and mrmatch::RuleGidsMatcher.

6.73.2.5 matchWithCrossProduct()

```
void mrmatch::Matcher::matchWithCrossProduct (
            minerule::SourceTable & st )  [virtual]
```

Definition at line 38 of file Matcher.cpp.

```
00038                                                       {
00039                 SourceTable::Iterator it = st.newIterator(SourceTable::FullIterator);
00040
00041                 while(!it.isAfterLast()) {
00042                         ItemType gid = it->getGroup();
00043
00044                         RuleTransaction<RulesMatcher::RuleSetType> transaction;
00045                         transaction.load(gid, it);
00046
00047                         matchRuleTransaction(gid, transaction);
00048                 }
00049         }
```

6.73.2.6 matchWithoutCrossProduct()

```
void mrmatch::Matcher::matchWithoutCrossProduct (
            minerule::SourceTable & st )  [virtual]
```

Definition at line 51 of file Matcher.cpp.

```
00051                                                           {
00052                 SourceTable::Iterator bodyIt = st.newIterator(SourceTable::BodyIterator);
00053                 SourceTable::Iterator headIt = st.newIterator(SourceTable::HeadIterator);
00054
00055                 while(!bodyIt.isAfterLast()) {
00056                         ItemType gid = bodyIt->getGroup();
00057
00058                         ItemTransaction<RulesMatcher::ItemSetType> bodies;
00059                         ItemTransaction<RulesMatcher::ItemSetType> heads;
00060
00061                         bodies.loadBody(gid, bodyIt);                   // this advances the body
      iterator
00062
00063                         if( !TransactionBase<RulesMatcher::ItemSetType>::findGid(gid, headIt) ) {//
      positioning the head iterator
00064                                 break;                                                          // no
      more heads to load
00065                         }
00066
00067                         heads.loadHead(gid, headIt);                    // loading the heads
00068
00069                         matchItemTransaction(gid, bodies, heads);
00070                 } // while
00071
00072         } // matchWithoutCrossProduct
```

6.73.2.7 newMatcher()

```
Matcher * mrmatch::Matcher::newMatcher (
            const Options & opts,
            const minerule::ParsedMinerule & p )  [static]
```

Definition at line 25 of file Matcher.cpp.

```
00025                                                                                    {
00026                 MatcherSpecs specs = opts.matcherSpecs();
00027                 if((specs & MatchOutputMask) == OutOnDB) {
00028                         return new RuleGidsDBMatcher(opts.getMatchOutputTableName(), minerule);
00029                 }
00030
00031                 switch(specs & MatchKindMask) {
00032                         case RuleGids: return new RuleGidsMatcher();
00033                         case GidRules: return new GidRulesMatcher();
00034                         default: throw MineruleException( MR_ERROR_INTERNAL, "Unknown or unsupported
      matcher kind");
00035                 }
00036         }
```

6.73.2.8 printMatches()

```
virtual void mrmatch::Matcher::printMatches ( ) const  [pure virtual]
```

Implemented in mrmatch::GidRulesMatcher, mrmatch::RuleGidsDBMatcher, and mrmatch::RuleGidsMatcher.

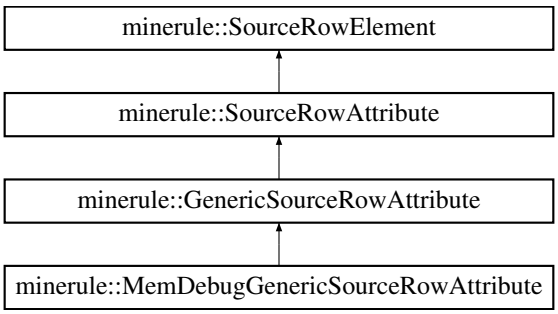
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.hpp
- /Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.cpp

6.74 minerule::MemDebugGenericSourceRowAttribute Class Reference

```
#include <SourceRowAttribute.hpp>
```

Inheritance diagram for minerule::MemDebugGenericSourceRowAttribute:



Public Types

- typedef char ElementType

Public Member Functions

- MemDebugGenericSourceRowAttribute ()
- MemDebugGenericSourceRowAttribute (mrdb::ResultSet *_rs, int _elem, mrdb::Types::SQLType _type)
- MemDebugGenericSourceRowAttribute (const MemDebugGenericSourceRowAttribute &rhs)
- virtual ~MemDebugGenericSourceRowAttribute ()
- virtual SourceRowElement * copy () const
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const
- virtual mrdb::Types::SQLType getType () const
- virtual void setValue (mrdb::ResultSet *, int)
- virtual void setValue (const std::string &)
- virtual int compareTo (const SourceRowAttribute &) const
- virtual std::string asString (const std::string &sep=",") const
- virtual std::string getSQLData () const
- virtual ElementType getElementType () const
- virtual void serialize (std::ostream &os) const
- virtual void deserialize (std::istream &is)
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &s1) const
- virtual bool empty () const

  return true if this attribute is empty
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual std::string getFullElementType () const

Static Public Member Functions

- static size_t getInstanceCounter ()
- static SourceRowAttribute * createAttribute (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, int elem)
- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

6.74.1 Detailed Description

This class keep track of how many instances has been created. It is useful in order to check if a program correctly deallocates sourcerowattribute instances.

Definition at line 209 of file SourceRowAttribute.hpp.

6.74.2 Member Typedef Documentation

6.74.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.74.3 Constructor & Destructor Documentation

6.74.3.1 MemDebugGenericSourceRowAttribute() [1/3]

```
minerule::MemDebugGenericSourceRowAttribute::MemDebugGenericSourceRowAttribute ( )  [inline]
```

Definition at line 222 of file SourceRowAttribute.hpp.

```
00222                                       : GenericSourceRowAttribute() {
00223     incCounter();
00224   }
```

6.74.3.2 MemDebugGenericSourceRowAttribute() [2/3]

```
minerule::MemDebugGenericSourceRowAttribute::MemDebugGenericSourceRowAttribute (
            mrdb::ResultSet * _rs,
            int _elem,
            mrdb::Types::SQLType _type )  [inline]
```

Definition at line 226 of file SourceRowAttribute.hpp.

```
00228       : GenericSourceRowAttribute(_rs, _elem, _type) {
00229     incCounter();
00230   }
```

6.74.3.3 MemDebugGenericSourceRowAttribute() [3/3]

```
minerule::MemDebugGenericSourceRowAttribute::MemDebugGenericSourceRowAttribute (
            const MemDebugGenericSourceRowAttribute & rhs )  [inline]
```

Definition at line 232 of file SourceRowAttribute.hpp.

```
00234       : GenericSourceRowAttribute(rhs) {
00235     incCounter();
00236   }
```

6.74.3.4 ~MemDebugGenericSourceRowAttribute()

```
virtual minerule::MemDebugGenericSourceRowAttribute::~MemDebugGenericSourceRowAttribute ( )
[inline], [virtual]
```

Definition at line 238 of file SourceRowAttribute.hpp.

```
00238 { decCounter(); }
```

6.74.4 Member Function Documentation

6.74.4.1 asString()

```
std::string minerule::GenericSourceRowAttribute::asString (
            const std::string & sep = "," ) const  [virtual], [inherited]
```

Convert this attribute to a string.

Parameters

sep	a separator string. Composite attributes should use this value to separate fields.

Returns

the std::string representation of the value of this attribute.

Implements minerule::SourceRowAttribute.

Definition at line 84 of file SourceRowAttribute.cpp.

```
00084                                                                    {
00085                 if(value == "")
00086                         return "(not set)";
00087                 else
00088                         return value;
00089         }
```

6.74.4.2 compareTo()

```
int minerule::GenericSourceRowAttribute::compareTo (
            const SourceRowAttribute &  ) const  [virtual], [inherited]
```

Compares the present attribute value with the one of the given attribute.

Returns

-1 if the present attribute is less than the argument, 0 if they are equal +1 otherwise

Implements minerule::SourceRowAttribute.

Definition at line 74 of file SourceRowAttribute.cpp.

```
00074                                                                {
00075                 return value.compare(dynamic_cast<const GenericSourceRowAttribute&>(rhs).value);
00076         }
```

6.74.4.3 copy()

```
virtual SourceRowElement * minerule::MemDebugGenericSourceRowAttribute::copy ( ) const  [inline],
[virtual]
```

Returns

a fresh allocated copy of the current object

Reimplemented from minerule::GenericSourceRowAttribute.

Definition at line 240 of file SourceRowAttribute.hpp.

```
00240                                       {
00241     SourceRowAttribute *newRow = new MemDebugGenericSourceRowAttribute(*this);
00242     return newRow;
00243   }
```

6.74.4.4 createAttribute()

```
SourceRowAttribute * minerule::SourceRowAttribute::createAttribute (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            int elem )  [static], [inherited]
```

Factory method.

Returns

an attribute capable of handling types of the given type

Factory method - returns an attribute capable of handling types of the given type

Definition at line 34 of file SourceRowAttribute.cpp.

```
00037                 {
00038                 mrdb::Types::SQLType colType = (mrdb::Types::SQLType) rsmd->getColumnType(elem);
00039 #ifdef DEBUG
00040 #warning MemDebugGeneric...
00041                 return new MemDebugGenericSourceRowAttribute(rs,elem,colType);
00042 #else
00043                 if(colType==mrdb::Types::INTEGER || colType==mrdb::Types::BIGINT)
00044                         return new NumericSourceRowAttribute(rs,elem);
00045                 else
00046                         return new GenericSourceRowAttribute(rs,elem,colType);
00047 #endif
00048         }
```

6.74.4.5 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.74.4.6 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.74.4.7 deserialize()

```
void minerule::GenericSourceRowAttribute::deserialize (
            std::istream & is )  [virtual], [inherited]
```

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.cpp.

```
00097                                                                   {
00098     // In order to avoid missing spaces etc. we resort to use istream::get method
00099     // instead of » operator. This means that we need to read one char at a time
00100     // and implement a simple automata in order to parse the regular expression / @[ .* @] /
00101     // Pro: quite general
00102     // Cons: Nobody ensures that the strings @[ and @] does not appear in the value itself
00103     //       When this happens everything break
00104     // Solution: We should escape any '@' character in thestd::string before inserting it
00105     // into the database. (Cons: "The insertion process becomes more costly...")
00106
00107
00108                 typedef enum {
00109                         AS_NORMAL, // normal automa state
00110                                 AS_EXITING // The state in which the automa will be when ? has been
      read as the last char
00111                                         } AutomaState;
00112
00113                         std::string buf;
00114                         char ch;
00115
00116                         assert(is);
00117                         is » buf;
00118                         assert(is);
00119
00120                         if(buf!="@[")
00121                                 throw MineruleException(MR_ERROR_INTERNAL, "Expected `@[', but:
      `"+buf+"' found!");
00122
00123                         is.get(); // throwing away spurious space.
00124
00125                         bool finished = false;
00126                         value = "";
00127
00128                         AutomaState state = AS_NORMAL;
00129                         while(is.good() && !finished) {
00130                                 ch=is.get();
00131
00132                                 if( ch==']' && state==AS_EXITING )
00133                                         finished=true;
```

```
00134                                   else {
00135                                           switch( ch ) {
00136                                           case '@':
00137                                                   state = AS_EXITING;
00138                                                   break;
00139
00140                                           default:
00141                                                   if(state==AS_EXITING)
00142                                                           value += '@';
00143
00144                                                   value += ch;
00145                                                   break;
00146                                           }
00147                                   }
00148                           }
00149
00150                           if(!is)
00151                                   throw MineruleException(MR_ERROR_INTERNAL,
00152                                           "Unfinished Generic value, value read 'till now:"+value);
00153                   }
```

6.74.4.8 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                   {
00068                   mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                   size_t numColumns = rsmd->getColumnCount();
00070                   std::vector<int> descr;
00071
00072                   for(size_t i=start_index; i<=numColumns; ++i)
00073                           descr.push_back(i);
00074
00075                   SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                   return elem;
00077           }
```

6.74.4.9 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                                   {
00082                   std::istringstream sstream(strRepr);
00083                   assert(sstream);
00084                   ElementType elType;
00085                   sstream>>elType;
00086                   assert(sstream);
00087
00088                   SourceRowElement* elem = createElementFromType(elType);
00089                   elem->deserialize(sstream);
00090                   return elem;
00091           }
```

6.74.4.10 empty()

```
virtual bool minerule::SourceRowAttribute::empty ( ) const  [inline], [virtual], [inherited]
```

return true if this attribute is empty

Implements minerule::SourceRowElement.

Definition at line 113 of file SourceRowAttribute.hpp.

```
00113 { return false; }
```

6.74.4.11 getElementType()

```
virtual ElementType minerule::GenericSourceRowAttribute::getElementType ( ) const  [inline],
[virtual], [inherited]
```

Serialization methods

Reimplemented from minerule::SourceRowElement.

Definition at line 199 of file SourceRowAttribute.hpp.

```
00199 { return 'g'; }
```

6.74.4.12 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                          {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.74.4.13 getInstanceCounter()

```
static size_t minerule::MemDebugGenericSourceRowAttribute::getInstanceCounter ( )  [inline],
[static]
```

Definition at line 245 of file SourceRowAttribute.hpp.

```
00245 { return instanceCount; }
```

6.74.4.14 getSQLData()

```
std::string minerule::GenericSourceRowAttribute::getSQLData ( ) const  [virtual], [inherited]
```

Returns

the std::string representation for the attribute in a format suitable to be used in SQL queries (tipically it is implemented as return "'"+this->asString()+"'"

Implements minerule::SourceRowAttribute.

Definition at line 156 of file SourceRowAttribute.cpp.

```
00156                                                   {
00157                         return std::string("'")+value+"'";
00158                 }
```

6.74.4.15 getType()

```
mrdb::Types::SQLType minerule::GenericSourceRowAttribute::getType ( ) const  [virtual], [inherited]
```

Returns

the type of the attribute

Implements minerule::SourceRowAttribute.

Definition at line 79 of file SourceRowAttribute.cpp.

```
00079                                                   {
00080                 return type;
00081         }
```

6.74.4.16 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                   {
00054     return !this->operator==(el);
00055   }
```

6.74.4.17 operator()()

```
virtual bool minerule::SourceRowAttribute::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [inline], [virtual], [inherited]
```

Returns

true iff s1<s2

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.hpp.

```
00098                                                       {
00099     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00100     const SourceRowAttribute &attr2 = dynamic_cast<const SourceRowAttribute &>(s2);
00101
00102     return attr1.compareTo(attr2) < 0;
00103   }
```

6.74.4.18 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                  {
00058     return this->operator()(*this, el);
00059   }
```

6.74.4.19 operator<<()

```
virtual std::ostream & minerule::SourceRowAttribute::operator<< (
            std::ostream & os ) const  [inline], [virtual], [inherited]
```

Inserts the value of this attribute in the provided ostream and returns the ostream.

Parameters

os	an output stream

Returns

the output stream given in input

Implements minerule::SourceRowElement.

Definition at line 129 of file SourceRowAttribute.hpp.

```
00129                                                  {
00130     os << asString();
00131     return os;
00132   }
```

6.74.4.20 operator==()

```
virtual bool minerule::SourceRowAttribute::operator== (
            const SourceRowElement & s1 ) const  [inline], [virtual], [inherited]
```

Returns

true iff *this == s1

Implements minerule::SourceRowElement.

Definition at line 106 of file SourceRowAttribute.hpp.

```
00106                                                                   {
00107     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00108
00109     return this->compareTo(attr1) == 0;
00110   }
```

6.74.4.21 serialize()

```
void minerule::GenericSourceRowAttribute::serialize (
            std::ostream & os ) const  [virtual], [inherited]
```

Implements minerule::SourceRowElement.

Definition at line 92 of file SourceRowAttribute.cpp.

```
00092                                                                   {
00093                 os « " @[ " « value « "@] ";
00094         }
```

6.74.4.22 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
        {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.74.4.23 setPreparedStatementParameters()

```
virtual void minerule::GenericSourceRowAttribute::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [inline], [virtual], [inherited]
```

Implements minerule::SourceRowElement.

Definition at line 172 of file SourceRowAttribute.hpp.

```
00173                                                                   {
00174     state->setString(start_index, value);
00175   }
```

6.74.4.24 setValue() [1/2]

```
void minerule::GenericSourceRowAttribute::setValue (
            const std::string & value )  [virtual], [inherited]
```

Sets the value of the attribute according to the given string (converting the value to the proper type when necessary).

Parameters

value	a string representation of the value to be set

Implements minerule::SourceRowAttribute.

Definition at line 66 of file SourceRowAttribute.cpp.

```
00066                                                                    {
00067
00068                 type = (mrdb::Types::SQLType) 4;
00069                 value = inStr;
00070         }
```

6.74.4.25 setValue() [2/2]

```
void minerule::GenericSourceRowAttribute::setValue (
            mrdb::ResultSet * rs,
            int col )  [virtual], [inherited]
```

Sets the value of the attribute accordingly to the value of the specified column of the current row of the given result set

Parameters

rs	the result set source of the value
col	the index (starting from 1) of the column in the result set

Implements minerule::SourceRowAttribute.

Definition at line 57 of file SourceRowAttribute.cpp.

```
00057                                                                    {
00058                 assert(rs!=NULL);
00059                 assert(col<=rs->getMetaData()->getColumnCount());
00060
00061                 type = (mrdb::Types::SQLType) rs->getMetaData()->getColumnType(col);
00062                 value = rs->getString(col);
00063         }
```

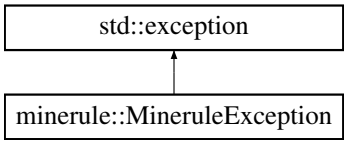
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp

6.75 minerule::MineruleException Class Reference

`#include <MineruleException.hpp>`

Inheritance diagram for minerule::MineruleException:



Public Member Functions

- MineruleException (std::string sourceFile, int sourceLine, size_t errCode, std::string msg)
- virtual ~MineruleException () _NOEXCEPT
- virtual const char * what () const _NOEXCEPT
- virtual const char * unformattedMessage () const _NOEXCEPT
- virtual size_t getErrorCode () const _NOEXCEPT

6.75.1 Detailed Description

Definition at line 30 of file MineruleException.hpp.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 MineruleException()

```
minerule::MineruleException::MineruleException (
            std::string sourceFile,
            int sourceLine,
            size_t errCode,
            std::string msg )
```

Definition at line 25 of file MineruleException.cpp.

```
00025
              : message(msg), file(sourceFile), line(sourceLine), errorCode(errCode) {
00026                             formatMessage();
00027         }
```

6.75.2.2 ~MineruleException()

```
virtual minerule::MineruleException::~MineruleException ( )  [inline], [virtual]
```

Definition at line 42 of file MineruleException.hpp.

```
00042                                       {
00043   }
```

6.75.3 Member Function Documentation

6.75.3.1 getErrorCode()

```
virtual size_t minerule::MineruleException::getErrorCode ( ) const  [inline], [virtual]
```

Definition at line 48 of file MineruleException.hpp.

```
00048                                           {
00049     return errorCode;
00050   }
```

6.75.3.2 unformattedMessage()

```
const char * minerule::MineruleException::unformattedMessage ( ) const  [virtual]
```

Definition at line 52 of file MineruleException.cpp.

```
00052                                                                    {
00053      return message.c_str();
00054         }
```

6.75.3.3 what()

```
const char * minerule::MineruleException::what ( ) const  [virtual]
```

Definition at line 48 of file MineruleException.cpp.

```
00048                                                {
00049                 return formattedMessage.c_str();
00050         }
```

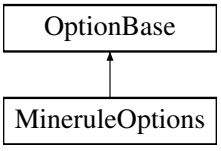
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleException.hpp
- /Users/esposito/Software/minerule/src/Utils/MineruleException.cpp

6.76 MineruleOptions Class Reference

```
#include <root.hpp>
```

Inheritance diagram for MineruleOptions:



Public Member Functions

- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- virtual OptionBase & subclassForName (const std::string &oclass)
- void readFromFile (std::string filename)
- void readFromString (const std::string &)
- void setMineruleSourceName (const std::string &name)
- const std::string & getMineruleSourceName () const
- void setMineruleName (const std::string &name)
- const std::map< std::string, MRLogger * > & getKnownStreams () const
- const std::string & getMineruleName () const
- Mrdb & getMRDB ()
- const Mrdb & getMRDB () const
- const Safety & getSafety () const
- Safety & getSafety ()
- Optimizations & getOptimizations ()
- const Optimizations & getOptimizations () const
- MiningAlgorithms & getMiningAlgorithms ()
- const MiningAlgorithms & getMiningAlgorithms () const
- std::ostream & getLogStream () const
- std::ostream & getErrStream () const
- std::ostream & getWarnStream () const
- std::ostream & getDebugStream () const
- OutStream & getLogStreamObj ()
- const OutStream & getLogStreamObj () const
- OutStream & getErrStreamObj ()
- const OutStream & getErrStreamObj () const
- OutStream & getWarnStreamObj ()
- const OutStream & getWarnStreamObj () const
- OutStream & getDebugStreamObj ()
- const OutStream & getDebugStreamObj () const
- Parsers & getParsers ()
- const Parsers & getParsers () const
- std::ostream & saveOptions (std::ostream &os) const
- std::map< std::string, std::string > & getUserOptions ()

Static Public Member Functions

- static MineruleOptions & getSharedOptions ()

Static Public Attributes

- static const std::string DEFAULT_FILE_NAME

Protected Member Functions

- virtual ~MineruleOptions ()
- MineruleOptions ()
- MineruleOptions (const MineruleOptions &rhs)

6.76.1 Detailed Description

Definition at line 20 of file root.hpp.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 ~MineruleOptions()

```
MineruleOptions::~MineruleOptions ( )  [protected], [virtual]
```

Definition at line 32 of file MineruleOptions.cpp.

```
00032                                   {
00033   std::map<std::string, MRLogger *>::iterator it;
00034   for (it = knownStreams.begin(); it != knownStreams.end(); it++) {
00035     std::ostream *ostr = NULL;
00036
00037     if (it->first != "<stdout>" && it->first != "<stderr>") {
00038       ostr = it->second->getStream();
00039     }
00040     delete it->second;
00041     if (ostr != NULL)
00042       delete ostr;
00043   }
00044 }
```

6.76.2.2 MineruleOptions() [1/2]

```
MineruleOptions::MineruleOptions ( )  [inline], [protected]
```

Definition at line 117 of file root.hpp.

```
00117                     :
00118     logStream(this,"logstream"),
00119     errStream(this,"errstream"),
00120     warnStream(this,"warnstream"),
00121     debugStream(this,"debugstream") {  }
```

6.76.2.3 MineruleOptions() [2/2]

```
MineruleOptions::MineruleOptions (
            const MineruleOptions & rhs )  [inline], [protected]
```

Definition at line 123 of file root.hpp.

```
00123                                       :
00124     mrdb_db(rhs.mrdb_db),
00125     miningAlgorithms(rhs.miningAlgorithms),
00126     logStream(this,"logstream"),
00127     errStream(this,"errstream"),
00128     warnStream(this,"warnstream"),
00129     debugStream(this,"debugstream")  {  }
```

6.76.3 Member Function Documentation

6.76.3.1 className()

```
virtual std::string MineruleOptions::className ( ) const  [inline], [virtual]
```

Definition at line 137 of file root.hpp.

```
00137                                       {
00138     return "root";
00139   }
```

6.76.3.2 getDebugStream()

```
std::ostream & MineruleOptions::getDebugStream ( ) const  [inline]
```

Definition at line 250 of file root.hpp.

```
00250                                 {
00251     return debugStream.getStream() << StringUtils::toOrange("DEBUG:");
00252   }
```

6.76.3.3 getDebugStreamObj() [1/2]

```
OutStream & MineruleOptions::getDebugStreamObj ( )  [inline]
```

Definition at line 285 of file root.hpp.

```
00285                                    {
00286     return debugStream;
00287   }
```

6.76.3.4 getDebugStreamObj() [2/2]

```
const OutStream & MineruleOptions::getDebugStreamObj ( ) const  [inline]
```

Definition at line 290 of file root.hpp.

```
00290                                          {
00291     return debugStream;
00292   }
```

6.76.3.5 getErrStream()

```
std::ostream & MineruleOptions::getErrStream ( ) const  [inline]
```

Definition at line 240 of file root.hpp.

```
00240                               {
00241     return errStream.getStream() << StringUtils::toRed("ERROR:");
00242   }
```

6.76.3.6 getErrStreamObj() [1/2]

```
OutStream & MineruleOptions::getErrStreamObj ( )  [inline]
```

Definition at line 265 of file root.hpp.

```
00265                         {
00266     return errStream;
00267   }
```

6.76.3.7 getErrStreamObj() [2/2]

```
const OutStream & MineruleOptions::getErrStreamObj ( ) const  [inline]
```

Definition at line 270 of file root.hpp.

```
00270                               {
00271     return errStream;
00272   }
```

6.76.3.8 getKnownStreams()

```
const std::map< std::string, MRLogger * > & MineruleOptions::getKnownStreams ( ) const  [inline]
```

Definition at line 184 of file root.hpp.

```
00184                               {
00185     return knownStreams;
00186   }
```

6.76.3.9 getLogStream()

```
std::ostream & MineruleOptions::getLogStream ( ) const  [inline]
```

Definition at line 235 of file root.hpp.

```
00235                               {
00236                 return logStream.getStream() << StringUtils::toGreen("Log:");
00237   }
```

6.76.3.10 getLogStreamObj() [1/2]

```
OutStream & MineruleOptions::getLogStreamObj ( )  [inline]
```

Definition at line 255 of file root.hpp.

```
00255                         {
00256     return logStream;
00257   }
```

6.76.3.11 getLogStreamObj() [2/2]

```
const OutStream & MineruleOptions::getLogStreamObj ( ) const  [inline]
```

Definition at line 260 of file root.hpp.

```
00260                                  {
00261     return logStream;
00262   }
```

6.76.3.12 getMineruleName()

```
const std::string & MineruleOptions::getMineruleName ( ) const  [inline]
```

Definition at line 190 of file root.hpp.

```
00190                                  {
00191     return mineruleName;
00192   }
```

6.76.3.13 getMineruleSourceName()

```
const std::string & MineruleOptions::getMineruleSourceName ( ) const  [inline]
```

Definition at line 173 of file root.hpp.

```
00173                                        {
00174     return mineruleSourceName;
00175   }
```

6.76.3.14 getMiningAlgorithms() [1/2]

```
MiningAlgorithms & MineruleOptions::getMiningAlgorithms ( )  [inline]
```

Definition at line 225 of file root.hpp.

```
00225                                  {
00226     return miningAlgorithms;
00227   }
```

6.76.3.15 getMiningAlgorithms() [2/2]

```
const MiningAlgorithms & MineruleOptions::getMiningAlgorithms ( ) const  [inline]
```

Definition at line 230 of file root.hpp.

```
00230                                        {
00231     return miningAlgorithms;
00232   }
```

6.76.3.16 getMRDB() [1/2]

```
Mrdb & MineruleOptions::getMRDB ( )  [inline]
```

Definition at line 195 of file root.hpp.

```
00195                {
00196     return mrdb_db;
00197   }
```

6.76.3.17 getMRDB() [2/2]

```
const Mrdb & MineruleOptions::getMRDB ( ) const  [inline]
```

Definition at line 200 of file root.hpp.

```
00200                          {
00201     return mrdb_db;
00202   }
```

6.76.3.18 getOptimizations() [1/2]

```
Optimizations & MineruleOptions::getOptimizations ( )  [inline]
```

Definition at line 215 of file root.hpp.

```
00215                            {
00216     return optimizations;
00217   }
```

6.76.3.19 getOptimizations() [2/2]

```
const Optimizations & MineruleOptions::getOptimizations ( ) const  [inline]
```

Definition at line 220 of file root.hpp.

```
00220                                    {
00221     return optimizations;
00222   }
```

6.76.3.20 getParsers() [1/2]

```
Parsers & MineruleOptions::getParsers ( )  [inline]
```

Definition at line 295 of file root.hpp.

```
00295                  {
00296     return parsers;
00297   }
```

6.76.3.21 getParsers() [2/2]

```
const Parsers & MineruleOptions::getParsers ( ) const  [inline]
```

Definition at line 300 of file root.hpp.

```
00300                                 {
00301     return parsers;
00302   }
```

6.76.3.22 getSafety() [1/2]

```
Safety & MineruleOptions::getSafety ( )  [inline]
```

Definition at line 210 of file root.hpp.

```
00210                {
00211     return safety;
00212   }
```

6.76.3.23 getSafety() [2/2]

```
const Safety & MineruleOptions::getSafety ( ) const  [inline]
```

Definition at line 205 of file root.hpp.

```
00205                      {
00206     return safety;
00207   }
```

6.76.3.24 getSharedOptions()

```
static MineruleOptions & MineruleOptions::getSharedOptions ( )  [inline], [static]
```

Definition at line 151 of file root.hpp.

```
00151                                          {
00152     if( !sharedOptions.isReady() )
00153         sharedOptions.init();
00154
00155     return sharedOptions;
00156   }
```

6.76.3.25 getUserOptions()

```
std::map< std::string, std::string > & MineruleOptions::getUserOptions ( )  [inline]
```

Definition at line 307 of file root.hpp.

```
00307                                                {
00308     return userOptions;
00309   }
```

6.76.3.26 getWarnStream()

```
std::ostream & MineruleOptions::getWarnStream ( ) const  [inline]
```

Definition at line 245 of file root.hpp.

```
00245                                {
00246     return warnStream.getStream() << StringUtils::toYellow("WARNING:");
00247   }
```

6.76.3.27 getWarnStreamObj() [1/2]

```
OutStream & MineruleOptions::getWarnStreamObj ( )  [inline]
```

Definition at line 275 of file root.hpp.

```
00275                           {
00276     return warnStream;
00277   }
```

6.76.3.28 getWarnStreamObj() [2/2]

```
const OutStream & MineruleOptions::getWarnStreamObj ( ) const  [inline]
```

Definition at line 280 of file root.hpp.

```
00280                                 {
00281     return warnStream;
00282   }
```

6.76.3.29 readFromFile()

```
void MineruleOptions::readFromFile (
            std::string filename )
```

Definition at line 101 of file MineruleOptions.cpp.

```
00101                                                    {
00102   FILE *file = fopen(filename.c_str(), "r");
00103
00104   try {
00105     if (file == NULL) {
00106       throw MineruleException(MR_ERROR_INPUT_FILE_NOT_FOUND,
00107                               "Cannot open file:" + filename);
00108     }
00109     initializeOptionsFromFile(*this, file);
00110     mrdb_db.resetConnection();
00111   } catch (MineruleException &e) {
00112     if (file != NULL)
00113       fclose(file);
00114
00115     throw;
00116   } catch (mrdb::SQLException &e) {
00117     if (file != NULL)
00118       fclose(file);
00119
00120     throw;
00121   }
00122
00123   fclose(file);
00124 }
```

6.76.3.30 readFromString()

```
void MineruleOptions::readFromString (
            const std::string & str )
```

Definition at line 126 of file MineruleOptions.cpp.

```
00126                                                          {
00127   initializeOptionsFromString(*this, str);
00128
00129   mrdb_db.resetConnection();
00130 }
```

6.76.3.31 saveOptions()

```
std::ostream & MineruleOptions::saveOptions (
            std::ostream & os ) const
```

Definition at line 132 of file MineruleOptions.cpp.

```
00132                                                                {
00133   os « "# Options related to the MRDB connection" « std::endl;
00134   os « "mrdb::{" « std::endl « "  +name=" « getMRDB().getName() « std::endl
00135      « "  +username=" « getMRDB().getUsername() « std::endl
00136      « "  +password=" « getMRDB().getPassword() « std::endl
00137      « "  +cacheWrites=" « Converter(getMRDB().getCacheWrites()).toString()
00138      « std::endl « "# dbms allows one to specify the underlying dbms, "
00139                  "supported dbms are presently" « std::endl
00140      « "# mysql and postgres" « std::endl « "  +dbms=" « getMRDB().getDBMS()
00141      « std::endl « "}" « std::endl « std::endl;
00142
00143   os « "# Options related to data safety issues" « std::endl;
00144   os « "safety::{" « std::endl
00145      « "# if the following option is set to 'True', then the" « std::endl
00146      « "# system will delete old results whenever a new minerule" « std::endl
00147      « "# having the same name of an old one is inserted. Otherwise"
00148      « std::endl « "# the system will report an error message and exit."
00149      « std::endl « "  +overwriteHomonymMinerules="
00150      « Converter(getSafety().getOverwriteHomonymMinerules()).toString()
00151      « std::endl « "# if overwriteHomonymMinerules is set to True, then the"
00152      « std::endl
00153      « "# following option decides whether the system should delete"
00154      « std::endl « "# also the minerules for which the result depends on the "
00155      « std::endl « "# deleted one. If the option is set to True, those "
00156      « std::endl « "# minerule will be deleted as well, otherwise the system "
00157      « std::endl « "# with halt reporting an error." « std::endl
00158      « "  +allowCascadeDeletes="
00159      « Converter(getSafety().getAllowCascadeDeletes()).toString() « std::endl
00160      « "}" « std::endl « std::endl;
00161
00162   os « "# Options related to mining algorithms" « std::endl;
00163   os « "miningalgorithms::{" « std::endl;
00164   os « "  #options for configuring rule mining algorithms" « std::endl;
00165   os « "  rulesmining::{" « std::endl;
00166   os « "    +preferredAlgorithm="
00167      « algorithmTypeToString(getMiningAlgorithms()
00168                              .getRulesMiningAlgorithms()
00169                              .getPreferredAlgorithm()) « std::endl
00170      « std::endl;
00171   os « "    # Options related to PartitionBase algorithm" « std::endl;
00172   os « "    partitionbase::{" « std::endl « "      +rowsPerPartition="
00173      « getMiningAlgorithms()
00174            .getRulesMiningAlgorithms()
00175            .getPartitionBase()
00176            .getRowsPerPartition() « std::endl « "    }" « std::endl
00177      « std::endl;
00178
00179   os « "    # Options related to PartitionWithClusters algorithm" « std::endl;
00180   os « "    partitionwithclusters::{" « std::endl
00181      « "      +rowsPerPartition="
00182      « getMiningAlgorithms()
00183            .getRulesMiningAlgorithms()
00184            .getPartitionWithClusters()
00185            .getRowsPerPartition() « std::endl « "    }" « std::endl
00186      « std::endl;
00187
00188   std::string algoType;
```

```
00189   if (getMiningAlgorithms()
00190           .getRulesMiningAlgorithms()
00191           .getFPGrowth()
00192           .getAlgoType() ==
00193       MiningAlgorithms::RulesMiningAlgorithms::FPGrowth::Original)
00194     algoType = "Original";
00195   else
00196     algoType = "SingleReorder";
00197
00198   os « "    # Options related to FPGrowth algorithms" « std::endl;
00199   os « "    fpgrowth::{" « std::endl « "      +algoType=" « algoType
00200      « std::endl « "    }" « std::endl;
00201   os « "  }" « std::endl;
00202
00203   os « " itemsetsmining::{" « std::endl;
00204   os « "    +preferredAlgorithm="
00205      « algorithmTypeToString(getMiningAlgorithms()
00206                                  .getItemsetsMiningAlgorithms()
00207                                  .getPreferredAlgorithm()) « std::endl
00208      « std::endl;
00209   os « " }" « std::endl;
00210   os « "}" « std::endl « std::endl;
00211
00212   std::string optimizations;
00213   if (getOptimizations().getTryOptimizations())
00214     optimizations = "True";
00215   else
00216     optimizations = "False";
00217   std::string incrAlgorithm;
00218   switch (getOptimizations().getIncrementalAlgorithm()) {
00219   case Optimizations::ConstructiveAlgo:
00220     incrAlgorithm = "constructive";
00221     break;
00222   case Optimizations::DestructiveAlgo:
00223     incrAlgorithm = "destructive";
00224     break;
00225   case Optimizations::AutochooseIncrAlgo:
00226     incrAlgorithm = "Auto";
00227     break;
00228   }
00229
00230   os « "# Options related to Optimizations" « std::endl;
00231   os « "optimizations::{" « std::endl
00232      « "  +enableOptimizations=" « optimizations « std::endl
00233      « "# If set to True, this option will disable the detection of dominant"
00234      « std::endl « "# queries (this imply also that the system will not try "
00235                    "to find equivalent" « std::endl
00236      « "# queries, since they are a particular case of dominance)" « std::endl
00237      « "  +avoidDominanceDetection="
00238      « Converter(getOptimizations().getAvoidDominanceDetection()).toString()
00239      « std::endl « "# If set to True this option will make the optimizer to "
00240      « std::endl
00241      « "# consider equivalent queries as if they were dominant ones"
00242      « std::endl « "# (i.e., it will call an incremental algorithm instead of"
00243      « std::endl « "# dealing with the equivalence)." « std::endl
00244      « "  +avoidEquivalenceDetection="
00245      « Converter(getOptimizations().getAvoidEquivalenceDetection()).toString()
00246      « std::endl « "# If set to True the optimizer will not try to find "
00247      « std::endl
00248      « "# a combinations of previous queries equivalent to the current one."
00249      « std::endl
00250      « "# Notice that the search for combination may be a slow process"
00251      « std::endl « "  +avoidCombinationDetection="
00252      « Converter(getOptimizations().getAvoidCombinationDetection()).toString()
00253      « std::endl « "# The following option allows the user to specify how a "
00254      « std::endl
00255      « "# particular incremental algorithm  have to be chosen. The"
00256      « std::endl « "# following values are allowed:{Constructive,Destructive,"
00257      « std::endl « "# Auto}" « std::endl
00258      « "# Constructive and Destructive force the corresponding " « std::endl
00259      « "# algorithm to be chosen. " « std::endl
00260      « "# Auto leaves the choice to the optimizer." « std::endl
00261      « "  +incrementalAlgorithm=" « incrAlgorithm « std::endl
00262      « "# Options related to the query combinator algorithm" « std::endl
00263      « "  combinator::{" « std::endl
00264      « "# amount of time the search for a combination is allowed to run "
00265      « std::endl « "    +timeOut="
00266      « Converter(getOptimizations().getCombinator().getTimeOutThreshold())
00267            .toString() « std::endl « "# Max number of disjuncts. It is the "
00268                                      "number of disjuncts that is considered"
00269      « std::endl « "# during the search. Notice that increasing this number "
00270                    "has a strong impact" « std::endl
00271      « "# on the dimension of the search space." « std::endl
00272      « "    +maxDisjuncts="
00273      « Converter(long(getOptimizations().getCombinator().getMaxDisjuncts()))
00274            .toString() « std::endl « "# Max number of queries. Max number "
00275                                      "of distinct queries the user allows to"
```

```
00276       « std::endl « "# be combined in the result. Formulae with a larger "
00277                      "number of queries are" « std::endl
00278       « "# penalized in the evaluation function." « std::endl
00279       « "    +maxQueries="
00280       « Converter(long(getOptimizations().getCombinator().getMaxQueries()))
00281             .toString() « std::endl « "# Max distinct predicates. Max number "
00282                                       "of distinct predicates that the user"
00283       « std::endl « "# allows. This afflict the response time: the time spent "
00284                     "in assessing each" « std::endl
00285       « "# formula grows exponentially fast with the number of predicates."
00286       « std::endl « "    +maxDistinctPredicates="
00287       « Converter(
00288             long(getOptimizations().getCombinator().getMaxDistinctPredicates()))
00289             .toString() « std::endl « "  }" « std::endl « "}" « std::endl
00290       « std::endl;
00291    os « "# Options related to the parsing algorithms" « std::endl;
00292    os « "parsers::{" « std::endl;
00293    os « "# Parsers log stream, valid names are:" « std::endl;
00294    os « "#   <stdout>, <stderr> and any writeable file." « std::endl;
00295    os « " +logfile=/dev/null" « std::endl;
00296    os « "# The following four options allows to set constraint on" « std::endl
00297       « "# cardinalities of elements which appears in the body/head"
00298       « std::endl
00299       « "# part of rules. The constraints set here 'win' on the ones"
00300       « std::endl
00301       « "# in minerules (i.e., if you say '1..n' as BODY in your minerule"
00302       « std::endl
00303       « "# but set it to 1..5 here, than 1..5 will be used instead."
00304       « std::endl;
00305    os « " +minBodyElems=" « getParsers().getBodyCardinalities().getMin()
00306       « std::endl;
00307    os « " +maxBodyElems=" « getParsers().getBodyCardinalities().getMax()
00308       « std::endl;
00309    os « " +minHeadElems=" « getParsers().getHeadCardinalities().getMin()
00310       « std::endl;
00311    os « " +maxHeadElems=" « getParsers().getHeadCardinalities().getMax()
00312       « std::endl;
00313    os « "}" « std::endl « std::endl;
00314
00315    os « "# Options related to streams, note that they are commented."
00316       « std::endl
00317       « "# the reason is that the following conresponds to default "
00318       « std::endl « "# settings instead of the actual ones." « std::endl
00319       « "# In specifying the stream parameter, you can:" « std::endl
00320       « "# 1) Specify a file path" « std::endl
00321       « "# 2) Specify <stdout>,<stderr> in order to specify the standard"
00322       « std::endl « "#    output and the standard error respectively"
00323       « std::endl « "# 3) Specify a file path including the %m and %i symbols"
00324       « std::endl
00325       « "# In case 3) %m is expandend to the current minerule name as"
00326       « std::endl
00327       « "# it appear in the minerule text, %i is expanded to the value"
00328       « std::endl « "# of the -i parameter if any, to 'mr' otherwise"
00329       « std::endl
00330       « "# The loglevel option allows you to select how deep the log nesting"
00331       « std::endl « "# can grow. The default (loglevel==100) means: \"grow as "
00332                     "much as needed\"" « std::endl
00333       « "# (no function in minerule will ever nest logs more than a few "
00334          "levels)." « std::endl « "# If set to 0, then the log is actually "
00335                                   "suppressed (the first level is 1)."
00336       « std::endl
00337       « "# Otherwise, if set to n, only the first n levels will be displayed."
00338       « std::endl « "#" « std::endl « "# logstream::{" « std::endl
00339       « "#    +stream=<stdout> " « std::endl « "#    +loglevel=100"
00340       « std::endl « "# }" « std::endl « "# errstream::{" « std::endl
00341       « "#    +stream=<stderr> " « std::endl « "#    +loglevel=100"
00342       « std::endl « "# }" « std::endl « "# warnstream::{" « std::endl
00343       « "#    +stream=<stdout> " « std::endl « "#    +loglevel=100"
00344       « std::endl « "# }" « std::endl « "# debugstream::{" « std::endl
00345       « "#    +stream=<stderr> " « std::endl « "#    +loglevel=100"
00346       « std::endl « "# }" « std::endl;
00347
00348    return os;
00349 }
```

6.76.3.32 setMineruleName()

```
void MineruleOptions::setMineruleName (
            const std::string & name )  [inline]
```

Definition at line 179 of file root.hpp.

```
00179                                                {
00180     mineruleName = name;
00181   }
```

6.76.3.33 setMineruleSourceName()

```
void MineruleOptions::setMineruleSourceName (
            const std::string & name )  [inline]
```

Definition at line 167 of file root.hpp.

```
00167                                                    {
00168     mineruleSourceName=name;
00169   }
```

6.76.3.34 setOption()

```
virtual void MineruleOptions::setOption (
            const std::string & name,
            const std::string & value )  [inline], [virtual]
```

Definition at line 141 of file root.hpp.

```
00142     {
00143     throw MineruleException(MR_ERROR_OPTION_PARSING,
00144                             "Attempting to set an option in the root class");
00145   }
```

6.76.3.35 subclassForName()

```
OptionBase & MineruleOptions::subclassForName (
            const std::string & oclass )  [virtual]
```

Definition at line 46 of file MineruleOptions.cpp.

```
00046                                                                {
00047   if (oclass == "mrdb")
00048     return getMRDB();
00049   else if (oclass == "safety")
00050     return getSafety();
00051   else if (oclass == "miningalgorithms")
00052     return getMiningAlgorithms();
00053   else if (oclass == "optimizations")
00054     return getOptimizations();
00055   else if (oclass == "logstream")
00056     return getLogStreamObj();
00057   else if (oclass == "errstream")
00058     return getErrStreamObj();
00059   else if (oclass == "warnstream")
00060     return getWarnStreamObj();
00061   else if (oclass == "debugstream")
00062     return getDebugStreamObj();
00063   else if (oclass == "parsers")
00064     return getParsers();
00065   else
00066     return OptionBase::subclassForName(oclass);
00067 }
```

6.76.4 Field Documentation

6.76.4.1 DEFAULT_FILE_NAME

```
const std::string MineruleOptions::DEFAULT_FILE_NAME  [static]
```

Definition at line 22 of file root.hpp.

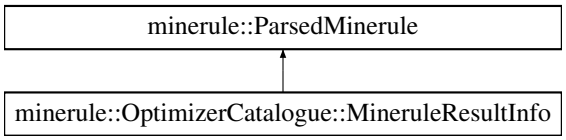
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/root.hpp
- /Users/esposito/Software/minerule/src/Utils/MineruleOptions.cpp

6.77 minerule::OptimizerCatalogue::MineruleResultInfo Class Reference

```
#include <OptimizerCatalogue.hpp>
```

Inheritance diagram for minerule::OptimizerCatalogue::MineruleResultInfo:



Public Types

- typedef std::vector< std::string > AttrVector
- typedef Bem_cond * bem_c

Public Member Functions

- MineruleResultInfo (const ParsedMinerule &mr)
- MineruleResultInfo (const MineruleResultInfo &mri)
- void init (const std::string &minerule_text)
- bool requiresClusters () const
- bool hasCrossConditions (const list_OR_node *cond) const
- bool hasCrossConditions () const
- bool hasDisjuctionsInMC () const
- bool hasSameBodyHead () const
- std::string getMinesequenceText () const
- std::string getMineruleText () const
- std::string getMineitemsetsText () const
- std::string getText () const
- std::string getBEMText () const

Data Fields

- std::string resultset
- AttrVector ga
- AttrVector oa
- AttrVector ca
- AttrVector ra
- AttrVector ba
- AttrVector ha
- list_OR_node * mc
- list_OR_node * gc
- list_OR_node * cc
- AttrVector c_aggr_list
- float sup
- float conf
- MinMaxPair bodyCardinalities
- MinMaxPair headCardinalities
- std::string tab_source
- std::string tab_result
- bool tautologies
- bool body_coincident_head
- bool distinct
- MinMaxPair sequenceAllowedGaps
- MinMaxPair length
- std::vector< Dist_cond * > seq_dist_vect
- std::vector< Bem_cond * > seq_bem_vect
- std::string filter_condition
- MiningTasks miningTask

6.77.1 Detailed Description

Definition at line 153 of file OptimizerCatalogue.hpp.

6.77.2 Member Typedef Documentation

6.77.2.1 AttrVector

```
typedef std::vector<std::string> minerule::ParsedMinerule::AttrVector  [inherited]
```

Definition at line 139 of file ParsedMinerule.hpp.

6.77.2.2 bem_c

```
typedef Bem_cond* minerule::ParsedMinerule::bem_c  [inherited]
```

Definition at line 140 of file ParsedMinerule.hpp.

6.77.3 Constructor & Destructor Documentation

6.77.3.1 MineruleResultInfo() [1/2]

```
minerule::OptimizerCatalogue::MineruleResultInfo::MineruleResultInfo (
            const ParsedMinerule & mr )  [inline]
```

Definition at line 157 of file OptimizerCatalogue.hpp.

```
00157 : ParsedMinerule(mr), resultset(mr.tab_result) {};
```

6.77.3.2 MineruleResultInfo() [2/2]

```
minerule::OptimizerCatalogue::MineruleResultInfo::MineruleResultInfo (
            const MineruleResultInfo & mri )  [inline]
```

Definition at line 158 of file OptimizerCatalogue.hpp.

```
00158 : ParsedMinerule(mri), resultset(mri.resultset) {};
```

6.77.4 Member Function Documentation

6.77.4.1 getBEMText()

```
std::string minerule::ParsedMinerule::getBEMText ( ) const  [inherited]
```

Definition at line 394 of file ParsedMinerule.cpp.

```
00394                                                {
00395           std::string ris = "";
00396
00397           for(int i = 0 ; i < seq_bem_vect.size(); ++i)
00398               ris += seq_bem_vect[i]->type +"."+seq_bem_vect[i]->attr+" "+seq_bem_vect[i]->op+"
      "+seq_bem_vect[i]->val+" ";
00399
00400           return ris;
00401       }
```

6.77.4.2 getMineitemsetsText()

```
std::string minerule::ParsedMinerule::getMineitemsetsText ( ) const  [inherited]
```

Definition at line 370 of file ParsedMinerule.cpp.

```
00370                                                                   {
00371                         std::string result;
00372                         result =
00373                                 "MINE ITEMSET " + tab_result + " AS " +
00374                                         "SELECT DISTINCT " +
00375                                                 getCardsText(bodyCardinalities) + " " +
      getAttrText(ba) + " AS BODY" +
00376                                                         ", SUPPORT ";
00377
00378                         if( mc!=NULL )
00379                                 result += "WHERE "+getCondText(mc) + " ";
00380
00381                         result +=
00382                                 "FROM "+tab_source + " "
00383                                         "GROUP BY " + getAttrText(ga) + " ";
00384
00385                         if( gc!=NULL )
00386                                 result += getCondText(gc) + " ";
00387
00388                         result +=
00389                                 "EXTRACTING ITEMSETS WITH SUPPORT:" + Converter(sup).toString();
00390
00391                         return result;
00392                 }
```

6.77.4.3 getMineruleText()

```
std::string minerule::ParsedMinerule::getMineruleText ( ) const  [inherited]
```

Definition at line 338 of file ParsedMinerule.cpp.

```
00338                                                                   {
00339                         std::string result;
00340                         result =
00341                                 "MINE RULE " + tab_result + " AS " +
00342                                 "SELECT DISTINCT " +
00343                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) + " AS BODY,"
    +
00344                                 getCardsText(headCardinalities) + " " + getAttrText(ha) + " AS HEAD"
00345                                 ", SUPPORT, CONFIDENCE ";
00346
00347                         if( mc!=NULL )
00348                                 result += "WHERE "+getCondText(mc) + " ";
00349
00350                         result +=
00351                                 "FROM "+tab_source + " "
00352                                         "GROUP BY " + getAttrText(ga) + " ";
00353
00354                         if( gc!=NULL )
00355                                 result += getCondText(gc) + " ";
00356
00357                         if( !ca.empty() ) {
00358                                 result += "CLUSTER BY " + getAttrText(ca) + " ";
00359
00360                                 if( cc!=NULL )
00361                                         result+= "HAVING " +getCondText(cc) + " ";
00362                         }
00363
00364                         result +=
00365                                 "EXTRACTING RULES WITH SUPPORT:" + Converter(sup).toString() + ","+
00366                                         "CONFIDENCE:"+Converter(conf).toString();
00367
00368                         return result;
00369                 }
```

6.77.4.4 getMinesequenceText()

```
std::string minerule::ParsedMinerule::getMinesequenceText ( ) const  [inherited]
```

Definition at line 308 of file ParsedMinerule.cpp.

```
00308                                                                   {
00309                         std::string result;
00310             std::string distinct= this->distinct? " DISTINCT " : " " ;
00311             std::string Bem_text= getBEMText();
00312             if(Bem_text.size()>0)
00313                 Bem_text= " HAVING " + Bem_text;
00314                         result =
00315                                 "MINE SEQUENCE " + tab_result + " AS " +
00316                 "SELECT " + distinct +
00317                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) +
00318                     ", SUPPORT FROM "+tab_source+" "+Bem_text+" GROUP BY "+getAttrText(ga)+
00319                                 " ORDER BY "+getAttrText(oa)+ " EXTRACTING SEQUENCES WITH SUPPORT:" +
00320                                 Converter(sup).toString();
00321
00322                         return result;
00323                 }
```

6.77.4.5 getText()

```
std::string minerule::ParsedMinerule::getText ( ) const  [inherited]
```

Definition at line 324 of file ParsedMinerule.cpp.

```
00324                                                   {
00325                         switch( miningTask ) {
00326                                 case MTMineRules:
00327                                 return getMineruleText();
00328                                 case MTMineSequences:
00329                                 return getMinesequenceText();
00330                                 case MTMineItemsets:
00331                                 return getMineitemsetsText();
00332                                 default:
00333                                 throw MineruleException( MR_ERROR_MINERULETEXT_PARSING,
00334                                         "Current query is not currently supported by the system "
00335                                                 "(i.e., it is not a MINE RULE nor a MINE SEQUENCE
      query" );
00336                         }
00337                 }
```

6.77.4.6 hasCrossConditions() [1/2]

```
bool minerule::ParsedMinerule::hasCrossConditions ( ) const  [inline], [inherited]
```

Definition at line 264 of file ParsedMinerule.hpp.

```
00264                                   {
00265     return hasCrossConditions(mc) || hasCrossConditions(gc) || hasCrossConditions(cc);
00266   }
```

6.77.4.7 hasCrossConditions() [2/2]

```
bool minerule::ParsedMinerule::hasCrossConditions (
            const list_OR_node * cond ) const  [inherited]
```

Definition at line 403 of file ParsedMinerule.cpp.

```
00403                                                                                {
00404                         MRLog()<<"Checking whether the minerule contains cross predicates" << std::endl;
00405
00406                         const list_OR_node* curr_OR;
00407                         for(curr_OR=cond; curr_OR!=NULL; curr_OR=curr_OR->next) {
00408                                 list_AND_node* curr_AND;
00409                                 for(curr_AND=curr_OR->l_and; curr_AND!=NULL; curr_AND=curr_AND->next)
      {
00410                                         assert(curr_AND->sp!=NULL);
00411                                         MRLog() << curr_AND->sp->val1 << " " << curr_AND->sp->val2 <<
      std::endl;
00412                                         bool hasHead =
00413                                                 strstr(curr_AND->sp->val1,
      "HEAD.")==curr_AND->sp->val1 ||
00414                                                         strstr(curr_AND->sp->val2,
      "HEAD.")==curr_AND->sp->val2;
00415                                         bool hasBody =
00416                                                 strstr(curr_AND->sp->val1,
      "BODY.")==curr_AND->sp->val1 ||
00417                                                         strstr(curr_AND->sp->val2,
      "BODY.")==curr_AND->sp->val2;
00418                                         if( hasHead && hasBody ) {
00419                                                 MRLog() << "Cross predicate found!" << std::endl;
00420                                                 return true;
00421                                         }
00422                                 }
00423                         }
00424
00425                         MRLog() << "Cross predicate not found!" << std::endl;
00426                         return false;
00427                 }
```

6.77.4.8 hasDisjuctionsInMC()

```
bool minerule::ParsedMinerule::hasDisjuctionsInMC ( ) const  [inline], [inherited]
```

Definition at line 268 of file ParsedMinerule.hpp.

```
00268                                   {
00269     return mc!=NULL && mc->next!=NULL;
00270   }
```

6.77.4.9 hasSameBodyHead()

```
bool minerule::ParsedMinerule::hasSameBodyHead ( ) const  [inline], [inherited]
```

Definition at line 272 of file ParsedMinerule.hpp.

```
00272                                  {
00273           return ba == ha;
00274   }
```

6.77.4.10 init()

```
void minerule::ParsedMinerule::init (
            const std::string & minerule_text )  [inherited]
```

Definition at line 239 of file ParsedMinerule.cpp.

```
00239                                                                    {
00240                       parseMinerule( minerule_text, *this );
00241                }
```

6.77.4.11 requiresClusters()

```
bool minerule::ParsedMinerule::requiresClusters ( ) const  [inline], [inherited]
```

Definition at line 257 of file ParsedMinerule.hpp.

```
00257                                {
00258     return !ca.empty();
00259     //  return !(body_coincident_head && ca.empty());
00260   }
```

6.77.5 Field Documentation

6.77.5.1 ba

```
AttrVector minerule::ParsedMinerule::ba  [inherited]
```

Definition at line 158 of file ParsedMinerule.hpp.

6.77.5.2 body_coincident_head

```
bool minerule::ParsedMinerule::body_coincident_head  [inherited]
```

Definition at line 177 of file ParsedMinerule.hpp.

6.77.5.3 bodyCardinalities

```
MinMaxPair minerule::ParsedMinerule::bodyCardinalities  [inherited]
```

Definition at line 171 of file ParsedMinerule.hpp.

6.77.5.4 c_aggr_list

```
AttrVector minerule::ParsedMinerule::c_aggr_list  [inherited]
```

Definition at line 167 of file ParsedMinerule.hpp.

6.77.5.5 ca

```
AttrVector minerule::ParsedMinerule::ca  [inherited]
```

Definition at line 155 of file ParsedMinerule.hpp.

6.77.5.6 cc

```
list_OR_node* minerule::ParsedMinerule::cc  [inherited]
```

Definition at line 165 of file ParsedMinerule.hpp.

6.77.5.7 conf

```
float minerule::ParsedMinerule::conf  [inherited]
```

Definition at line 170 of file ParsedMinerule.hpp.

6.77.5.8 distinct

```
bool minerule::ParsedMinerule::distinct  [inherited]
```

Definition at line 178 of file ParsedMinerule.hpp.

6.77.5.9 filter_condition

```
std::string minerule::ParsedMinerule::filter_condition  [inherited]
```

Definition at line 185 of file ParsedMinerule.hpp.

6.77.5.10 ga

```
AttrVector minerule::ParsedMinerule::ga  [inherited]
```

Definition at line 153 of file ParsedMinerule.hpp.

6.77.5.11 gc

```
list_OR_node* minerule::ParsedMinerule::gc  [inherited]
```

Definition at line 164 of file ParsedMinerule.hpp.

6.77.5.12 ha

```
AttrVector minerule::ParsedMinerule::ha  [inherited]
```

Definition at line 159 of file ParsedMinerule.hpp.

6.77.5.13 headCardinalities

```
MinMaxPair minerule::ParsedMinerule::headCardinalities  [inherited]
```

Definition at line 172 of file ParsedMinerule.hpp.

6.77.5.14 length

```
MinMaxPair minerule::ParsedMinerule::length  [inherited]
```

Definition at line 182 of file ParsedMinerule.hpp.

6.77.5.15 mc

```
list_OR_node* minerule::ParsedMinerule::mc  [inherited]
```

Definition at line 163 of file ParsedMinerule.hpp.

6.77.5.16 miningTask

```
MiningTasks minerule::ParsedMinerule::miningTask  [inherited]
```

Definition at line 188 of file ParsedMinerule.hpp.

6.77.5.17 oa

```
AttrVector minerule::ParsedMinerule::oa  [inherited]
```

Definition at line 154 of file ParsedMinerule.hpp.

6.77.5.18 ra

```
AttrVector minerule::ParsedMinerule::ra  [inherited]
```

Definition at line 156 of file ParsedMinerule.hpp.

6.77.5.19 resultset

```
std::string minerule::OptimizerCatalogue::MineruleResultInfo::resultset
```

Definition at line 155 of file OptimizerCatalogue.hpp.

6.77.5.20 seq_bem_vect

```
std::vector<Bem_cond*> minerule::ParsedMinerule::seq_bem_vect  [inherited]
```

Definition at line 184 of file ParsedMinerule.hpp.

6.77.5.21 seq_dist_vect

```
std::vector<Dist_cond*> minerule::ParsedMinerule::seq_dist_vect  [inherited]
```

Definition at line 183 of file ParsedMinerule.hpp.

6.77.5.22 sequenceAllowedGaps

```
MinMaxPair minerule::ParsedMinerule::sequenceAllowedGaps  [inherited]
```

Definition at line 181 of file ParsedMinerule.hpp.

6.77.5.23 sup

```
float minerule::ParsedMinerule::sup  [inherited]
```

Definition at line 169 of file ParsedMinerule.hpp.

6.77.5.24 tab_result

```
std::string minerule::ParsedMinerule::tab_result  [inherited]
```

Definition at line 174 of file ParsedMinerule.hpp.

6.77.5.25 tab_source

```
std::string minerule::ParsedMinerule::tab_source  [inherited]
```

Definition at line 173 of file ParsedMinerule.hpp.

6.77.5.26 tautologies

```
bool minerule::ParsedMinerule::tautologies  [inherited]
```

Definition at line 176 of file ParsedMinerule.hpp.

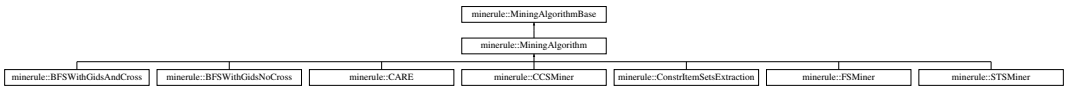
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp

6.78 minerule::MiningAlgorithm Class Reference

```
#include <MiningAlgorithmBase.hpp>
```

Inheritance diagram for minerule::MiningAlgorithm:



Public Member Functions

- MiningAlgorithm (const OptimizedMinerule &m)
- virtual void initialize ()
- virtual void execute ()
- virtual void mineRules ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual bool canHandleMinerule () const
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- AlgorithmsOptions options
- Connection connection
- const OptimizedMinerule & minerule

6.78.1 Detailed Description

Definition at line 59 of file MiningAlgorithmBase.hpp.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 MiningAlgorithm()

```
minerule::MiningAlgorithm::MiningAlgorithm (
            const OptimizedMinerule & m )  [inline]
```

Definition at line 64 of file MiningAlgorithmBase.hpp.

```
00064 : MiningAlgorithmBase(m) {}
```

6.78.3 Member Function Documentation

6.78.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.78.3.2 canHandleMinerule()

```
virtual bool minerule::MiningAlgorithmBase::canHandleMinerule ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CARE, minerule::CCSMiner, minerule::ConstrItemSetsExtraction, minerule::FSMiner, and minerule::STSMiner.

Definition at line 48 of file MiningAlgorithmBase.hpp.

```
00048                                                {
00049                         return false;
00050                 }
```

6.78.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                       {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.78.3.4 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                       {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068 
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073 
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077 
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081 
00082 
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084 
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085 
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087 
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.78.3.5 mineRules()

```
virtual void minerule::MiningAlgorithm::mineRules ( )  [inline], [virtual]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CARE, minerule::CCSMiner, minerule::ConstrItemSetsExtraction, minerule::FSMiner, and minerule::STSMiner.

Definition at line 98 of file MiningAlgorithmBase.hpp.

```
00098                                       {
00099                                 throw MineruleException(MR_ERROR_INTERNAL, "This method should be
      implemented in sub-classes!");
00100                 }
```

6.78.3.6 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.78.3.7 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::MiningAlgorithmBase::sourceTableRequirements ( )
const  [inline], [virtual], [inherited]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CCSMiner, minerule::FSMiner, and minerule::STSMiner.

Definition at line 43 of file MiningAlgorithmBase.hpp.

```
00043                                                                    {
00044                         return SourceTableRequirements();
00045                 };
```

6.78.4 Field Documentation

6.78.4.1 connection

```
Connection minerule::MiningAlgorithm::connection  [protected]
```

Definition at line 62 of file MiningAlgorithmBase.hpp.

6.78.4.2 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

6.78.4.3 options

```
AlgorithmsOptions minerule::MiningAlgorithm::options  [protected]
```

Definition at line 61 of file MiningAlgorithmBase.hpp.

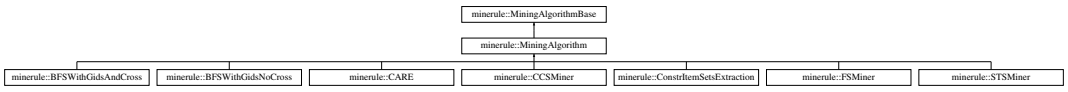
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/MiningAlgorithmBase.hpp

6.79 minerule::MiningAlgorithmBase Class Reference

```
#include <MiningAlgorithmBase.hpp>
```

Inheritance diagram for minerule::MiningAlgorithmBase:



Public Member Functions

- MiningAlgorithmBase (const OptimizedMinerule &mr)
- virtual ~MiningAlgorithmBase ()
- virtual void execute ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual bool canHandleMinerule () const
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- const OptimizedMinerule & minerule

6.79.1 Detailed Description

Base class from which each new mining algorithm will be derived

Definition at line 32 of file MiningAlgorithmBase.hpp.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 MiningAlgorithmBase()

```
minerule::MiningAlgorithmBase::MiningAlgorithmBase (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 36 of file MiningAlgorithmBase.hpp.

```
00036 : minerule(mr) {}
```

6.79.2.2 ~MiningAlgorithmBase()

```
virtual minerule::MiningAlgorithmBase::~MiningAlgorithmBase ( )  [inline], [virtual]
```

Definition at line 37 of file MiningAlgorithmBase.hpp.

```
00037 {}
```

6.79.3 Member Function Documentation

6.79.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.79.3.2 canHandleMinerule()

```
virtual bool minerule::MiningAlgorithmBase::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CARE, minerule::CCSMiner, minerule::ConstrItemSetsExtraction, minerule::FSMiner, and minerule::STSMiner.

Definition at line 48 of file MiningAlgorithmBase.hpp.

```
00048                                                {
00049                         return false;
00050                 }
```

6.79.3.3 execute()

```
virtual void minerule::MiningAlgorithmBase::execute ( )  [inline], [virtual]
```

Reimplemented in minerule::MiningAlgorithm.

Definition at line 39 of file MiningAlgorithmBase.hpp.

```
00039                                        {
00040                         throw MineruleException( MR_ERROR_INTERNAL, "This method should never be
      executed!");
00041                 }
```

6.79.3.4 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.79.3.5 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::MiningAlgorithmBase::sourceTableRequirements ( )
const  [inline], [virtual]
```

Reimplemented in minerule::BFSWithGidsAndCross, minerule::BFSWithGidsNoCross, minerule::CCSMiner, minerule::FSMiner, and minerule::STSMiner.

Definition at line 43 of file MiningAlgorithmBase.hpp.

```
00043                                                                    {
00044                         return SourceTableRequirements();
00045                 };
```

6.79.4 Field Documentation

6.79.4.1 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

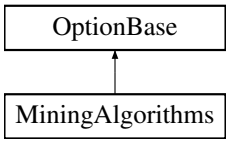
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/MiningAlgorithmBase.hpp
- /Users/esposito/Software/minerule/src/Algorithms/MiningAlgorithmBase.cpp

6.80 MiningAlgorithms Class Reference

```
#include <miningalgorithms.hpp>
```

Inheritance diagram for MiningAlgorithms:



Public Member Functions

- virtual ~MiningAlgorithms ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- virtual OptionBase & subclassForName (const std::string &subclassName)
- RulesMiningAlgorithms & getRulesMiningAlgorithms ()
- const RulesMiningAlgorithms & getRulesMiningAlgorithms () const
- const ItemsetsMiningAlgorithms & getItemsetsMiningAlgorithms () const
- const SequencesMiningAlgorithms & getSequencesMiningAlgorithms () const

6.80.1 Detailed Description

Definition at line 16 of file miningalgorithms.hpp.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 ~MiningAlgorithms()

```
virtual MiningAlgorithms::~MiningAlgorithms ( )  [inline], [virtual]
```

Definition at line 32 of file miningalgorithms.hpp.

```
00032 { };
```

6.80.3 Member Function Documentation

6.80.3.1 className()

```
virtual std::string MiningAlgorithms::className ( ) const  [inline], [virtual]
```

Definition at line 34 of file miningalgorithms.hpp.

```
00034                                       {
00035         return "MiningAlgorithm";
00036   }
```

6.80.3.2 getItemsetsMiningAlgorithms()

```
const ItemsetsMiningAlgorithms & MiningAlgorithms::getItemsetsMiningAlgorithms ( ) const  [inline]
```

Definition at line 57 of file miningalgorithms.hpp.

```
00057 { return itemsetsMiningAlgorithms; }
```

6.80.3.3 getRulesMiningAlgorithms() [1/2]

```
RulesMiningAlgorithms & MiningAlgorithms::getRulesMiningAlgorithms ( )  [inline]
```

Definition at line 55 of file miningalgorithms.hpp.

```
00055 { return rulesMiningAlgorithms; }
```

6.80.3.4 getRulesMiningAlgorithms() [2/2]

```
const RulesMiningAlgorithms & MiningAlgorithms::getRulesMiningAlgorithms ( ) const  [inline]
```

Definition at line 56 of file miningalgorithms.hpp.

```
00056 { return rulesMiningAlgorithms; }
```

6.80.3.5 getSequencesMiningAlgorithms()

```
const SequencesMiningAlgorithms & MiningAlgorithms::getSequencesMiningAlgorithms ( ) const
[inline]
```

Definition at line 58 of file miningalgorithms.hpp.

```
00058 { return sequencesMiningAlgorithms; }
```

6.80.3.6 setOption()

```
virtual void MiningAlgorithms::setOption (
            const std::string & name,
            const std::string & value )  [inline], [virtual]
```

Definition at line 38 of file miningalgorithms.hpp.

```
00038                                                                    {
00039         throw MineruleException( MR_ERROR_OPTION_PARSING, "MiningAlgorithms option class does not
      support option:"+value );
00040   }
```

6.80.3.7 subclassForName()

```
virtual OptionBase & MiningAlgorithms::subclassForName (
            const std::string & subclassName )  [inline], [virtual]
```

Definition at line 42 of file miningalgorithms.hpp.

```
00042                                                                  {
00043         if( subclassName=="rulesmining" ) {
00044           return rulesMiningAlgorithms;
00045         }  else if(subclassName=="itemsetsmining") {
00046           return itemsetsMiningAlgorithms;
00047         }  else if(subclassName=="sequencesmining") {
00048           return sequencesMiningAlgorithms;
00049           } else {
00050           return OptionBase::subclassForName(subclassName);
00051         }
00052   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms.hpp

6.81 minerule::MinMax Class Reference

`#include <Bodymap.hpp>`

Public Member Functions

- MinMax ()
- MinMax (int m1, int m2)
- int minValue ()
- int maxValue ()

Data Fields

- int min
- int max

6.81.1 Detailed Description

Definition at line 58 of file Bodymap.hpp.

6.81.2 Constructor & Destructor Documentation

6.81.2.1 MinMax() [1/2]

```
minerule::MinMax::MinMax ( )  [inline]
```

Definition at line 62 of file Bodymap.hpp.

```
00062 : min(INT_MAX), max(0) {}
```

6.81.2.2 MinMax() [2/2]

```
minerule::MinMax::MinMax (
            int m1,
            int m2 )  [inline]
```

Definition at line 63 of file Bodymap.hpp.

```
00063 : min(m1), max(m2) {}
```

6.81.3 Member Function Documentation

6.81.3.1 maxValue()

```
int minerule::MinMax::maxValue ( )  [inline]
```

Definition at line 65 of file Bodymap.hpp.

```
00065 { return max; }
```

6.81.3.2 minValue()

```
int minerule::MinMax::minValue ( )  [inline]
```

Definition at line 64 of file Bodymap.hpp.

```
00064 { return min; }
```

6.81.4 Field Documentation

6.81.4.1 max

```
int minerule::MinMax::max
```

Definition at line 61 of file Bodymap.hpp.

6.81.4.2 min

```
int minerule::MinMax::min
```

Definition at line 60 of file Bodymap.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp

6.82 minerule::MinMaxPair Class Reference

```
#include <MinMaxPair.hpp>
```

Public Member Functions

- MinMaxPair (int m, int M)
- ~MinMaxPair ()
- void applyConstraints (const MinMaxPair &constr)
- void setMin (int m)
- void setMax (int M)
- int getMin () const
- int getMax () const
- int getDefaultMax () const
- void setDefaultMax ()
- bool validate (int value) const
- bool contains (const MinMaxPair &mm) const

6.82.1 Detailed Description

Definition at line 24 of file MinMaxPair.hpp.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 MinMaxPair()

```
minerule::MinMaxPair::MinMaxPair (
            int m,
            int M )  [inline]
```

Definition at line 44 of file MinMaxPair.hpp.

```
00044                                : _min(m), _max(M) {
00045     assert( _min <= _max );
00046   };
```

6.82.2.2 ~MinMaxPair()

```
minerule::MinMaxPair::~MinMaxPair ( )  [inline]
```

Definition at line 48 of file MinMaxPair.hpp.

```
00048 {};
```

6.82.3 Member Function Documentation

6.82.3.1 applyConstraints()

```
void minerule::MinMaxPair::applyConstraints (
            const MinMaxPair & constr )  [inline]
```

Definition at line 50 of file MinMaxPair.hpp.

```
00050                                                {
00051     assert( _min<=_max );
00052     assert( constr._min <= constr._max );
00053
00054     // the minimal el, cannot be lower than constr.minimal so we set
00055     // it to the maximum of the two
00056     _min = _MAX( _min, constr._min );
00057     // at the same time it cannot be higher than constr.maximal so we
00058     // set it to the minimum of the two
00059     _min = _MIN( _min, constr._max );
00060
00061     // An analogous reasoning lead to the following two lines of code
00062     _max = _MIN( _max, constr._max );
00063     _max = _MAX( _max, constr._min );
00064
00065     assert( _min<=_max );
00066   }
```

6.82.3.2 contains()

```
bool minerule::MinMaxPair::contains (
            const MinMaxPair & mm ) const  [inline]
```

Definition at line 99 of file MinMaxPair.hpp.

```
00099                                                {
00100     return _min<=mm._min && mm._max<=_max;
00101   }
```

6.82.3.3 getDefaultMax()

```
int minerule::MinMaxPair::getDefaultMax ( ) const  [inline]
```

Definition at line 84 of file MinMaxPair.hpp.

```
00084                                {
00085     return 1000;
00086   }
```

6.82.3.4 getMax()

```
int minerule::MinMaxPair::getMax ( ) const  [inline]
```

Definition at line 80 of file MinMaxPair.hpp.

```
00080                       {
00081     return _max;
00082   }
```

6.82.3.5 getMin()

```
int minerule::MinMaxPair::getMin ( ) const  [inline]
```

Definition at line 76 of file MinMaxPair.hpp.

```
00076                       {
00077     return _min;
00078   }
```

6.82.3.6 setDefaultMax()

```
void minerule::MinMaxPair::setDefaultMax ( )  [inline]
```

Definition at line 88 of file MinMaxPair.hpp.

```
00088                        {
00089     setMax(getDefaultMax());
00090   }
```

6.82.3.7 setMax()

```
void minerule::MinMaxPair::setMax (
            int M )  [inline]
```

Definition at line 72 of file MinMaxPair.hpp.

```
00072                       {
00073     _max=M;
00074   }
```

6.82.3.8 setMin()

```
void minerule::MinMaxPair::setMin (
            int m )  [inline]
```

Definition at line 68 of file MinMaxPair.hpp.

```
00068                       {
00069     _min=m;
00070   }
```

6.82.3.9 validate()

```
bool minerule::MinMaxPair::validate (
            int value ) const  [inline]
```

Definition at line 92 of file MinMaxPair.hpp.

```
00092                                  {
00093     if( _min <= value && value <= _max )
00094       return true;
00095     else
00096       return false;
00097   }
```

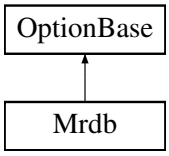
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MinMaxPair.hpp

6.83 Mrdb Class Reference

`#include <mrdb.hpp>`

Inheritance diagram for Mrdb:



Public Member Functions

- Mrdb ()
- virtual ~Mrdb ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- std::string getName () const
- std::string getUsername () const
- std::string getPassword () const
- std::string getDBMS () const
- bool getCacheWrites () const
- mrdb::Connection * getMRDBConnection () const
- void setName (const std::string &str)
- void setUsername (const std::string &str)
- void setPassword (const std::string &str)
- void setCacheWrites (bool v)
- void setDBMS (std::string v)
- void setConnection (mrdb::Connection *con)
- void clearConnection ()
- std::string getLibName (const std::string &dbms)
- void resetConnection ()

6.83.1 Detailed Description

Definition at line 17 of file mrdb.hpp.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 Mrdb()

`Mrdb::Mrdb ( ) [inline]`

Definition at line 26 of file mrdb.hpp.

```
00026 : connection(NULL), cacheWrites(false), dbms("postgres"){};
```

6.83.2.2 ~Mrdb()

```
virtual Mrdb::~Mrdb ( )  [inline], [virtual]
```

Definition at line 28 of file mrdb.hpp.

```
00028 {}
```

6.83.3 Member Function Documentation

6.83.3.1 className()

```
virtual std::string Mrdb::className ( ) const  [inline], [virtual]
```

Definition at line 30 of file mrdb.hpp.

```
00030 { return "mrdb"; }
```

6.83.3.2 clearConnection()

```
void Mrdb::clearConnection ( )  [inline]
```

Definition at line 69 of file mrdb.hpp.

```
00069                          {
00070     if (connection != NULL) {
00071       delete connection;
00072       connection = NULL;
00073     }
00074   }
```

6.83.3.3 getCacheWrites()

```
bool Mrdb::getCacheWrites ( ) const  [inline]
```

Definition at line 42 of file mrdb.hpp.

```
00042 { return cacheWrites; }
```

6.83.3.4 getDBMS()

```
std::string Mrdb::getDBMS ( ) const  [inline]
```

Definition at line 40 of file mrdb.hpp.

```
00040 { return dbms; }
```

6.83.3.5 getLibName()

```
std::string Mrdb::getLibName (
            const std::string & dbms )  [inline]
```

Definition at line 76 of file mrdb.hpp.

```
00076                                                {
00077     if(dbms=="postgres") {
00078       return "libmrdb_pg.so";
00079     } else {
00080       throw MineruleException(MR_ERROR_DATABASE_ERROR,
00081       "Database " + dbms + " not supported yet.");
00082     }
00083   }
```

6.83.3.6 getMRDBConnection()

```
mrdb::Connection * Mrdb::getMRDBConnection ( ) const  [inline]
```

Definition at line 46 of file mrdb.hpp.

```
00046                                              {
00047     if (connection == NULL) {
00048       throw MineruleException(
00049           MR_ERROR_DATABASE_ERROR,
00050           (std::string) "Check out the parameters, it looks like that"
00051                         " you did not specified any mrdb data source");
00052     }
00053
00054     return connection;
00055   }
```

6.83.3.7 getName()

```
std::string Mrdb::getName ( ) const  [inline]
```

Definition at line 35 of file mrdb.hpp.

```
00035 { return name; }
```

6.83.3.8 getPassword()

```
std::string Mrdb::getPassword ( ) const  [inline]
```

Definition at line 38 of file mrdb.hpp.

```
00038 { return password; }
```

6.83.3.9 getUsername()

```
std::string Mrdb::getUsername ( ) const  [inline]
```

Definition at line 37 of file mrdb.hpp.

```
00037 { return username; }
```

6.83.3.10 resetConnection()

```
void Mrdb::resetConnection ( )  [inline]
```

Definition at line 87 of file mrdb.hpp.

```
00087                           {
00088     clearConnection();
00089
00090     std::string libName(getLibName(getDBMS()));
00091     static void *handle = dlopen( libName.c_str(), RTLD_LAZY);
00092
00093     if (handle == NULL) {
00094       throw MineruleException(MR_ERROR_DATABASE_ERROR, dlerror());
00095     }
00096
00097     MRDBConnectFun connect = (MRDBConnectFun)dlsym(handle, "connect");
00098
00099     if (connect == NULL) {
00100       throw MineruleException(MR_ERROR_DATABASE_ERROR, dlerror());
00101     }
00102
00103     setConnection(connect(getName().c_str(), getUsername().c_str(),
00104                           getPassword().c_str()));
00105   }
```

6.83.3.11 setCacheWrites()

```
void Mrdb::setCacheWrites (
             bool v )  [inline]
```

Definition at line 63 of file mrdb.hpp.

```
00063 { cacheWrites = v; }
```

6.83.3.12 setConnection()

```
void Mrdb::setConnection (
             mrdb::Connection * con )  [inline]
```

Definition at line 67 of file mrdb.hpp.

```
00067 { connection = con; }
```

6.83.3.13 setDBMS()

```
void Mrdb::setDBMS (
             std::string v )  [inline]
```

Definition at line 65 of file mrdb.hpp.

```
00065 { dbms = v; }
```

6.83.3.14 setName()

```
void Mrdb::setName (
            const std::string & str )  [inline]
```

Definition at line 57 of file mrdb.hpp.
```
00057 { name = str; }
```

6.83.3.15 setOption()

```
virtual void Mrdb::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.83.3.16 setPassword()

```
void Mrdb::setPassword (
            const std::string & str )  [inline]
```

Definition at line 61 of file mrdb.hpp.
```
00061 { password = str; }
```

6.83.3.17 setUsername()

```
void Mrdb::setUsername (
            const std::string & str )  [inline]
```

Definition at line 59 of file mrdb.hpp.
```
00059 { username = str; }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/mrdb.hpp

6.84 minerule::MRDebugPusher Class Reference

`#include <MineruleLogs.hpp>`

Public Member Functions

- MRDebugPusher (const std::string &str)
- ~MRDebugPusher ()

6.84.1 Detailed Description

This is the safest way to call for MRDebugPush & MRDebugPop. If this class is used correctly, MRDebugPop is automatically called in the destructor (this also ensure exception safety!

Definition at line 114 of file MineruleLogs.hpp.

6.84.2 Constructor & Destructor Documentation

6.84.2.1 MRDebugPusher()

```
minerule::MRDebugPusher::MRDebugPusher (
            const std::string & str )  [inline]
```

Definition at line 116 of file MineruleLogs.hpp.

```
00116                                          {
00117       MRDebugPush(str);
00118     }
```

6.84.2.2 ~MRDebugPusher()

```
minerule::MRDebugPusher::~MRDebugPusher ( )  [inline]
```

Definition at line 120 of file MineruleLogs.hpp.

```
00120                       {
00121       MRDebugPop();
00122     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp

6.85 minerule::MRErrPusher Class Reference

```
#include <MineruleLogs.hpp>
```

Public Member Functions

- MRErrPusher (const std::string &str)
- ~MRErrPusher ()

6.85.1 Detailed Description

This is the safest way to call for MRErrPush & MRErrPop. If this class is used correctly, MRErrPop is automatically called in the destructor (this also ensure exception safety!

Definition at line 81 of file MineruleLogs.hpp.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 MRErrPusher()

```
minerule::MRErrPusher::MRErrPusher (
            const std::string & str )  [inline]
```

Definition at line 83 of file MineruleLogs.hpp.

```
00083                                       {
00084       MRErrPush(str);
00085     }
```

6.85.2.2 ~MRErrPusher()

```
minerule::MRErrPusher::~MRErrPusher ( )  [inline]
```

Definition at line 87 of file MineruleLogs.hpp.

```
00087                     {
00088       MRErrPop();
00089     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp

6.86 minerule::MRLogger Class Reference

```
#include <MRLogger.hpp>
```

Public Member Functions

- MRLogger (std::ostream &ostr)
- MRLogger ()
- ~MRLogger (void)
- void setStream (std::ostream &ostr)
- std::ostream * getStream () const
- void updateIndentString ()
- void push (const std::string &descr)
- void pop ()
- double getCurrentCpuSecs () const
- double getCurrentTimeSecs () const
- double getCurrentCpuDelta () const
- double getCurrentTimeDelta () const
- std::ostream & log ()
- size_t getLogLevel ()
- void setLogLevel (size_t newLevel)
- size_t getIndentLen ()
- void startMeasuring (std::string description)
- void stopMeasuring (std::string description)
- void logMeasurements ()

6.86.1 Detailed Description

Definition at line 26 of file MRLogger.hpp.

6.86.2 Constructor & Destructor Documentation

6.86.2.1 MRLogger() [1/2]

```
minerule::MRLogger::MRLogger (
            std::ostream & ostr )
```

Definition at line 67 of file MRLogger.cpp.

```
00067                                   : indentInset("  "), os(NULL), logLevel(100), curLogLevel(1) {
00068     setStream(ostr);
00069     updateIndentString();
00070   }
```

6.86.2.2 MRLogger() [2/2]

```
minerule::MRLogger::MRLogger (
            void  )
```

Definition at line 63 of file MRLogger.cpp.

```
00063                       : indentInset("  "), os(NULL), logLevel(100), curLogLevel(1) {
00064     updateIndentString();
00065   }
```

6.86.2.3 ~MRLogger()

```
minerule::MRLogger::~MRLogger (
            void  )
```

Definition at line 76 of file MRLogger.cpp.

```
00076                           {
00077     while(!logStack.empty()) {
00078       logStack.pop_back();
00079     }
00080   }
```

6.86.3 Member Function Documentation

6.86.3.1 getCurrentCpuDelta()

```
double minerule::MRLogger::getCurrentCpuDelta ( ) const
```

Definition at line 48 of file MRLogger.cpp.

```
00048                                          {
00049     LogInfo curInfo("");
00050     return getCpuSecs(curInfo,logStack.back());
00051   }
```

6.86.3.2 getCurrentCpuSecs()

```
double minerule::MRLogger::getCurrentCpuSecs ( ) const
```

Definition at line 43 of file MRLogger.cpp.

```
00043                                         {
00044     LogInfo curInfo("");
00045     return getCpuSecs(curInfo,logStack.front());
00046   }
```

6.86.3.3 getCurrentTimeDelta()

```
double minerule::MRLogger::getCurrentTimeDelta ( ) const
```

Definition at line 58 of file MRLogger.cpp.

```
00058                                           {
00059     LogInfo curInfo("");
00060     return getTimeSecs(curInfo,logStack.back());
00061   }
```

6.86.3.4 getCurrentTimeSecs()

```
double minerule::MRLogger::getCurrentTimeSecs ( ) const
```

Definition at line 53 of file MRLogger.cpp.

```
00053                                           {
00054     LogInfo curInfo("");
00055     return getTimeSecs(curInfo,logStack.front());
00056   }
```

6.86.3.5 getIndentLen()

```
size_t minerule::MRLogger::getIndentLen ( )  [inline]
```

Definition at line 129 of file MRLogger.hpp.

```
00129 { return indentString.size(); }
```

6.86.3.6 getLogLevel()

```
size_t minerule::MRLogger::getLogLevel ( )  [inline]
```

Definition at line 121 of file MRLogger.hpp.

```
00121                                 {
00122                         return logLevel;
00123                 }
```

6.86.3.7 getStream()

```
std::ostream * minerule::MRLogger::getStream ( ) const  [inline]
```

Definition at line 99 of file MRLogger.hpp.

```
00099                                           {
00100                         return os;
00101                 }
```

6.86.3.8 log()

```
std::ostream & minerule::MRLogger::log ( )  [inline]
```

Definition at line 113 of file MRLogger.hpp.

```
00113                                   {
00114                         if(curLogLevel>logLevel)
00115                                 return nullLog;
00116                         indent();
00117                         *os<<" ";
00118                         return *os;
00119                 }
```

6.86.3.9 logMeasurements()

```
void minerule::MRLogger::logMeasurements ( )
```

Definition at line 145 of file MRLogger.cpp.

```
00145                                  {
00146           push("Showing measured execution times:");
00147
00148                 for( Measurements::const_iterator it=measurements.begin(); it!=measurements.end();
      ++it ) {
00149                   logMeasurement(it->first, it->second);
00150           }
00151
00152           pop();
00153   }
```

6.86.3.10 pop()

```
void minerule::MRLogger::pop ( )
```

Definition at line 115 of file MRLogger.cpp.

```
00115                     {
00116     if(logStack.empty()) return;
00117
00118     if(curLogLevel>logLevel) {
00119       curLogLevel--;
00120       return;
00121     } else {
00122       curLogLevel--;
00123     }
00124     assert(os!=NULL);
00125     LogInfo curInfo("");
00126     std::string timeMemInfo=evalTimeMemInfo(curInfo);
00127
00128     logStack.pop_back();
00129     updateIndentString();
00130     indent();
00131
00132     if(!logStack.empty())
00133       *os«indentInset«END_SEPARATOR« StringUtils::toBold(timeMemInfo) «std::endl;
00134     else
00135           *os«END_SEPARATOR«StringUtils::toBold(timeMemInfo)«std::endl;
00136
00137     indent();
00138     *os«std::endl;
00139   }
```

6.86.3.11 push()

```
void minerule::MRLogger::push (
            const std::string & descr )
```

Definition at line 90 of file MRLogger.cpp.

```
00090                                          {
00091     curLogLevel++;
00092     if(curLogLevel>logLevel)
00093       return;
00094
00095     assert(os!=NULL);
00096     indent();
00097     *os«std::endl;
00098
00099    std::string insetSep;
00100     if(logStack.empty())
00101       insetSep="";
00102     else
00103       insetSep=indentInset;
```

```
00104
00105     LogInfo li(insetSep+CONT_SEPARATOR);
00106     logStack.push_back(li);
00107
00108     indent();
00109     *os«insetSep«"+"« StringUtils::toBold(descr) «std::endl;
00110     updateIndentString();
00111   }
```

6.86.3.12 setLogLevel()

```
void minerule::MRLogger::setLogLevel (
            size_t newLevel )  [inline]
```

Definition at line 125 of file MRLogger.hpp.

```
00125                                                {
00126                         logLevel=newLevel;
00127                 }
```

6.86.3.13 setStream()

```
void minerule::MRLogger::setStream (
            std::ostream & ostr )
```

Definition at line 72 of file MRLogger.cpp.

```
00072                                           {
00073     os=&ostr;
00074   };
```

6.86.3.14 startMeasuring()

```
void minerule::MRLogger::startMeasuring (
            std::string description )  [inline]
```

Definition at line 131 of file MRLogger.hpp.

```
00131                                                        {
00132                         MeasurementInfo& m = measurements[description];
00133                         assert(!m.startIsValid);
00134
00135                         m.start = LogInfo(""); // saving start time
00136                         m.startIsValid = true;
00137                 }
```

6.86.3.15 stopMeasuring()

```
void minerule::MRLogger::stopMeasuring (
            std::string description )  [inline]
```

Definition at line 139 of file MRLogger.hpp.

```
00139                                                       {
00140                         MeasurementInfo& m = measurements[description];
00141                         LogInfo stop("");
00142
00143                         assert(m.startIsValid);
00144
00145                         m.totCpu += getCpuSecs(stop, m.start);
00146                         m.totTime += getTimeSecs(stop, m.start);
00147                         m.startIsValid = false;
00148                 }
```

6.86.3.16 updateIndentString()

```
void minerule::MRLogger::updateIndentString ( )
```

Definition at line 82 of file MRLogger.cpp.

```
00082                                    {
00083     LogStack::const_iterator it;
00084     indentString="";
00085     for(it=logStack.begin(); it!=logStack.end(); it++) {
00086       indentString+=it->indent;
00087     }
00088   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/MRLogger.hpp
- /Users/esposito/Software/minerule/src/Utils/MRLogger.cpp

6.87 minerule::MRLogPusher Class Reference

```
#include <MineruleLogs.hpp>
```

Public Member Functions

- MRLogPusher (const std::string &str)
- ~MRLogPusher ()

6.87.1 Detailed Description

This is the safest way to call for MRLogPush & MRLogPop. If this class is used correctly, MRLogPop is automatically called in the destructor (this also ensure exception safety!

Definition at line 63 of file MineruleLogs.hpp.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 MRLogPusher()

```
minerule::MRLogPusher::MRLogPusher (
            const std::string & str )  [inline]
```

Definition at line 65 of file MineruleLogs.hpp.

```
00065                                    {
00066       MRLogPush(str);
00067     }
```

6.87.2.2 ~MRLogPusher()

```
minerule::MRLogPusher::~MRLogPusher ( )  [inline]
```

Definition at line 69 of file MineruleLogs.hpp.

```
00069                     {
00070       MRLogPop();
00071     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp

6.88 MRResultSetIterator Class Reference

```
#include <MRResultSet.hpp>
```

Public Member Functions

- MRResultSetIterator (mrdb::PreparedStatement *q)
- bool next ()
- void reset ()
- mrdb::ResultSet * getResultSet ()

6.88.1 Detailed Description

Definition at line 23 of file MRResultSet.hpp.

6.88.2 Constructor & Destructor Documentation

6.88.2.1 MRResultSetIterator()

```
MRResultSetIterator::MRResultSetIterator (
            mrdb::PreparedStatement * q )  [inline]
```

Definition at line 28 of file MRResultSet.hpp.

```
00028                                                : query(q), rs(NULL) {
00029     rs = query->executeQuery();
00030   }
```

6.88.3 Member Function Documentation

6.88.3.1 getResultSet()

```
mrdb::ResultSet * MRResultSetIterator::getResultSet ( )  [inline]
```

Definition at line 42 of file MRResultSet.hpp.

```
00042 { return rs; }
```

6.88.3.2 next()

```
bool MRResultSetIterator::next ( )  [inline]
```

Definition at line 32 of file MRResultSet.hpp.

```
00032 { return rs->next(); }
```

6.88.3.3 reset()

```
void MRResultSetIterator::reset ( )  [inline]
```

Definition at line 34 of file MRResultSet.hpp.

```
00034                {
00035     if (rs != NULL) {
00036       delete rs;
00037     }
00038
00039     rs = query->executeQuery();
00040   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/MRResultSet.hpp

6.89 minerule::MRWarnPusher Class Reference

```
#include <MineruleLogs.hpp>
```

Public Member Functions

- MRWarnPusher (const std::string &str)
- ~MRWarnPusher ()

6.89.1 Detailed Description

This is the safest way to call for MRWarnPush & MRWarnPop. If this class is used correctly, MRWarnPop is automatically called in the destructor (this also ensure exception safety!

Definition at line 97 of file MineruleLogs.hpp.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 MRWarnPusher()

```
minerule::MRWarnPusher::MRWarnPusher (
            const std::string & str )  [inline]
```

Definition at line 99 of file MineruleLogs.hpp.

```
00099                                       {
00100       MRWarnPush(str);
00101     }
```

6.89.2.2 ~MRWarnPusher()

```
minerule::MRWarnPusher::~MRWarnPusher ( )  [inline]
```

Definition at line 103 of file MineruleLogs.hpp.

```
00103                       {
00104       MRWarnPop();
00105     }
```

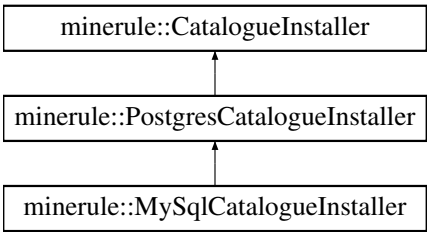
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleLogs.hpp

6.90 minerule::MySqlCatalogueInstaller Class Reference

```
#include <MySqlCatalogueInstaller.hpp>
```

Inheritance diagram for minerule::MySqlCatalogueInstaller:



Public Types

- enum SupportedDbms { MySql , Postgres }

Public Member Functions

- MySqlCatalogueInstaller ()
- virtual ~MySqlCatalogueInstaller ()
- virtual void installMRQuery ()
- virtual void installMRAttList ()
- virtual void installMREqKeys ()
- virtual void installMREqKeysCol ()
- virtual void installMRDepFun ()
- virtual void installMRDepFunCol ()
- virtual void installMRAutoincrement ()
- virtual void initializeAutoincrement ()
- virtual void dropMRQuery ()
- virtual void dropMRAttList ()
- virtual void dropMREqKeys ()
- virtual void dropMREqKeysCol ()
- virtual void dropMRDepFun ()
- virtual void dropMRDepFunCol ()
- virtual void dropMRAutoincrement ()
- virtual void install ()
- virtual void uninstall ()

Static Public Member Functions

- static CatalogueInstaller * newInstaller (SupportedDbms dbms)
- static CatalogueInstaller * newInstaller ()

Protected Attributes

- mrdb::Connection * _connection
- mrdb::Statement * _statement

6.90.1 Detailed Description

Definition at line 25 of file MySqlCatalogueInstaller.hpp.

6.90.2 Member Enumeration Documentation

6.90.2.1 SupportedDbms

```
enum minerule::CatalogueInstaller::SupportedDbms  [inherited]
```

Enumerator

MySql	
Postgres	

Definition at line 29 of file CatalogueInstaller.hpp.
```
00029 { MySql, Postgres } SupportedDbms;
```

6.90.3 Constructor & Destructor Documentation

6.90.3.1 MySqlCatalogueInstaller()

```
minerule::MySqlCatalogueInstaller::MySqlCatalogueInstaller ( )  [inline]
```

Definition at line 27 of file MySqlCatalogueInstaller.hpp.
```
00027 : PostgresCatalogueInstaller() {} ;
```

6.90.3.2 ~MySqlCatalogueInstaller()

```
virtual minerule::MySqlCatalogueInstaller::~MySqlCatalogueInstaller ( )  [inline], [virtual]
```

Definition at line 28 of file MySqlCatalogueInstaller.hpp.
```
00028 {};
```

6.90.4 Member Function Documentation

6.90.4.1 dropMRAttList()

```
void minerule::PostgresCatalogueInstaller::dropMRAttList ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 24 of file PostgresCatalogueInstaller.cpp.
```
00024                                                   {
00025                 _statement->execute("DROP TABLE IF EXISTS mr_att_lists");
00026         }
```

6.90.4.2 dropMRAutoincrement()

```
void minerule::PostgresCatalogueInstaller::dropMRAutoincrement ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 44 of file PostgresCatalogueInstaller.cpp.
```
00044                                                       {
00045                 _statement->execute("DROP TABLE IF EXISTS mr_autoincrement");
00046         }
```

6.90.4.3 dropMRDepFun()

```
void minerule::PostgresCatalogueInstaller::dropMRDepFun ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 36 of file PostgresCatalogueInstaller.cpp.

```
00036                                                   {
00037                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun");
00038         }
```

6.90.4.4 dropMRDepFunCol()

```
void minerule::PostgresCatalogueInstaller::dropMRDepFunCol ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 40 of file PostgresCatalogueInstaller.cpp.

```
00040                                                      {
00041                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun_col");
00042         }
```

6.90.4.5 dropMREqKeys()

```
void minerule::PostgresCatalogueInstaller::dropMREqKeys ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 28 of file PostgresCatalogueInstaller.cpp.

```
00028                                                   {
00029                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys");
00030         }
```

6.90.4.6 dropMREqKeysCol()

```
void minerule::PostgresCatalogueInstaller::dropMREqKeysCol ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 32 of file PostgresCatalogueInstaller.cpp.

```
00032                                                      {
00033                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys_col");
00034         }
```

6.90.4.7 dropMRQuery()

```
void minerule::PostgresCatalogueInstaller::dropMRQuery ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 20 of file PostgresCatalogueInstaller.cpp.

```
00020                                                       {
00021                 _statement->execute("DROP TABLE IF EXISTS mr_query");
00022         }
```

6.90.4.8 initializeAutoincrement()

```
void minerule::PostgresCatalogueInstaller::initializeAutoincrement ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 118 of file PostgresCatalogueInstaller.cpp.

```
00118                                                                {
00119                 _statement->execute("INSERT INTO mr_autoincrement (table_name, current_id) VALUES
      ('mr_att_lists',0),('mr_eq_keys',0),('mr_query',0)");
00120         }
```

6.90.4.9 install()

```
virtual void minerule::CatalogueInstaller::install ( )  [inline], [virtual], [inherited]
```

Definition at line 57 of file CatalogueInstaller.hpp.

```
00057                                {
00058                         uninstall();
00059
00060                         installMRQuery();
00061                         installMRAttList();
00062                         installMREqKeys();
00063                         installMREqKeysCol();
00064                         installMRDepFun();
00065                         installMRDepFunCol();
00066                         installMRAutoincrement();
00067                         initializeAutoincrement();
00068                 }
```

6.90.4.10 installMRAttList()

```
void minerule::PostgresCatalogueInstaller::installMRAttList ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 66 of file PostgresCatalogueInstaller.cpp.

```
00066                                                       {
00067                 _statement->execute("CREATE TABLE mr_att_lists (att_list_id  integer NOT NULL,col_name
      varchar(128) NOT NULL)");
00068                 _statement->execute("CREATE INDEX mr_att_lists_att_list_id_index ON
      mr_att_lists(att_list_id)");
00069         }
```

6.90.4.11 installMRAutoincrement()

```
void minerule::PostgresCatalogueInstaller::installMRAutoincrement ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 109 of file PostgresCatalogueInstaller.cpp.

```
00109                                                       {
00110                 _statement->execute("CREATE TABLE mr_autoincrement (table_name varchar(128) NOT NULL,"
00111                                                     "current_id integer NOT NULL,"
00112                                                     "CONSTRAINT mr_autoincrement_primary PRIMARY
      KEY (table_name))");
00113
00114         }
```

6.90.4.12 installMRDepFun()

```
void minerule::PostgresCatalogueInstaller::installMRDepFun ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 91 of file PostgresCatalogueInstaller.cpp.

```
00091                                                   {
00092                 _statement->execute("CREATE TABLE mr_dep_fun (lhs_tab_name  varchar(128) NOT NULL,"
00093                                                         "lhs_att_list_id  integer NOT NULL,"
00094                                                         "rhs_tab_name  varchar(128) NOT NULL,"
00095                                                         "rhs_att_list_id  integer NOT NULL,"
00096                                                         "order_type char(1),"
00097                                                         "CONSTRAINT mr_dep_fun_primary PRIMARY
      KEY(lhs_tab_name,lhs_att_list_id,rhs_tab_name, rhs_att_list_id))");
00098         }
```

6.90.4.13 installMRDepFunCol()

```
void minerule::PostgresCatalogueInstaller::installMRDepFunCol ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 102 of file PostgresCatalogueInstaller.cpp.

```
00102                                                       {
00103                 _statement->execute("CREATE TABLE mr_dep_fun_col (col_id  integer NOT NULL, col_name
      varchar(128) NOT NULL)");
00104                 _statement->execute("CREATE INDEX mr_dep_fun_col_col_id_index ON
      mr_dep_fun_col(col_id)");
00105         }
```

6.90.4.14 installMREqKeys()

```
void minerule::PostgresCatalogueInstaller::installMREqKeys ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 73 of file PostgresCatalogueInstaller.cpp.

```
00073                                                   {
00074                 _statement->execute("CREATE TABLE mr_eq_keys (tab_name  varchar(128) NOT NULL,"
00075                                                         "key_id  integer NOT NULL,"
00076                                                         "ref_key_id  integer NOT NULL,"
00077                                                         "order_type char(1),"
00078                                                         "CONSTRAINT mr_eq_keys_primary PRIMARY
      KEY(tab_name,key_id,ref_key_id))");
00079         }
```

6.90.4.15 installMREqKeysCol()

```
void minerule::PostgresCatalogueInstaller::installMREqKeysCol ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 83 of file PostgresCatalogueInstaller.cpp.

```
00083                                                   {
00084                 _statement->execute("CREATE TABLE mr_eq_keys_col (key_id  integer NOT NULL,"
00085                                                     "col_name  varchar(128) NOT NULL)");
00086                 _statement->execute("CREATE INDEX mr_eq_keys_col_key_id_index ON
      mr_eq_keys_col(key_id)");
00087         }
```

6.90.4.16 installMRQuery()

```
void minerule::PostgresCatalogueInstaller::installMRQuery ( )  [virtual], [inherited]
```

Implements minerule::CatalogueInstaller.

Definition at line 50 of file PostgresCatalogueInstaller.cpp.

```
00050                                                   {
00051                 _statement->execute("CREATE TABLE mr_query (query_id  integer NOT NULL PRIMARY KEY,"
00052                                                     "query_text  text NOT NULL,"
00053                                                     "query_name  varchar(128) NOT NULL,     "
00054                                                     "tab_results_name varchar(128) NOT NULL,"
00055                                                     "source_tab_name varchar(128) NOT NULL,"
00056                                                     "gal  integer NOT NULL,"
00057                                                     "ral  integer NOT NULL,"
00058                                                       "cal  integer)");
00059                 _statement->execute("CREATE INDEX mr_query_tab_results_name_index ON
      mr_query(tab_results_name)");
00060                 _statement->execute("CREATE INDEX mr_query_query_name_index ON mr_query
      (query_name)");
00061                 _statement->execute("CREATE INDEX mr_query_source_tab_name_index ON mr_query
      (source_tab_name)");
00062         }
```

6.90.4.17 newInstaller() [1/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller ( )  [static], [inherited]
```

Definition at line 34 of file CatalogueInstaller.cpp.

```
00034                                                   {
00035                 std::string dbmsStr = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00036
00037                 if(dbmsStr == "mysql") return newInstaller(MySql);
00038                 if(dbmsStr == "postgres") return newInstaller(Postgres);
00039
00040                 throw MineruleException(MR_ERROR_OPTION_CONFIGURATION,std::string("Option mrdb::dbms
      is set to an unsupported value.") +
00041                         " Current value is:"+dbmsStr+ ". Supported values are 'postgres' and
      'mysql'.");
00042         }
```

6.90.4.18 newInstaller() [2/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller (
            SupportedDbms dbms )  [static], [inherited]
```

Definition at line 25 of file CatalogueInstaller.cpp.

```
00025                                                                   {
00026                 switch(dbms) {
00027                 case MySql:
00028                         return new MySqlCatalogueInstaller();
00029                 case Postgres:
00030                         return new PostgresCatalogueInstaller();
00031                 }
00032         }
```

6.90.4.19 uninstall()

```
virtual void minerule::CatalogueInstaller::uninstall ( )  [inline], [virtual], [inherited]
```

Definition at line 70 of file CatalogueInstaller.hpp.

```
00070                                 {
00071                         dropMRQuery();
00072                         dropMRAttList();
00073                         dropMREqKeys();
00074                         dropMREqKeysCol();
00075                         dropMRDepFun();
00076                         dropMRDepFunCol();
00077                         dropMRAutoincrement();
00078                 }
```

6.90.5 Field Documentation

6.90.5.1 _connection

```
mrdb::Connection* minerule::CatalogueInstaller::_connection  [protected], [inherited]
```

Definition at line 26 of file CatalogueInstaller.hpp.

6.90.5.2 _statement

```
mrdb::Statement* minerule::CatalogueInstaller::_statement  [protected], [inherited]
```

Definition at line 27 of file CatalogueInstaller.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/MySqlCatalogueInstaller.hpp

6.91 minerule::NewRule Class Reference

`#include <Bodymap.hpp>`

Public Member Functions

- NewRule ()
- NewRule (std::map< ItemType, BodyMapElement >::iterator b, GidList &g)
- NewRule (NewRule &r, std::map< ItemType, BodyMapElement >::iterator b, GidList &g)
- NewRule (std::map< ItemType, BodyMapElement >::iterator b, std::map< ItemType, MapElement >::iterator h, GidList &g, int bs)
- NewRule (NewRule &r, std::map< ItemType, BodyMapElement >::iterator b, GidList &g, int bs)
- NewRule (NewRule &r, std::map< ItemType, BodyMapElement >::iterator b, int bs, GidList &g)
- NewRule (NewRule &r, std::map< ItemType, MapElement >::iterator h, GidList &g)

Data Fields

- std::vector< ItemType > body
- std::vector< ItemType > head
- std::map< ItemType, BodyMapElement >::iterator lastBody
- std::map< ItemType, MapElement >::iterator lastHead
- bool satisfy
- GidList gids
- int bodySupp

Friends

- std::ostream & operator<< (std::ostream &out, NewRule r)

6.91.1 Detailed Description

Definition at line 90 of file Bodymap.hpp.

6.91.2 Constructor & Destructor Documentation

6.91.2.1 NewRule() [1/7]

`minerule::NewRule::NewRule ( )  [inline]`

Definition at line 98 of file Bodymap.hpp.

```
00098 : satisfy(false) {}
```

6.91.2.2 NewRule() [2/7]

```
minerule::NewRule::NewRule (
            std::map< ItemType, BodyMapElement >::iterator b,
            GidList & g )  [inline]
```

Definition at line 99 of file Bodymap.hpp.

```
00099                                                                   : lastBody(b),
      satisfy(false), gids(g) {
00100                         body.push_back(b->first);
00101                         lastHead = b->second.heads.begin();
00102                 }
```

6.91.2.3 NewRule() [3/7]

```
minerule::NewRule::NewRule (
            NewRule & r,
            std::map< ItemType, BodyMapElement >::iterator b,
            GidList & g )  [inline]
```

Definition at line 103 of file Bodymap.hpp.

```
00103                                                                                 :
      body(r.body), lastBody(b), satisfy(r.satisfy), gids(g) {
00104                         body.push_back(b->first);
00105                         lastHead = b->second.heads.begin();
00106                 }
```

6.91.2.4 NewRule() [4/7]

```
minerule::NewRule::NewRule (
            std::map< ItemType, BodyMapElement >::iterator b,
            std::map< ItemType, MapElement >::iterator h,
            GidList & g,
            int bs )  [inline]
```

Definition at line 107 of file Bodymap.hpp.

```
00107
                              : lastBody(b), lastHead(h), satisfy(false), gids(g), bodySupp(bs) {
00108                         body.push_back(b->first);
00109                         head.push_back(h->first);
00110                 }
```

6.91.2.5 NewRule() [5/7]

```
minerule::NewRule::NewRule (
            NewRule & r,
            std::map< ItemType, BodyMapElement >::iterator b,
            GidList & g,
            int bs )  [inline]
```

Definition at line 111 of file Bodymap.hpp.

```
00111                                                                                 :
      body(r.body), head(r.head), lastBody(b), lastHead(r.lastHead), satisfy(r.satisfy), gids(g),
      bodySupp(bs) {
00112                         body.push_back(b->first);
00113                 }
```

6.91.2.6 NewRule() [6/7]

```
minerule::NewRule::NewRule (
            NewRule & r,
            std::map< ItemType, BodyMapElement >::iterator b,
            int bs,
            GidList & g ) [inline]
```

Definition at line 114 of file Bodymap.hpp.

```
00114 : body(r.body), head(r.head), lastBody(b), lastHead(r.lastHead), satisfy(r.satisfy), gids(g), bodySupp(bs) {
00115                 head.push_back(b->first);
00116         }
```

6.91.2.7 NewRule() [7/7]

```
minerule::NewRule::NewRule (
            NewRule & r,
            std::map< ItemType, MapElement >::iterator h,
            GidList & g ) [inline]
```

Definition at line 117 of file Bodymap.hpp.

```
00117 : body(r.body), head(r.head), lastBody(r.lastBody), lastHead(h), satisfy(r.satisfy), gids(g), bodySupp(r.bodySupp) {
00118                 head.push_back(h->first);
00119         }
```

6.91.3 Friends And Related Function Documentation

6.91.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            NewRule r ) [friend]
```

Definition at line 120 of file Bodymap.hpp.

```
00120                                                         {
00121                 std::vector<ItemType>::iterator i;
00122                 for (i=r.body.begin(); i!=r.body.end(); i++) {
00123                         if (i != r.body.begin()) out « ", ";
00124                         out « *i;
00125                 }
00126                 out « " -> ";
00127                 for (i=r.head.begin(); i!=r.head.end(); i++) {
00128                         if (i != r.head.begin()) out « ", ";
00129                         out « *i;
00130                 }
00131         //out « std::endl;
00132                 return out;
00133         }
```

6.91.4 Field Documentation

6.91.4.1 body

```
std::vector<ItemType> minerule::NewRule::body
```

Definition at line 92 of file Bodymap.hpp.

6.91.4.2 bodySupp

```
int minerule::NewRule::bodySupp
```

Definition at line 97 of file Bodymap.hpp.

6.91.4.3 gids

```
GidList minerule::NewRule::gids
```

Definition at line 96 of file Bodymap.hpp.

6.91.4.4 head

```
std::vector<ItemType> minerule::NewRule::head
```

Definition at line 92 of file Bodymap.hpp.

6.91.4.5 lastBody

```
std::map<ItemType,BodyMapElement>::iterator minerule::NewRule::lastBody
```

Definition at line 93 of file Bodymap.hpp.

6.91.4.6 lastHead

```
std::map<ItemType,MapElement>::iterator minerule::NewRule::lastHead
```

Definition at line 94 of file Bodymap.hpp.

6.91.4.7 satisfy

```
bool minerule::NewRule::satisfy
```

Definition at line 95 of file Bodymap.hpp.

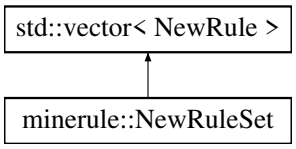
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp

6.92 minerule::NewRuleSet Class Reference

```
#include <Bodymap.hpp>
```

Inheritance diagram for minerule::NewRuleSet:



Data Fields

- T elements
  STL member.

6.92.1 Detailed Description

Definition at line 136 of file Bodymap.hpp.

6.92.2 Field Documentation

6.92.2.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

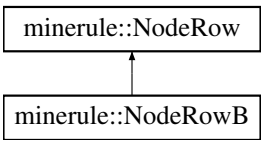
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp

6.93 minerule::NodeRow Class Reference

`#include <IncrAlgoClasses.hpp>`

Inheritance diagram for minerule::NodeRow:



Public Member Functions

- NodeRow ()
- NodeRow (const NodeRow &nr)
- ~NodeRow ()
- double getSupp ()
- void setSupp (double &s)
- void incremSupp ()
- void decremSupp ()
- Head * getChild ()
- Head * setChild ()
- void mark ()
- void demark ()
- bool isMarked ()

6.93.1 Detailed Description

Definition at line 98 of file IncrAlgoClasses.hpp.

6.93.2 Constructor & Destructor Documentation

6.93.2.1 NodeRow() [1/2]

`minerule::NodeRow::NodeRow ( ) [inline]`

Definition at line 107 of file IncrAlgoClasses.hpp.

```
00107 : child(NULL),support(0) {marked=0;}
```

6.93.2.2 NodeRow() [2/2]

```
minerule::NodeRow::NodeRow (
            const NodeRow & nr )  [inline]
```

Definition at line 108 of file IncrAlgoClasses.hpp.

```
00108 : support(nr.support) {}
```

6.93.2.3 ~NodeRow()

```
minerule::NodeRow::~NodeRow ( )  [inline]
```

Definition at line 109 of file IncrAlgoClasses.hpp.

```
00109 {delete child;}
```

6.93.3 Member Function Documentation

6.93.3.1 decremSupp()

```
void minerule::NodeRow::decremSupp ( )  [inline]
```

Definition at line 114 of file IncrAlgoClasses.hpp.

```
00114 {support--;/*cout«"ora il supp e' "«support«"  ";*/}
```

6.93.3.2 demark()

```
void minerule::NodeRow::demark ( )  [inline]
```

Definition at line 118 of file IncrAlgoClasses.hpp.

```
00118 {marked=0;}
```

6.93.3.3 getChild()

```
Head * minerule::NodeRow::getChild ( )  [inline]
```

Definition at line 115 of file IncrAlgoClasses.hpp.

```
00115 {return child;}
```

6.93.3.4 getSupp()

```
double minerule::NodeRow::getSupp ( )  [inline]
```

Definition at line 111 of file IncrAlgoClasses.hpp.

```
00111 {return support;}
```

6.93.3.5 incremSupp()

```
void minerule::NodeRow::incremSupp ( )  [inline]
```

Definition at line 113 of file IncrAlgoClasses.hpp.

```
00113 {support++;/*cout«"ora il supp e' "«support«"  ";*/}
```

6.93.3.6 isMarked()

```
bool minerule::NodeRow::isMarked ( )  [inline]
```

Definition at line 119 of file IncrAlgoClasses.hpp.

```
00119 {return marked;}
```

6.93.3.7 mark()

```
void minerule::NodeRow::mark ( )  [inline]
```

Definition at line 117 of file IncrAlgoClasses.hpp.

```
00117 {marked=1;}
```

6.93.3.8 setChild()

```
Head * minerule::NodeRow::setChild ( )  [inline]
```

Definition at line 116 of file IncrAlgoClasses.hpp.

```
00116 {child = new Head(); return child;}
```

6.93.3.9 setSupp()

```
void minerule::NodeRow::setSupp (
            double & s )  [inline]
```

Definition at line 112 of file IncrAlgoClasses.hpp.

```
00112 {support=s;/*cout«"ho fissato il supp a "«support«"  ";*/}
```

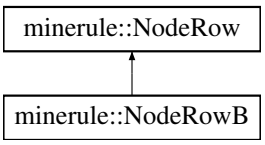
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp

6.94 minerule::NodeRowB Class Reference

`#include <IncrAlgoClasses.hpp>`

Inheritance diagram for minerule::NodeRowB:



Public Member Functions

- NodeRowB ()
- NodeRowB (const NodeRowB &nr)
- ~NodeRowB ()
- Body * getChild ()
- Body * setChild ()
- Head * setHead ()
- Head * getHead ()
- double getSupp ()
- void setSupp (double &s)
- void incremSupp ()
- void decremSupp ()
- void mark ()
- void demark ()
- bool isMarked ()

6.94.1 Detailed Description

Definition at line 131 of file IncrAlgoClasses.hpp.

6.94.2 Constructor & Destructor Documentation

6.94.2.1 NodeRowB() [1/2]

`minerule::NodeRowB::NodeRowB ( ) [inline]`

Definition at line 137 of file IncrAlgoClasses.hpp.

```
00137 : head(NULL),child(NULL){ }
```

6.94.2.2 NodeRowB() [2/2]

```
minerule::NodeRowB::NodeRowB (
            const NodeRowB & nr )  [inline]
```

Definition at line 138 of file IncrAlgoClasses.hpp.
```
00138 {}
```

6.94.2.3 ~NodeRowB()

```
minerule::NodeRowB::~NodeRowB ( )  [inline]
```

Definition at line 139 of file IncrAlgoClasses.hpp.
```
00139 {delete head; delete child;}
```

6.94.3 Member Function Documentation

6.94.3.1 decremSupp()

```
void minerule::NodeRow::decremSupp ( )  [inline], [inherited]
```

Definition at line 114 of file IncrAlgoClasses.hpp.
```
00114 {support--;/*cout«"ora il supp e' "«support«"  ";*/}
```

6.94.3.2 demark()

```
void minerule::NodeRow::demark ( )  [inline], [inherited]
```

Definition at line 118 of file IncrAlgoClasses.hpp.
```
00118 {marked=0;}
```

6.94.3.3 getChild()

```
Body * minerule::NodeRowB::getChild ( )  [inline]
```

Definition at line 140 of file IncrAlgoClasses.hpp.
```
00140 {return child;}
```

6.94.3.4 getHead()

```
Head * minerule::NodeRowB::getHead ( )  [inline]
```

Definition at line 146 of file IncrAlgoClasses.hpp.

```
00146 {return head;}
```

6.94.3.5 getSupp()

```
double minerule::NodeRow::getSupp ( )  [inline], [inherited]
```

Definition at line 111 of file IncrAlgoClasses.hpp.

```
00111 {return support;}
```

6.94.3.6 incremSupp()

```
void minerule::NodeRow::incremSupp ( )  [inline], [inherited]
```

Definition at line 113 of file IncrAlgoClasses.hpp.

```
00113 {support++;/*cout«"ora il supp e' "«support«"  ";*/}
```

6.94.3.7 isMarked()

```
bool minerule::NodeRow::isMarked ( )  [inline], [inherited]
```

Definition at line 119 of file IncrAlgoClasses.hpp.

```
00119 {return marked;}
```

6.94.3.8 mark()

```
void minerule::NodeRow::mark ( )  [inline], [inherited]
```

Definition at line 117 of file IncrAlgoClasses.hpp.

```
00117 {marked=1;}
```

6.94.3.9 setChild()

```
Body * minerule::NodeRowB::setChild ( )  [inline]
```

Definition at line 141 of file IncrAlgoClasses.hpp.

```
00141 {child = new Body(); return child;}
```

6.94.3.10 setHead()

```
Head * minerule::NodeRowB::setHead ( )  [inline]
```

Definition at line 142 of file IncrAlgoClasses.hpp.

```
00142                   {
00143     if (head==NULL)
00144       head = new Head();
00145     return head;}
```

6.94.3.11 setSupp()

```
void minerule::NodeRow::setSupp (
             double & s )  [inline], [inherited]
```

Definition at line 112 of file IncrAlgoClasses.hpp.

```
00112 {support=s;/*cout«"ho fissato il supp a "«support«"  ";*/}
```

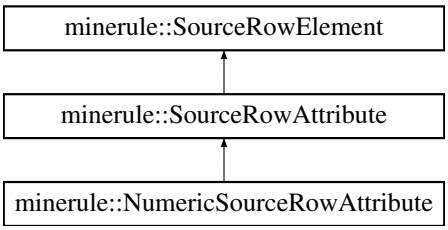
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrAlgoClasses.hpp

6.95 minerule::NumericSourceRowAttribute Class Reference

```
#include <SourceRowAttribute.hpp>
```

Inheritance diagram for minerule::NumericSourceRowAttribute:



Public Types

- typedef char ElementType

Public Member Functions

- NumericSourceRowAttribute (mrdb::ResultSet *_rs, int _elem)
- NumericSourceRowAttribute ()
- NumericSourceRowAttribute (const NumericSourceRowAttribute &rhs)
- virtual ~NumericSourceRowAttribute ()
- virtual SourceRowElement * copy () const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const
- virtual mrdb::Types::SQLType getType () const
- virtual void setValue (mrdb::ResultSet *, int)
- virtual void setValue (const std::string &)
- virtual int compareTo (const SourceRowAttribute &) const
- virtual std::string asString (const std::string &sep=",") const
- virtual std::string getSQLData () const
- virtual SourceRowElement & operator= (const SourceRowElement &_rhs)
- virtual ElementType getElementType () const
- virtual void serialize (std::ostream &os) const
- virtual void deserialize (std::istream &is)
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &s1) const
- virtual bool empty () const

  return true if this attribute is empty
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual std::string getFullElementType () const

Static Public Member Functions

- static SourceRowAttribute * createAttribute (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, int elem)
- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

Static Public Attributes

- static const SourceRowAttribute::ElementType elementType

Friends

- std::ostream & operator<< (std::ostream &os, const SourceRowAttribute &ia)

6.95.1 Detailed Description

This class represents int valued source row attributes. Probably "numeric" is a misnomer since it hints that the attribute can handle also real numbers. It can't, only integers are supported.

Definition at line 253 of file SourceRowAttribute.hpp.

6.95.2 Member Typedef Documentation

6.95.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.95.3 Constructor & Destructor Documentation

6.95.3.1 NumericSourceRowAttribute() [1/3]

```
minerule::NumericSourceRowAttribute::NumericSourceRowAttribute (
            mrdb::ResultSet * _rs,
            int _elem )  [inline]
```

Definition at line 263 of file SourceRowAttribute.hpp.

```
00264       : value(_rs->getInt(_elem)) {}
```

6.95.3.2 NumericSourceRowAttribute() [2/3]

```
minerule::NumericSourceRowAttribute::NumericSourceRowAttribute ( )  [inline]
```

Definition at line 266 of file SourceRowAttribute.hpp.

```
00266 : value(0){};
```

6.95.3.3 NumericSourceRowAttribute() [3/3]

```
minerule::NumericSourceRowAttribute::NumericSourceRowAttribute (
            const NumericSourceRowAttribute & rhs )  [inline]
```

Definition at line 269 of file SourceRowAttribute.hpp.

```
00270       : value(rhs.value) {}
```

6.95.3.4 ~NumericSourceRowAttribute()

```
virtual minerule::NumericSourceRowAttribute::~NumericSourceRowAttribute ( )  [inline], [virtual]
```

Definition at line 272 of file SourceRowAttribute.hpp.

```
00272                                       {
00273     //      std::cout « "Numeric... destructor" « std::endl;
00274   }
```

6.95.4 Member Function Documentation

6.95.4.1 asString()

```
std::string minerule::NumericSourceRowAttribute::asString (
            const std::string & sep = "," ) const  [virtual]
```

Convert this attribute to a string.

Parameters

sep	a separator string. Composite attributes should use this value to separate fields.

Returns

the std::string representation of the value of this attribute.

Implements minerule::SourceRowAttribute.

Definition at line 184 of file SourceRowAttribute.cpp.

```
00184                                                                    {
00185                         std::stringstream ss;
00186                         ss « value;
00187                         return ss.str();
00188                 }
```

6.95.4.2 compareTo()

```
int minerule::NumericSourceRowAttribute::compareTo (
            const SourceRowAttribute &  ) const  [virtual]
```

Compares the present attribute value with the one of the given attribute.

Returns

-1 if the present attribute is less than the argument, 0 if they are equal +1 otherwise

Implements minerule::SourceRowAttribute.

Definition at line 178 of file SourceRowAttribute.cpp.

```
00178                                                                    {
00179                         long int res= value - dynamic_cast<const
      NumericSourceRowAttribute&>(rhs).value;
00180                         return res;
00181                 }
```

6.95.4.3 copy()

```
virtual SourceRowElement * minerule::NumericSourceRowAttribute::copy ( ) const  [inline], [virtual]
```

Returns

a fresh allocated copy of the current object

Implements minerule::SourceRowAttribute.

Definition at line 278 of file SourceRowAttribute.hpp.

```
00278                                           {
00279     SourceRowAttribute *newRow = new NumericSourceRowAttribute(*this);
00280     return newRow;
00281   }
```

6.95.4.4 createAttribute()

```
SourceRowAttribute * minerule::SourceRowAttribute::createAttribute (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            int elem )  [static], [inherited]
```

Factory method.

Returns

an attribute capable of handling types of the given type

Factory method - returns an attribute capable of handling types of the given type

Definition at line 34 of file SourceRowAttribute.cpp.

```
00037                   {
00038                 mrdb::Types::SQLType colType = (mrdb::Types::SQLType) rsmd->getColumnType(elem);
00039 #ifdef DEBUG
00040 #warning MemDebugGeneric...
00041                 return new MemDebugGenericSourceRowAttribute(rs,elem,colType);
00042 #else
00043                 if(colType==mrdb::Types::INTEGER || colType==mrdb::Types::BIGINT)
00044                         return new NumericSourceRowAttribute(rs,elem);
00045                 else
00046                         return new GenericSourceRowAttribute(rs,elem,colType);
00047 #endif
00048         }
```

6.95.4.5 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.95.4.6 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.95.4.7 deserialize()

```
void minerule::NumericSourceRowAttribute::deserialize (
            std::istream & is )  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 207 of file SourceRowAttribute.cpp.

```
00207                                                                           {
00208                         std::string val;
00209                         is » val;
00210                         try {
00211                                 value = Converter(val).toLong();
00212                         } catch (MineruleException& e) {
00213                                 throw MineruleException( MR_ERROR_INTERNAL,
00214                                         "Cannot deserialize the requested NumericSourceRowAttribute,"
00215                                                 "the reason is:" + std::string(e.what()) );
00216                         }
00217                 }
```

6.95.4.8 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                   {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
```

6.95.4.9 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                        {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream>>elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
```

6.95.4.10 empty()

```
virtual bool minerule::SourceRowAttribute::empty ( ) const  [inline], [virtual], [inherited]
```

return true if this attribute is empty

Implements minerule::SourceRowElement.

Definition at line 113 of file SourceRowAttribute.hpp.

```
00113 { return false; }
```

6.95.4.11 getElementType()

```
virtual ElementType minerule::NumericSourceRowAttribute::getElementType ( ) const  [inline],
[virtual]
```

Serialization methods

Reimplemented from minerule::SourceRowElement.

Definition at line 308 of file SourceRowAttribute.hpp.

```
00308 { return 'n'; }
```

6.95.4.12 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                           {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.95.4.13 getSQLData()

```
std::string minerule::NumericSourceRowAttribute::getSQLData ( ) const  [virtual]
```

Returns

the std::string representation for the attribute in a format suitable to be used in SQL queries (tipically it is implemented as return "'"+this->asString()+"'"

Implements minerule::SourceRowAttribute.

Definition at line 191 of file SourceRowAttribute.cpp.

```
00191                                                       {
00192                         std::stringstream ss;
00193                         ss « "'" « value « "'";
00194                         return ss.str();
00195                 }
```

6.95.4.14 getType()

```
virtual mrdb::Types::SQLType minerule::NumericSourceRowAttribute::getType ( ) const  [inline],
[virtual]
```

Returns

the type of the attribute

Implements minerule::SourceRowAttribute.

Definition at line 288 of file SourceRowAttribute.hpp.

```
00288 { return mrdb::Types::INTEGER; }
```

6.95.4.15 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                       {
00054     return !this->operator==(el);
00055   }
```

6.95.4.16 operator()()

```
virtual bool minerule::SourceRowAttribute::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [inline], [virtual], [inherited]
```

Returns

true iff s1<s2

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.hpp.

```
00098                                                       {
00099     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00100     const SourceRowAttribute &attr2 = dynamic_cast<const SourceRowAttribute &>(s2);
00101
00102     return attr1.compareTo(attr2) < 0;
00103   }
```

6.95.4.17 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                  {
00058     return this->operator()(*this, el);
00059   }
```

6.95.4.18 operator<<()

```
virtual std::ostream & minerule::SourceRowAttribute::operator<< (
            std::ostream & os ) const  [inline], [virtual], [inherited]
```

Inserts the value of this attribute in the provided ostream and returns the ostream.

Parameters

os	an output stream

Returns

the output stream given in input

Implements minerule::SourceRowElement.

Definition at line 129 of file SourceRowAttribute.hpp.

```
00129                                                  {
00130     os << asString();
00131     return os;
00132   }
```

6.95.4.19 operator=()

```
virtual SourceRowElement & minerule::NumericSourceRowAttribute::operator= (
            const SourceRowElement & _rhs )  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 298 of file SourceRowAttribute.hpp.

```
00298                                                                    {
00299     const NumericSourceRowAttribute &rhs =
00300         dynamic_cast<const NumericSourceRowAttribute &>(_rhs);
00301     value = rhs.value;
00302     return *this;
00303   }
```

6.95.4.20 operator==()

```
virtual bool minerule::SourceRowAttribute::operator== (
            const SourceRowElement & s1 ) const  [inline], [virtual], [inherited]
```

Returns

true iff *this == s1

Implements minerule::SourceRowElement.

Definition at line 106 of file SourceRowAttribute.hpp.

```
00106                                                          {
00107     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00108
00109     return this->compareTo(attr1) == 0;
00110   }
```

6.95.4.21 serialize()

```
void minerule::NumericSourceRowAttribute::serialize (
            std::ostream & os ) const  [virtual]
```

Serialization methods

Implements minerule::SourceRowElement.

Definition at line 203 of file SourceRowAttribute.cpp.

```
00203                                                                    {
00204                         os « " " « value « " ";
00205                 }
```

6.95.4.22 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
        {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.95.4.23 setPreparedStatementParameters()

```
virtual void minerule::NumericSourceRowAttribute::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 283 of file SourceRowAttribute.hpp.

```
00284                                                                    {
00285     state->setLong(start_index, value);
00286   }
```

6.95.4.24 setValue() [1/2]

```
void minerule::NumericSourceRowAttribute::setValue (
            const std::string & value )  [virtual]
```

Sets the value of the attribute according to the given string (converting the value to the proper type when necessary).

Parameters

value	a string representation of the value to be set

Implements minerule::SourceRowAttribute.

Definition at line 173 of file SourceRowAttribute.cpp.

```
00173                                                                {
00174                         value = (long int) strtod(inStr.c_str(), NULL);
00175                 }
```

6.95.4.25 setValue() [2/2]

```
void minerule::NumericSourceRowAttribute::setValue (
            mrdb::ResultSet * rs,
            int col )  [virtual]
```

Sets the value of the attribute accordingly to the value of the specified column of the current row of the given result set

Parameters

rs	the result set source of the value
col	the index (starting from 1) of the column in the result set

Implements minerule::SourceRowAttribute.

Definition at line 164 of file SourceRowAttribute.cpp.

```
00164                                                                                {
00165                         assert(rs!=NULL);
00166                         assert(col<=rs->getMetaData()->getColumnCount());
00167                         assert( getType() == (mrdb::Types::SQLType)
       rs->getMetaData()->getColumnType(col) );
00168
00169                         value = rs->getLong(col);
00170                 }
```

6.95.5 Friends And Related Function Documentation

6.95.5.1 operator<<

```
std::ostream & operator<< (
             std::ostream & os,
             const SourceRowAttribute & ia )  [friend]
```

Definition at line 313 of file SourceRowAttribute.hpp.

```
00314                                                                    {
00315   attr.operator«(os);
00316   return os;
00317 }
```

6.95.6 Field Documentation

6.95.6.1 elementType

```
const SourceRowAttribute::ElementType minerule::NumericSourceRowAttribute::elementType  [static]
```

Definition at line 261 of file SourceRowAttribute.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp

6.96 minerule::OptimizedMinerule::OptimizationInfo Class Reference

`#include <OptimizedMinerule.hpp>`

Data Fields

- MineruleRelationship relationship
- ParsedMinerule minerule
- std::vector< ParsedMinerule > minerulesToCombine
- GAQueryCombinator::QueryOrList combinationFormula

6.96.1 Detailed Description

Definition at line 38 of file OptimizedMinerule.hpp.

6.96.2 Field Documentation

6.96.2.1 combinationFormula

`GAQueryCombinator::QueryOrList minerule::OptimizedMinerule::OptimizationInfo::combination↩Formula`

Definition at line 47 of file OptimizedMinerule.hpp.

6.96.2.2 minerule

`ParsedMinerule minerule::OptimizedMinerule::OptimizationInfo::minerule`

Definition at line 41 of file OptimizedMinerule.hpp.

6.96.2.3 minerulesToCombine

`std::vector<ParsedMinerule> minerule::OptimizedMinerule::OptimizationInfo::minerulesToCombine`

Definition at line 45 of file OptimizedMinerule.hpp.

6.96.2.4 relationship

```
MineruleRelationship minerule::OptimizedMinerule::OptimizationInfo::relationship
```

Definition at line 40 of file OptimizedMinerule.hpp.

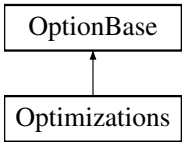
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizedMinerule.hpp

6.97 Optimizations Class Reference

```
#include <optimizations.hpp>
```

Inheritance diagram for Optimizations:



Data Structures

- class Combinator

Public Types

- enum PreferredIncrementalAlgorithm { ConstructiveAlgo , DestructiveAlgo , AutochooseIncrAlgo }

Public Member Functions

- Optimizations ()
- virtual ~Optimizations ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- virtual OptionBase & subclassForName (const std::string &subclassName)
- Combinator & getCombinator ()
- const Combinator & getCombinator () const
- bool getTryOptimizations () const
- void setTryOptimizations (bool val)
- bool getAvoidEquivalenceDetection () const
- void setAvoidEquivalenceDetection (bool val)
- bool getAvoidCombinationDetection () const
- void setAvoidCombinationDetection (bool val)
- bool getAvoidDominanceDetection () const
- void setAvoidDominanceDetection (bool val)
- PreferredIncrementalAlgorithm getIncrementalAlgorithm () const
- void setIncrementalAlgorithm (PreferredIncrementalAlgorithm p)
- OptimizerCatalogue & getCatalogue ()

6.97.1 Detailed Description

Definition at line 16 of file optimizations.hpp.

6.97.2 Member Enumeration Documentation

6.97.2.1 PreferredIncrementalAlgorithm

```
enum Optimizations::PreferredIncrementalAlgorithm
```

Enumerator

ConstructiveAlgo	
DestructiveAlgo	
AutochooseIncrAlgo	

Definition at line 18 of file optimizations.hpp.

```
00018                 {
00019         ConstructiveAlgo,
00020         DestructiveAlgo,
00021         AutochooseIncrAlgo
00022   } PreferredIncrementalAlgorithm;
```

6.97.3 Constructor & Destructor Documentation

6.97.3.1 Optimizations()

```
Optimizations::Optimizations ( )  [inline]
```

Definition at line 66 of file optimizations.hpp.

```
00066                     :
00067         catalogue(NULL),
00068         preferredIncrementalAlgorithm(AutochooseIncrAlgo),
00069         avoidDominanceDetection(false),
00070         avoidEquivalenceDetection(false),
00071         avoidCombinationDetection(false) {};
```

6.97.3.2 ~Optimizations()

```
virtual Optimizations::~Optimizations ( )  [inline], [virtual]
```

Definition at line 73 of file optimizations.hpp.

```
00073                           {
00074         if( catalogue!=NULL )
00075           delete catalogue;
00076   };
```

6.97.4 Member Function Documentation

6.97.4.1 className()

```
virtual std::string Optimizations::className ( ) const  [inline], [virtual]
```

Definition at line 78 of file optimizations.hpp.

```
00078                                       {
00079         return "optimizations";
00080     }
```

6.97.4.2 getAvoidCombinationDetection()

```
bool Optimizations::getAvoidCombinationDetection ( ) const  [inline]
```

Definition at line 113 of file optimizations.hpp.

```
00113                                                {
00114         return avoidCombinationDetection;
00115     }
```

6.97.4.3 getAvoidDominanceDetection()

```
bool Optimizations::getAvoidDominanceDetection ( ) const  [inline]
```

Definition at line 121 of file optimizations.hpp.

```
00121                                               {
00122         return avoidDominanceDetection;
00123     }
```

6.97.4.4 getAvoidEquivalenceDetection()

```
bool Optimizations::getAvoidEquivalenceDetection ( ) const  [inline]
```

Definition at line 105 of file optimizations.hpp.

```
00105                                                {
00106         return avoidEquivalenceDetection;
00107     }
```

6.97.4.5 getCatalogue()

```
OptimizerCatalogue & Optimizations::getCatalogue ( )  [inline]
```

Definition at line 138 of file optimizations.hpp.

```
00138                                       {
00139         if( catalogue==NULL ) {
00140           catalogue = new OptimizerCatalogue();
00141         }
00142
00143         return *catalogue;
00144   }
```

6.97.4.6 getCombinator() [1/2]

```
Combinator & Optimizations::getCombinator ( )  [inline]
```

Definition at line 94 of file optimizations.hpp.

```
00094 { return combinator; }
```

6.97.4.7 getCombinator() [2/2]

```
const Combinator & Optimizations::getCombinator ( ) const  [inline]
```

Definition at line 95 of file optimizations.hpp.

```
00095 { return combinator; }
```

6.97.4.8 getIncrementalAlgorithm()

```
PreferredIncrementalAlgorithm Optimizations::getIncrementalAlgorithm ( ) const  [inline]
```

Definition at line 130 of file optimizations.hpp.

```
00130                                       {
00131         return preferredIncrementalAlgorithm;
00132   }
```

6.97.4.9 getTryOptimizations()

```
bool Optimizations::getTryOptimizations ( ) const  [inline]
```

Definition at line 97 of file optimizations.hpp.

```
00097                                       {
00098         return tryOptimizations;
00099   }
```

6.97.4.10 setAvoidCombinationDetection()

```
void Optimizations::setAvoidCombinationDetection (
            bool val )  [inline]
```

Definition at line 117 of file optimizations.hpp.

```
00117                                                {
00118         avoidCombinationDetection=val;
00119   }
```

6.97.4.11 setAvoidDominanceDetection()

```
void Optimizations::setAvoidDominanceDetection (
            bool val )  [inline]
```

Definition at line 125 of file optimizations.hpp.

```
00125                                               {
00126         avoidDominanceDetection=val;
00127   }
```

6.97.4.12 setAvoidEquivalenceDetection()

```
void Optimizations::setAvoidEquivalenceDetection (
            bool val )  [inline]
```

Definition at line 109 of file optimizations.hpp.

```
00109                                                {
00110         avoidEquivalenceDetection=val;
00111   }
```

6.97.4.13 setIncrementalAlgorithm()

```
void Optimizations::setIncrementalAlgorithm (
            PreferredIncrementalAlgorithm p )  [inline]
```

Definition at line 134 of file optimizations.hpp.

```
00134                                                                    {
00135         preferredIncrementalAlgorithm=p;
00136   }
```

6.97.4.14 setOption()

```
virtual void Optimizations::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.97.4.15 setTryOptimizations()

```
void Optimizations::setTryOptimizations (
            bool val )  [inline]
```

Definition at line 101 of file optimizations.hpp.

```
00101                                    {
00102       tryOptimizations=val;
00103   }
```

6.97.4.16 subclassForName()

```
virtual OptionBase & Optimizations::subclassForName (
            const std::string & subclassName )  [inline], [virtual]
```

Definition at line 85 of file optimizations.hpp.

```
00086   {
00087         if( subclassName=="combinator" ) {
00088           return combinator;
00089         } else {
00090           return OptionBase::subclassForName(subclassName);
00091         }
00092   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/optimizations.hpp

6.98 minerule::OptimizedMinerule Class Reference

```
#include <OptimizedMinerule.hpp>
```

Data Structures

- class OptimizationInfo

Public Types

- enum MineruleRelationship {
  None , Dominance , Inclusion , Equivalence ,
  Combination }

Public Member Functions

- OptimizedMinerule (const std::string &minerule_text)
- void optimize ()
- const ParsedMinerule & getParsedMinerule () const
- const OptimizationInfo & getOptimizationInfo () const
- OptimizationInfo & getOptimizationInfo ()
- bool hasIDConstraints () const

Static Public Member Functions

- static bool isACandidateQuery (const ParsedMinerule &candidate, const ParsedMinerule &target)
- static bool firstMineruleIncludesSecondMinerule (const ParsedMinerule &, const ParsedMinerule &)
- static bool firstMineruleDominatesSecondMinerule (const ParsedMinerule &, const ParsedMinerule &)
- static MineruleRelationship getMineruleRelationship (const ParsedMinerule &, const ParsedMinerule &)

Protected Member Functions

- void dropTableIfExists (mrdb::Connection *, const std::string &tname) const
- MineruleRelationship getRelationshipType ()
- void checkForCombinedQueries ()

Static Protected Member Functions

- static void attributesInPredicate (const list_OR_node *l, std::set< std::string > &result)
- static bool predicateAttributesAreIncluded (const ParsedMinerule &mr1, const ParsedMinerule &mr2)

6.98.1 Detailed Description

Definition at line 27 of file OptimizedMinerule.hpp.

6.98.2 Member Enumeration Documentation

6.98.2.1 MineruleRelationship

```
enum minerule::OptimizedMinerule::MineruleRelationship
```

Enumerator

None	
Dominance	
Inclusion	
Equivalence	
Combination	

Definition at line 29 of file OptimizedMinerule.hpp.

```
00029                 {
00030       None,             // No relationship can be found
00031       Dominance,        // There exists a query which dominate the current one
00032       Inclusion,        // There exists a query which include the current one
00033       Equivalence,      // There exists a query equivalent to the current one
00034       Combination       // There exists a set of queries that can be combined
00035                         // into one equivalent to the current one
00036     } MineruleRelationship;
```

6.98.3 Constructor & Destructor Documentation

6.98.3.1 OptimizedMinerule()

```
minerule::OptimizedMinerule::OptimizedMinerule (
            const std::string & minerule_text )  [inline]
```

Definition at line 67 of file OptimizedMinerule.hpp.

```
00067                                                   {
00068       minerule.init(minerule_text);
00069       optInfo.relationship = None;
00070     }
```

6.98.4 Member Function Documentation

6.98.4.1 attributesInPredicate()

```
void minerule::OptimizedMinerule::attributesInPredicate (
            const list_OR_node * l,
            std::set< std::string > & result )  [static], [protected]
```

Definition at line 48 of file OptimizedMinerule.cpp.

```
00048
        {
00049                 while( l!=NULL ) {
00050                         list_AND_node* l_and = l->l_and;
00051                         while( l_and!=NULL ) {
00052                                 if( !Converter(l_and->sp->val1).isNumber() )
00053                                         result.insert(l_and->sp->val1);
00054
00055                                 if( !Converter(l_and->sp->val2).isNumber() )
00056                                         result.insert(l_and->sp->val2);
00057
00058                                 l_and=l_and->next;
00059                         }
00060
00061                         l=l->next;
00062                 }
00063         }
```

6.98.4.2 checkForCombinedQueries()

```
void minerule::OptimizedMinerule::checkForCombinedQueries ( )  [protected]
```

Definition at line 100 of file OptimizedMinerule.cpp.

```
00100                                                   {
00101                 MRLogPush("Searching for equivalences due to query combinations...");
00102                 std::vector<Predicate> candidates;
00103
00104                 std::vector<ParsedMinerule>::const_iterator it;
00105
00106     // disabling the logging of the Parser since it would confuse
00107     // the user.
00108                 MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00109
```

```
00110                 std::ostream& log = MRLog() « "Candidates are: ";
00111
00112                 for(it=optInfo.minerulesToCombine.begin(); it!=optInfo.minerulesToCombine.end(); it++)
      {
00113                         if(it!=optInfo.minerulesToCombine.begin()) log«", ";
00114
00115                         log « it->tab_result;
00116                         candidates.push_back(Predicate(it->mc));
00117                 }
00118                 log « std::endl;
00119
00120     // re-enabling the log
00121                 MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00122
00123                 Predicate target(minerule.mc);
00124
00125         // FIXME: The third parameters ("Sales") seems to be obsoleted and not used any more
00126                 GAQueryCombinator qCombinator(target, candidates);
00127                 MineruleOptions::Optimizations::Combinator& combopt =
00128                         MineruleOptions::getSharedOptions().getOptimizations().getCombinator();
00129
00130                 qCombinator.setMaxDisjuncts(combopt.getMaxDisjuncts());
00131                 qCombinator.setMaxQueries(combopt.getMaxQueries());
00132                 qCombinator.setMaxDistinctPredicates(combopt.getMaxDistinctPredicates());
00133                 qCombinator.setTimeOutThreshold(combopt.getTimeOutThreshold());
00134
00135                 try {
00136                         qCombinator.evolve();
00137                 } catch( TimeOutException& e ) {
00138                         MRLog() « "The algorithm timed out." « std::endl;
00139                 }
00140
00141
00142         // FIXME: all of the following is about output. It should be moved into an appropriate method
00143                 if( qCombinator.getResult().empty() ) {
00144                         MRLog() « "No valid combination found!" « std::endl;
00145                 } else {
00146                         std::string qryString;
00147
00148                         GAQueryCombinator::QueryOrList::const_iterator itOr;
00149                         for(itOr=qCombinator.getResult().begin(); itOr!=qCombinator.getResult().end();
       itOr++) {
00150                                 if(itOr!=qCombinator.getResult().begin())
00151                                         qryString += " OR ";
00152
00153                                 GAQueryCombinator::QueryAndList::const_iterator itAnd;
00154                                 for(itAnd=itOr->begin(); itAnd!=itOr->end(); itAnd++) {
00155                                         if( itAnd!=itOr->begin() )
00156                                                 qryString += " AND ";
00157
00158                                         qryString += optInfo.minerulesToCombine[*itAnd].tab_result;
00159                                 }
00160                         }
00161
00162
00163                         MRLog()    « "A combination of queries equivalent to the given one"
00164                                 « " has been found. The found query follows:" « std::endl;
00165                         MRLog()    « qryString « std::endl;
00166
00167                         optInfo.combinationFormula = qCombinator.getResult();
00168                         optInfo.relationship = Combination;
00169                 }
00170
00171                 MRLogPop();
00172         }
```

6.98.4.3 dropTableIfExists()

```
void minerule::OptimizedMinerule::dropTableIfExists (
            mrdb::Connection * ,
            const std::string & tname ) const  [protected]
```

6.98.4.4 firstMineruleDominatesSecondMinerule()

```
bool minerule::OptimizedMinerule::firstMineruleDominatesSecondMinerule (
            const ParsedMinerule & mr1,
            const ParsedMinerule & mr2 )  [static]
```

Definition at line 316 of file OptimizedMinerule.cpp.

```
00316
                    {
00317
00318
00319                 Predicate mr1pmc(mr1.mc);
00320                 Predicate mr2pmc(mr2.mc);
00321                 Predicate mr1pgc(mr1.gc);
00322                 Predicate mr2pgc(mr2.gc);
00323                 Predicate mr1pcc(mr1.cc);
00324                 Predicate mr2pcc(mr2.cc);
00325
00326                 PredicateUtils::PredicateRelationship mcrel =
00327                         PredicateUtils::getPredicateRelationship( mr1pmc, mr2pmc, mr1.tab_source );
00328                 PredicateUtils::PredicateRelationship gcrel =
00329                         PredicateUtils::getPredicateRelationship( mr1pgc, mr2pgc, mr1.tab_source );
00330                 PredicateUtils::PredicateRelationship ccrel =
00331                         PredicateUtils::getPredicateRelationship( mr1pcc, mr2pcc, mr1.tab_source );
00332
00333
00334                 return
00335                         mr1.ga==mr2.ga &&
00336                         mr1.ca==mr2.ca &&
00337                         mr1.ra==mr2.ra &&
00338                         mr1.ba==mr2.ba &&
00339                         mr1.ha==mr2.ha &&
00340                         mr1.c_aggr_list == mr2.c_aggr_list &&
00341                         mr1.sup<=mr2.sup &&
00342                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00343                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00344                         mr1.tab_source==mr2.tab_source &&
00345                         (mcrel==EncodedNF::FirstMoreGeneral || mcrel==EncodedNF::Equivalent)&&
00346                         (gcrel==EncodedNF::FirstMoreGeneral || gcrel==EncodedNF::Equivalent)&&
00347                         (ccrel==EncodedNF::FirstMoreGeneral || ccrel==EncodedNF::Equivalent);
00348         }
```

6.98.4.5 firstMineruleIncludesSecondMinerule()

```
bool minerule::OptimizedMinerule::firstMineruleIncludesSecondMinerule (
            const ParsedMinerule & mr1,
            const ParsedMinerule & mr2 )  [static]
```

Definition at line 291 of file OptimizedMinerule.cpp.

```
00291
                    {
00292                 Predicate mr1pmc(mr1.mc);
00293                 Predicate mr2pmc(mr2.mc);
00294                 Predicate mr1pgc(mr1.gc);
00295                 Predicate mr2pgc(mr2.gc);
00296                 Predicate mr1pcc(mr1.cc);
00297                 Predicate mr2pcc(mr2.cc);
00298
00299                 return
00300                         mr1.ga==mr2.ga &&
00301                         mr1.ca==mr2.ca &&
00302                         mr1.ra==mr2.ra &&
00303                         mr1.ba==mr2.ba &&
00304                         mr1.ha==mr2.ha &&
00305                         mr1.c_aggr_list == mr2.c_aggr_list &&
00306                         mr1.sup<=mr2.sup &&
00307                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00308                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00309                         mr1.tab_source==mr2.tab_source &&
00310                         PredicateUtils::predicatesAreEquivalent( mr1pmc, mr2pmc, mr1.tab_source ) &&
00311                         PredicateUtils::predicatesAreEquivalent( mr1pgc, mr2pgc, mr1.tab_source ) &&
00312                         PredicateUtils::predicatesAreEquivalent( mr1pcc, mr2pcc, mr1.tab_source );
00313         }
```

6.98.4.6 getMineruleRelationship()

```
OptimizedMinerule::MineruleRelationship minerule::OptimizedMinerule::getMineruleRelationship (
            const ParsedMinerule & mr1,
            const ParsedMinerule & mr2 )  [static]
```

Definition at line 351 of file OptimizedMinerule.cpp.

```
00351
      {
00352                 if(
      MineruleOptions::getSharedOptions().getOptimizations().getAvoidDominanceDetection() )
00353                         return None;
00354
00355                 if(!(mr1.ga==mr2.ga &&
00356                         mr1.ca==mr2.ca &&
00357                         mr1.ra==mr2.ra &&
00358                         mr1.ba==mr2.ba &&
00359                         mr1.ha==mr2.ha &&
00360                         mr1.c_aggr_list == mr2.c_aggr_list &&
00361                         mr1.sup<=mr2.sup &&
00362                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00363                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00364                         mr1.tab_source==mr2.tab_source))
00365                                 return None;
00366
00367                 Predicate mr1pmc(mr1.mc);
00368                 Predicate mr2pmc(mr2.mc);
00369                 Predicate mr1pgc(mr1.gc);
00370                 Predicate mr2pgc(mr2.gc);
00371                 Predicate mr1pcc(mr1.cc);
00372                 Predicate mr2pcc(mr2.cc);
00373
00374                 PredicateUtils::PredicateRelationship relations[3];
00375
00376                 relations[0] = PredicateUtils::getPredicateRelationship( mr1pmc, mr2pmc,
      mr1.tab_source );
00377                 relations[1] = PredicateUtils::getPredicateRelationship( mr1pgc, mr2pgc,
      mr1.tab_source );
00378                 relations[2] = PredicateUtils::getPredicateRelationship( mr1pcc, mr2pcc,
      mr1.tab_source );
00379
00380                 MineruleRelationship current = Equivalence;
00381
00382                 for(size_t i=0; i<3; i++) {
00383                         switch( relations[i] ) {
00384                                 case EncodedNF::FirstMoreGeneral:
00385                                         current=Dominance;
00386                                         break;
00387                                 case EncodedNF::Equivalent:
00388                                         // Nothing to do, just avoid to select the default
00389                                         // label
00390                                         break;
00391                                 default:
00392                                         return None;
00393                         };
00394                 }
00395
00396                 return current;
00397         }
```

6.98.4.7 getOptimizationInfo() [1/2]

```
OptimizationInfo & minerule::OptimizedMinerule::getOptimizationInfo ( )  [inline]
```

Definition at line 84 of file OptimizedMinerule.hpp.

```
00084                                   {
00085       return optInfo;
00086     }
```

6.98.4.8 getOptimizationInfo() [2/2]

```
const OptimizationInfo & minerule::OptimizedMinerule::getOptimizationInfo ( ) const  [inline]
```

Definition at line 79 of file OptimizedMinerule.hpp.

```
00079                                    {
00080       return optInfo;
00081     }
```

6.98.4.9 getParsedMinerule()

```
const ParsedMinerule & minerule::OptimizedMinerule::getParsedMinerule ( ) const  [inline]
```

Definition at line 74 of file OptimizedMinerule.hpp.

```
00074                                                {
00075       return minerule;
00076     }
```

6.98.4.10 getRelationshipType()

```
MineruleRelationship minerule::OptimizedMinerule::getRelationshipType ( )  [protected]
```

6.98.4.11 hasIDConstraints()

```
bool minerule::OptimizedMinerule::hasIDConstraints ( ) const
```

Definition at line 400 of file OptimizedMinerule.cpp.

```
00400                                                  {
00401                 return OptimizerCatalogue::hasIDConstraints(minerule);
00402         }
```

6.98.4.12 isACandidateQuery()

```
bool minerule::OptimizedMinerule::isACandidateQuery (
            const ParsedMinerule & candidate,
            const ParsedMinerule & target )  [static]
```

Definition at line 79 of file OptimizedMinerule.cpp.

```
00079
        {
00080                 return mr1.ga==mr2.ga &&
00081                         mr1.ca==mr2.ca &&
00082                         mr1.ra==mr2.ra &&
00083                         mr1.ba==mr2.ba &&
00084                         mr1.ha==mr2.ha &&
00085                         mr1.c_aggr_list == mr2.c_aggr_list &&
00086                         mr1.sup<=mr2.sup &&
00087                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00088                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00089                         mr1.tab_source==mr2.tab_source &&
00090                // above are the standard checking
00091                // if in addition both the queries are item dependent (mr1 is
00092                // item dependent by hypothesis).
00093                         OptimizerCatalogue::hasIDConstraints(mr1) &&
00094           // and the all the attributes on which mr1.mc predicates are included
00095           // in those on which mr2 predicates
00096                         predicateAttributesAreIncluded(mr1,mr2);
00097                   // then mr1 is a candidate query
00098         }
```

6.98.4.13 optimize()

```
void minerule::OptimizedMinerule::optimize ( )
```

Definition at line 175 of file OptimizedMinerule.cpp.

```
00176         {
00177                 if(minerule.miningTask!=MTMineRules) {
00178                         MRLog() « "Skipping optimization, " « miningTaskToString(minerule.miningTask)
      « " task is still unsupported by the optimizer" « std::endl;
00179                         MRDebug() « "Skipping optimization, " «
      miningTaskToString(minerule.miningTask) « " task is still unsupported by the optimizer" « std::endl;
00180                         return;
00181                 }
00182
00183                 std::list<ParsedMinerule> lPreviousMr;
00184                 if(!MineruleOptions::getSharedOptions().getOptimizations().getTryOptimizations())
00185                         return;
00186
00187                 bool avoidCombinationDetection =
00188
      MineruleOptions::getSharedOptions().getOptimizations().getAvoidCombinationDetection()
00189                                 || !hasIDConstraints();
00190
00191                 OptimizerCatalogue& catalogue =
00192                         MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00193
00194                 QueryNormalizer normalizer( catalogue, minerule );
00195                 normalizer.normalize();
00196                 optInfo.minerule = minerule;
00197                 optInfo.relationship = None;
00198                 MRDebug("Minerule text after normalization:" + minerule.getText());
00199
00200                 std::vector<CatalogueInfo> catInfos;
00201                 OptimizerCatalogue::getMRQueryInfos(catInfos);
00202
00203    //lettura delle minerules precedenti
00204                 try     {
00205                         MRLogPusher _("Determining if a relationship exists with previous
      minerules...");
00206
00207                         MRLog() « "Reading past minerules" « std::endl;
00208                         for(std::vector<CatalogueInfo>::const_iterator it=catInfos.begin();
      it!=catInfos.end(); ++it) {
00209                                 ParsedMinerule mr;
00210                                 mr.init(it->qryText);
00211                                 MRDebug("Optimize: trying minerule:" + mr.getText());
00212
00213                                 if( !avoidCombinationDetection && isACandidateQuery(mr,minerule) ) {
00214                                         optInfo.minerulesToCombine.push_back(mr);
00215                                 }
00216
00217                                 MineruleRelationship currentRel =
      getMineruleRelationship(mr,minerule);
00218
00219                                 if( currentRel==Equivalence &&
      MineruleOptions::getSharedOptions().getOptimizations().getAvoidEquivalenceDetection() )
00220                                                 currentRel=Dominance;
00221
00222                                 switch( currentRel ) {
00223                                         case Equivalence:
00224                                                 MRLog() « "A past minerule found which includes the
      current one!" « std::endl;
00225                                                 MRLog() « "Name of the found minerule:" « it->qryName
      « std::endl;
00226                                                 MRLog() « "(The algorithm will store this information
      in the catalogue" « std::endl;
00227                                                 MRLog() « " and exit)" « std::endl;
00228                                                 optInfo.relationship = Equivalence;
00229                                                 optInfo.minerule = mr;
00230                                                 break;
00231                                         case Dominance:
00232                                                 {
00233                                                         MRLog() « "Minerule named "" « mr.tab_result
      «"" dominates the current one." «std::endl;
00234                                                         if(optInfo.relationship==None) {
00235                         // since relationship==None, there is not yet any information
00236                         // about other Dominances. We simply store the found minerule
00237                                                                 optInfo.minerule = mr;
00238                                                                 MRLog() « "Minerule " « """ «
      mr.tab_result «"" is now the best promising"
00239                                                                         « " dominant minerule." «
      std::endl;
00240                                                         } else {
00241                                                                 // we must keep the current minerule
      only if its result set
```

```
00242                                                                   // is smaller than the one of the past
      Dominant query found
00243                                                                   CatalogueInfo oldQryInfo;
00244                                                                   OptimizerCatalogue::getMRQueryInfo(
      optInfo.minerule.tab_result, oldQryInfo );
00245                                                                   if( it->resSize<oldQryInfo.resSize) {
00246                                                                           optInfo.minerule = mr;
00247                                                                           MRLog() << "Minerule " << "`" <<
      mr.tab_result <<"'" is now the best promising"
00248                                                                                   << " dominant
      minerule." << std::endl;
00249                                                                   } else {
00250                                                                           MRLog() << "Keeping the
      previously found dominant minerule" << std::endl;
00251                                                                   }
00252                                                           }
00253
00254                                                           optInfo.relationship = Dominance;
00255                                                   }
00256                                                   break;
00257                                           default:
00258                                                   // nothing to do
00259                                                   break;
00260                                   };
00261
00262                             // if the found relationship is Equivalence, then we cannot do better
00263                             // and exit immediately from the loop, otherwise we check if we can
00264                             // find some minerule which is a better candidate for the incremental
00265                             // algorithms.
00266                                   if(optInfo.relationship==Equivalence)
00267                                           break;
00268                           }
00269
00270                           if(optInfo.relationship==None)
00271                                   MRLog() << "No interesting past minerules found." << std::endl;
00272
00273                           if(optInfo.relationship==Dominance)
00274                                   MRLog() << "The best promising dominant minerule found is `"
00275                                           << optInfo.minerule.tab_result << "'" << std::endl;
00276                   } catch (mrdb::SQLException& e) {
00277                           MRErr() << e.what() << std::endl;
00278                           throw;
00279                   }
00280
00281                   if(optInfo.relationship==None && !avoidCombinationDetection) {
00282                           checkForCombinedQueries();
00283                           if(optInfo.relationship==None)
00284                                   optInfo.minerulesToCombine.clear();
00285                   }
00286           }
```

6.98.4.14 predicateAttributesAreIncluded()

```
bool minerule::OptimizedMinerule::predicateAttributesAreIncluded (
            const ParsedMinerule & mr1,
            const ParsedMinerule & mr2 )  [static], [protected]
```

Definition at line 65 of file OptimizedMinerule.cpp.

```
00066                                   {
00067                   std::set<std::string> mr1Attrs;
00068                   std::set<std::string> mr2Attrs;
00069
00070                   attributesInPredicate( mr1.mc, mr1Attrs );
00071                   attributesInPredicate( mr2.mc, mr2Attrs );
00072
00073                   return includes(mr2Attrs.begin(), mr2Attrs.end(), mr1Attrs.begin(), mr1Attrs.end());
00074
00075           }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizedMinerule.hpp
- /Users/esposito/Software/minerule/src/Optimizer/OptimizedMinerule.cpp

6.99 minerule::OptimizerCatalogue Class Reference

`#include <OptimizerCatalogue.hpp>`

Data Structures

- class Catalogue
- class DepFunCatalogue
- class EqKeysCatalogue
- class MineruleResultInfo

Public Types

- enum CatalogueType { EQKEYS , DEPFUN }
- enum OrderType { Equal , Reversed , None }
- typedef std::set< std::string > KeyCols
- typedef std::multimap< KeyCols, std::pair< KeyCols, OrderType > > CatalogueEntry

Public Member Functions

- OptimizerCatalogue ()
- const Catalogue & getCatalogue (CatalogueType ct=EQKEYS) const
- bool isIDAttribute (const std::string &tableName, const std::vector< std::string > &itemCols, const std::string &attribute) const

Static Public Member Functions

- static bool hasIDConstraints (const ParsedMinerule &mineruel)
- static bool checkInstallation ()
- static void install ()
- static void uninstall ()
- static void addMineruleResult (const MineruleResultInfo &mri)
- static void addDerivedResult (const std::string &original, const std::string derived)
- static std::string addMineruleAttributeList (const ParsedMinerule::AttrVector &l)
- static std::string getNewAutoincrementValue (const std::string &tableName)
- static std::string getResultsetName (const std::string &queryname)
- static bool existsMinerule (const std::string &mrname)
- static void deleteMinerule (const std::string &mrname)
- static std::string getMRQueryName (size_t i)
- static void getMRQueryNames (std::vector< std::string > &nameVec)
- static void getMRQueryInfos (std::vector< CatalogueInfo > &catInfoVec, bool includeResultSize=true)
- static void getMRQueryInfo (const std::string &qryName, CatalogueInfo &catInfo, bool includeResult↩ Size=true)
- static void getMRQueryResultIterator (const std::string &qryName, QueryResult::Iterator &it, double sup=-1, double con=-1)

6.99.1 Detailed Description

Definition at line 62 of file OptimizerCatalogue.hpp.

6.99.2 Member Typedef Documentation

6.99.2.1 CatalogueEntry

```
typedef std::multimap< KeyCols,std::pair<KeyCols,OrderType> > minerule::OptimizerCatalogue::CatalogueEntry
```

Definition at line 75 of file OptimizerCatalogue.hpp.

6.99.2.2 KeyCols

```
typedef std::set<std::string> minerule::OptimizerCatalogue::KeyCols
```

Definition at line 70 of file OptimizerCatalogue.hpp.

6.99.3 Member Enumeration Documentation

6.99.3.1 CatalogueType

```
enum minerule::OptimizerCatalogue::CatalogueType
```

Enumerator

EQKEYS	
DEPFUN	

Definition at line 65 of file OptimizerCatalogue.hpp.

```
00065 { EQKEYS, DEPFUN } CatalogueType;
```

6.99.3.2 OrderType

```
enum minerule::OptimizerCatalogue::OrderType
```

Enumerator

Equal	
Reversed	
None	

Definition at line 66 of file OptimizerCatalogue.hpp.

```
00066 { Equal, Reversed, None } OrderType;
```

6.99.4 Constructor & Destructor Documentation

6.99.4.1 OptimizerCatalogue()

```
minerule::OptimizerCatalogue::OptimizerCatalogue ( )  [inline]
```

Definition at line 170 of file OptimizerCatalogue.hpp.

```
00170                           {
00171       /*     eqKeysCatalogue.initialize();
00172              depFunCatalogue.initialize();*/
00173     }
```

6.99.5 Member Function Documentation

6.99.5.1 addDerivedResult()

```
void minerule::OptimizerCatalogue::addDerivedResult (
             const std::string & original,
             const std::string derived )  [static]
```

Definition at line 352 of file OptimizerCatalogue.cpp.

```
00354                                          {
00355   minerule::CatalogueInfo originalInfo;
00356   bool derivedQueryAlreadyPresent = false;
00357
00358   try {
00359     minerule::OptimizerCatalogue::getMRQueryInfo(original, originalInfo, false);
00360   } catch (minerule::MineruleException &e) {
00361     throw MineruleException(
00362         MR_ERROR_CATALOGUE_ERROR,
00363         std::string(
00364             "Cannot retrieve the original minerule from the catalogue") +
00365             "The reason is:" + e.what());
00366   }
00367
00368   try {
00369     // checking the derived results do not already exist.
00370     minerule::CatalogueInfo derivedInfo;
00371     minerule::OptimizerCatalogue::getMRQueryInfo(derived, derivedInfo, false);
00372     derivedQueryAlreadyPresent = true;
00373   } catch (minerule::MineruleException &e) {
00374     // do nothing, we are "rooting" for this
00375   }
00376
00377   if (derivedQueryAlreadyPresent) {
00378     throw MineruleException(
00379         MR_ERROR_CATALOGUE_ERROR,
00380         std::string("The derived query name already exists in the catalogue. "
00381                     "Bailing out") +
00382             " so to avoid possible overwriting of useful results.");
00383   }
00384
00385   minerule::ParsedMinerule mr(originalInfo.qryText);
00386   mr.tab_result = derived;
00387   minerule::OptimizerCatalogue::addMineruleResult(
00388       minerule::OptimizerCatalogue::MineruleResultInfo(mr));
00389 }
```

6.99.5.2 addMineruleAttributeList()

```
std::string minerule::OptimizerCatalogue::addMineruleAttributeList (
            const ParsedMinerule::AttrVector & l )  [static]
```

Definition at line 245 of file OptimizerCatalogue.cpp.

```
00246                                                {
00247   if (l.empty())
00248     return "NULL";
00249
00250   std::string id = getNewAutoincrementValue("mr_att_lists");
00251   mrdb::Connection *connection =
00252       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00253
00254   std::string query =
00255       "INSERT INTO mr_att_lists (att_list_id, col_name) VALUES ";
00256
00257   ParsedMinerule::AttrVector::const_iterator it;
00258   for (it = l.begin(); it != l.end(); it++) {
00259     if (it != l.begin())
00260       query += ",";
00261
00262     query += "(" + SQLUtils::quote(id) + "," + SQLUtils::quote(*it) + ")";
00263   }
00264
00265   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00266   statement->execute(query);
00267
00268   return id;
00269 }
```

6.99.5.3 addMineruleResult()

```
void minerule::OptimizerCatalogue::addMineruleResult (
            const MineruleResultInfo & mri )  [static]
```

Definition at line 329 of file OptimizerCatalogue.cpp.

```
00331             {
00332
00333   mrdb::Connection *connection =
00334       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00335
00336   std::string galid = addMineruleAttributeList(mri.ga);
00337   std::string ralid = addMineruleAttributeList(mri.ra);
00338   std::string calid = addMineruleAttributeList(mri.ca);
00339   std::string qryid = getNewAutoincrementValue("mr_query");
00340   // std::cout«"PIPPO"«std::endl;
00341   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00342   // std::cout«"PLUTO"«std::endl;
00343   statement->execute(
00344       std::string("INSERT INTO mr_query ") +
00345       "(query_id, query_text, query_name, tab_results_name, source_tab_name," +
00346       "gal,ral,cal) VALUES (" + qryid + "," + SQLUtils::quote(mri.getText()) +
00347       "," + SQLUtils::quote(mri.tab_result) + "," +
00348       SQLUtils::quote(mri.resultset) + "," + SQLUtils::quote(mri.tab_source) +
00349       "," + galid + "," + ralid + "," + calid + ")");
00350 }
```

6.99.5.4 checkInstallation()

```
bool minerule::OptimizerCatalogue::checkInstallation ( )  [static]
```

Returns

true if the catalogue tables appear to be present.

Definition at line 271 of file OptimizerCatalogue.cpp.

```
00271                                                {
00272   mrdb::Connection *connection =
00273       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00274   mrdb::DatabaseMetaData *meta = connection->getMetaData();
00275   mrdb::ResultSet *rs = meta->getTables("mr_%");
00276
00277   typedef std::pair<std::string, bool> TableInfo;
00278   std::vector<TableInfo> tables;
00279   tables.push_back(TableInfo("mr_query", false));
00280   tables.push_back(TableInfo("mr_att_lists", false));
00281   tables.push_back(TableInfo("mr_eq_keys", false));
00282   tables.push_back(TableInfo("mr_eq_keys_col", false));
00283   tables.push_back(TableInfo("mr_dep_fun", false));
00284   tables.push_back(TableInfo("mr_dep_fun_col", false));
00285   tables.push_back(TableInfo("mr_autoincrement", false));
00286
00287   rs->next();
00288   while (!rs->isAfterLast()) {
00289     std::string currentTable = rs->getString(1);
00290     std::vector<TableInfo>::iterator it =
00291         find(tables.begin(), tables.end(),
00292              std::pair<std::string, bool>(currentTable, false));
00293
00294     if (it != tables.end()) {
00295       MRLog() << "Table: " << std::left << std::setw(50) << currentTable
00296               << StringUtils::toGreen(" OK") << std::endl;
00297       it->second = true;
00298     }
00299     rs->next();
00300   }
00301
00302   delete rs;
00303
00304   bool allTablePresents = true;
00305   for (std::vector<TableInfo>::iterator it = tables.begin(); it != tables.end();
00306        ++it) {
00307     allTablePresents &= it->second;
00308     if (!it->second) {
00309       MRLog() << "Table: " << std::left << std::setw(50) << it->first
00310               << StringUtils::toRed(" MISSING") << std::endl;
00311     }
00312   }
00313
00314   return allTablePresents;
00315 }
```

6.99.5.5 deleteMinerule()

```
void minerule::OptimizerCatalogue::deleteMinerule (
            const std::string & mrname )  [static]
```

Definition at line 45 of file OptimizerCatalogue.cpp.

```
00045                                                                {
00046   std::vector<std::string> qryNames;
00047
00048   MRLogPusher _("Deleting past minerules...");
00049
00050   Connection conn;
00051   conn.useMRDBConnection(
00052       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00053   conn.setOutTableName(mrname);
00054
00055   std::string query = (std::string) "SELECT query_id "
00056                                     "FROM mr_query "
00057                                     "WHERE query_name=" +
00058                       SQLUtils::quote(mrname) + " OR tab_results_name=" +
00059                       SQLUtils::quote(mrname);
00060
00061   std::auto_ptr<mrdb::Statement> stat(
00062       conn.getMRDBConnection()->createStatement());
00063   mrdb::ResultSet *rs = stat->executeQuery(query);
```

```
00064
00065   while (rs->next()) {
00066     MRDebug("Qry to delete:" + rs->getString(1));
00067     qryNames.push_back(rs->getString(1));
00068   }
00069
00070   delete rs;
00071
00072   MRLog() « "I found " « qryNames.size()
00073           « " minerules that need to be deleted" « std::endl;
00074
00075   assert(qryNames.size() > 0);
00076   if (qryNames.size() > 1 &&
00077       !MineruleOptions::getSharedOptions()
00078            .getSafety()
00079            .getAllowCascadeDeletes()) {
00080     throw MineruleException(MR_ERROR_SAFETY_PROBLEM,
00081                             "Requested the delete of"
00082                             " a minerule having dependent minerules, but option"
00083                             " safety::allowsCascadeDeletes is not set");
00084   }
00085
00086   std::vector<std::string>::const_iterator it;
00087   for (it = qryNames.begin(); it != qryNames.end(); it++) {
00088     MRLog() « "Deleting query: '" « *it « "'" « std::endl;
00089
00090     std::string delQry =
00091         "DELETE FROM mr_query WHERE query_id=" + SQLUtils::quote(*it);
00092
00093     try {
00094       stat->execute(delQry);
00095     } catch (mrdb::SQLException& e) {
00096       throw MineruleException(MR_ERROR_INTERNAL, "Cannot delete query " + *it + " reason:" +
      e.what());
00097     }
00098   }
00099
00100   MRLog() « "Dropping tables connected to query named:" « mrname « std::endl;
00101   conn.deleteDestTables();
00102 }
```

6.99.5.6 existsMinerule()

```
bool minerule::OptimizerCatalogue::existsMinerule (
             const std::string & mrname )  [static]
```

Parameters

mrname	

Returns

true if a query named `‘mrname’’ has been executed and a result can be found in the catalogue. it returns false otherwise.

Definition at line 32 of file OptimizerCatalogue.cpp.

```
00032                                                                    {
00033   mrdb::Connection *conn =
00034       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00035
00036   std::string query = (std::string) "SELECT * FROM mr_query WHERE query_name=" +
00037                       SQLUtils::quote(mrname);
00038
00039   std::auto_ptr<mrdb::Statement> stat(conn->createStatement());
00040   std::auto_ptr<mrdb::ResultSet> rs(stat->executeQuery(query));
00041
00042   return rs->next();
00043 }
```

6.99.5.7 getCatalogue()

```
const Catalogue & minerule::OptimizerCatalogue::getCatalogue (
            CatalogueType ct = EQKEYS ) const  [inline]
```

Definition at line 175 of file OptimizerCatalogue.hpp.

```
00175                                                                    {
00176                 switch( ct ) {
00177                         case EQKEYS:
00178                                 return eqKeysCatalogue;
00179                         case DEPFUN:
00180                                 return depFunCatalogue;
00181                         default:
00182                                 throw MineruleException(MR_ERROR_CATALOGUE_ERROR, "Requests for an
      unknown catalogue type!");
00183                 }
00184     }
```

6.99.5.8 getMRQueryInfo()

```
void minerule::OptimizerCatalogue::getMRQueryInfo (
            const std::string & qryName,
            CatalogueInfo & catInfo,
            bool includeResultSize = true )  [static]
```

As getMRQueryInfos, but the user specify a single qryName to be searched.

Definition at line 498 of file OptimizerCatalogue.cpp.

```
00500                                   {
00501   mrdb::Connection *mrdb_connection =
00502       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00503
00504   std::auto_ptr<mrdb::Statement> statement(mrdb_connection->createStatement());
00505   std::auto_ptr<mrdb::ResultSet> rs(
00506       statement->executeQuery("SELECT query_name,query_text, "
00507                               "  tab_results_name, source_tab_name "
00508                               "FROM mr_query "
00509                               "WHERE query_name='" +
00510                               qryName + "'"));
00511
00512   if (rs->next()) {
00513     setMRQueryInfoFromResultSet(rs.get(), info, includeResultSize);
00514   } else {
00515     throw MineruleException(
00516         MR_ERROR_CATALOGUE_ERROR,
00517         "Cannot find the query named:" + qryName +
00518             " "
00519             "Either there is some problem with the optimizer catalogue "
00520             "or a wrong query name has been specified");
00521   }
00522 }
```

6.99.5.9 getMRQueryInfos()

```
void minerule::OptimizerCatalogue::getMRQueryInfos (
            std::vector< CatalogueInfo > & catInfoVec,
            bool includeResultSize = true )  [static]
```

It behaves as getMRQueryNames, but provide more information to the caller.

Definition at line 479 of file OptimizerCatalogue.cpp.

```
00481                                   {
00482   mrdb::Connection *connection =
```

```
00483       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00484
00485   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00486   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(
00487       "SELECT query_name,query_text, tab_results_name, source_tab_name "
00488       "FROM mr_query"));
00489
00490   while (rs->next()) {
00491     CatalogueInfo info;
00492     setMRQueryInfoFromResultSet(rs.get(), info, includeResultSize);
00493
00494     catInfoVec.push_back(info);
00495   }
00496 }
```

6.99.5.10 getMRQueryName()

```
std::string minerule::OptimizerCatalogue::getMRQueryName (
            size_t i )  [static]
```

Returns the name of the i-th query in the catalogue.

Definition at line 415 of file OptimizerCatalogue.cpp.

```
00415                                           {
00416   //  -- This implementation is much more efficient but does not ensure that the
00417   //  result retrieved
00418   //             are in the same order of the ones returned by getMRQueryInfos
00419   //
00420   // std::string queryNumber = Converter(i-1).toString();
00421   // mrdb::Connection* connection =
00422   // MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00423   //
00424   // std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00425   // std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery("SELECT
00426   // query_name FROM mr_query LIMIT 1 OFFSET "+queryNumber));
00427   //
00428   // if( rs->next() && !rs->isAfterLast() ) {
00429   //    return rs->getString(1);
00430   // } else {
00431   //    throw MineruleException(MR_ERROR_CATALOGUE_ERROR, "Query number
00432   // "+Converter(i).toString()+" not found");
00433   // }
00434   //  -- NEW IMPLEMENTATION
00435
00436   std::vector<CatalogueInfo> catInfoVec;
00437   getMRQueryInfos(catInfoVec);
00438   if (i - 1 < catInfoVec.size()) {
00439     return catInfoVec[i - 1].resName;
00440   } else {
00441     throw MineruleException(MR_ERROR_CATALOGUE_ERROR,
00442                             "Query number " + Converter(i).toString() +
00443                                 " not found");
00444   }
00445 }
```

6.99.5.11 getMRQueryNames()

```
void minerule::OptimizerCatalogue::getMRQueryNames (
            std::vector< std::string > & nameVec )  [static]
```

The procedure fills the vector nameVec with the names of all queries that have been executed.

Definition at line 447 of file OptimizerCatalogue.cpp.

```
00448                                           {
00449   mrdb::Connection *connection =
00450       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00451
```

```
00452   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00453   std::auto_ptr<mrdb::ResultSet> rs(
00454       statement->executeQuery("SELECT query_name FROM mr_query"));
00455
00456   while (rs->next()) {
00457     nameVec.push_back(rs->getString(1));
00458   }
00459 }
```

6.99.5.12 getMRQueryResultIterator()

```
void minerule::OptimizerCatalogue::getMRQueryResultIterator (
            const std::string & qryName,
            QueryResult::Iterator & it,
            double sup = -1,
            double con = -1 )  [static]
```

It updates the QueryResultIterator so that it iterates through the results of query qryName

Parameters

qryName	the name of the query whose results the user want to iterate through
it	a user allocated QueryResultIterator which will be initialized in this function.
sup	Support threshold to be used to filter queries (the default is -1 and it means that no filter is needed)
con	Confidence threshold to be used to filter queries (the default is -1 and it means that no filter is needed)

Definition at line 524 of file OptimizerCatalogue.cpp.

```
00526                 {
00527   CatalogueInfo catInfo;
00528   getMRQueryInfo(qryName, catInfo);
00529
00530   ParsedMinerule p;
00531   MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00532   p.init(catInfo.qryText);
00533   MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00534   if (supp < 0)
00535     supp = p.sup;
00536   if (conf < 0)
00537     conf = p.conf;
00538
00539   it.init(catInfo.resName, supp, conf);
00540 }
```

6.99.5.13 getNewAutoincrementValue()

```
std::string minerule::OptimizerCatalogue::getNewAutoincrementValue (
            const std::string & tableName )  [static]
```

Definition at line 216 of file OptimizerCatalogue.cpp.

```
00217                                       {
00218   mrdb::Connection *connection =
00219       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00220
00221   std::string idstr;
00222
00223   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00224   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(
00225       "SELECT current_id FROM mr_autoincrement WHERE table_name=" +
00226       SQLUtils::quote(tableName)));
```

```
00227
00228   if (!rs->next())
00229     throw MineruleException(
00230         MR_ERROR_INSTALLATION_PROBLEM,
00231         "Cannot find the autoincrement value for table " + tableName +
00232             " in Optimizer catalogue (i.e. in table:'mr_autoincrement'),"
00233             " please check your installation");
00234
00235   size_t id = rs->getInt(1);
00236   idstr = Converter((long)++id).toString();
00237
00238   std::auto_ptr<mrdb::Statement> updateStatement(connection->createStatement());
00239   updateStatement->execute("UPDATE mr_autoincrement SET current_id=" + idstr +
00240                            " WHERE table_name=" + SQLUtils::quote(tableName));
00241
00242   return idstr;
00243 }
```

6.99.5.14 getResultsetName()

```
std::string minerule::OptimizerCatalogue::getResultsetName (
            const std::string & queryname )  [static]
```

Parameters

queryname	

Returns

astd::string containing the name of the result set of the given query. It throws a MineruleException(MR_↩ ERROR_CATALOGUE_ERROR,"...") in case queryname is not in the catalogue.

Definition at line 391 of file OptimizerCatalogue.cpp.

```
00392                                       {
00393   mrdb::Connection *connection =
00394       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00395
00396   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00397   std::auto_ptr<mrdb::ResultSet> rs(
00398       statement->executeQuery("SELECT tab_results_name "
00399                               "FROM mr_query "
00400                               "WHERE query_name=" +
00401                               SQLUtils::quote(queryname)));
00402
00403   if (!rs->next())
00404     throw MineruleException(
00405         MR_ERROR_CATALOGUE_ERROR,
00406         "Cannot find the query named:" + queryname +
00407             " "
00408             "either there is some problem with the optimizer catalogue "
00409             "or a wrong query name has been specified");
00410
00411   return rs->getString(1);
00412 }
```

6.99.5.15 hasIDConstraints()

```
bool minerule::OptimizerCatalogue::hasIDConstraints (
            const ParsedMinerule & minerule )  [static]
```

The method returns true if the constraints in the query are item dependent. Notice that, cross predicates are implicitly context dependent, even if the attributes depends on the item.

Definition at line 610 of file OptimizerCatalogue.cpp.

```
00610                                                                    {
00611   OptimizerCatalogue &optcat =
00612       MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00613
00614   list_OR_node *it_or = minerule.mc;
00615   for (; it_or != NULL; it_or = it_or->next) {
00616     list_AND_node *it_and = it_or->l_and;
00617     for (; it_and != NULL; it_and = it_and->next) {
00618       bool attr1 = SQLUtils::isAttribute(it_and->sp->val1);
00619       bool attr2 = SQLUtils::isAttribute(it_and->sp->val2);
00620       const ParsedMinerule::AttrVector *attrList;
00621       std::string theAttr;
00622
00623       if (attr1) {
00624         theAttr = it_and->sp->val1;
00625       }
00626
00627       if (attr2) {
00628         theAttr = it_and->sp->val2;
00629       }
00630
00631       if (attr1 && attr2)
00632         return false;
00633
00634       if (theAttr.substr(0, 5) == "BODY.")
00635         attrList = &minerule.ba;
00636       else if (theAttr.substr(0, 5) == "HEAD.")
00637         attrList = &minerule.ha;
00638       else
00639         throw MineruleException(MR_ERROR_MINERULETEXT_PARSING,
00640                                 "Found a condition defined over an attribute,"
00641                                 " but neither HEAD nor BODY selector has been"
00642                                 " specified.");
00643
00644       SQLUtils::removeHeadBodyFromAttrName(theAttr);
00645
00646       if ((attr1 || attr2) &&
00647           !optcat.isIDAttribute(minerule.tab_source, *attrList, theAttr))
00648         return false;
00649     }
00650   }
00651
00652   return true;
00653 }
```

6.99.5.16 install()

```
void minerule::OptimizerCatalogue::install ( )  [static]
```

Definition at line 317 of file OptimizerCatalogue.cpp.

```
00317                                  {
00318   CatalogueInstaller *installer = CatalogueInstaller::newInstaller();
00319   installer->install();
00320   delete installer;
00321 }
```

6.99.5.17 isIDAttribute()

```
bool minerule::OptimizerCatalogue::isIDAttribute (
            const std::string & tableName,
            const std::vector< std::string > & itemCols,
            const std::string & attribute ) const
```

Definition at line 570 of file OptimizerCatalogue.cpp.

```
00572                                                {
00573   std::set<std::string> itemColsSet;
00574   copy(itemCols.begin(), itemCols.end(),
00575        std::insert_iterator< std::set<std::string> >(itemColsSet,
```

```
00576                                                 itemColsSet.begin()));
00577
00578   if (itemColsSet.find(attribute) != itemColsSet.end()) {
00579     // if I am checking over the attributes that composes the attrlist then
00580     // the attribute is trivially item dependent.
00581     return true;
00582   }
00583
00584   Catalogue::const_iterator cat_it = depFunCatalogue.find(table);
00585   if (cat_it == depFunCatalogue.end())
00586     return false;
00587
00588   const CatalogueEntry &ce = cat_it->second;
00589   CatalogueEntry::const_iterator ce_it;
00590   for (ce_it = ce.begin(); ce_it != ce.end(); ce_it++) {
00591     // if I can find the attribute in the rhs of the fd
00592     // and the item cols includes the lhs of the fd
00593     if (ce_it->second.first.find(attribute) != ce_it->second.first.end() &&
00594         includes(itemColsSet.begin(), itemColsSet.end(), ce_it->first.begin(),
00595                  ce_it->first.end()))
00596       // then the attribute is item dependent
00597       return true;
00598   }
00599
00600   return false;
00601 }
```

6.99.5.18 uninstall()

```
void minerule::OptimizerCatalogue::uninstall ( )  [static]
```

Definition at line 323 of file OptimizerCatalogue.cpp.

```
00323                                  {
00324   CatalogueInstaller *installer = CatalogueInstaller::newInstaller();
00325   installer->uninstall();
00326   delete installer;
00327 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp
- /Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp

6.100 minerule::OptionBase Class Reference

```
#include <MineruleOptions.hpp>
```

Public Member Functions

- virtual ~OptionBase ()
- virtual void setOption (const std::string &name, const std::string &value)=0
- virtual OptionBase & subclassForName (const std::string &subclassName)
- virtual std::string className () const =0

6.100.1 Detailed Description

Definition at line 39 of file MineruleOptions.hpp.

6.100.2 Constructor & Destructor Documentation

6.100.2.1 ~OptionBase()

```
virtual minerule::OptionBase::~OptionBase ( )  [inline], [virtual]
```

Definition at line 41 of file MineruleOptions.hpp.

```
00041 {}
```

6.100.3 Member Function Documentation

6.100.3.1 className()

```
virtual std::string minerule::OptionBase::className ( ) const  [pure virtual]
```

6.100.3.2 setOption()

```
virtual void minerule::OptionBase::setOption (
            const std::string & name,
            const std::string & value )  [pure virtual]
```

6.100.3.3 subclassForName()

```
virtual OptionBase & minerule::OptionBase::subclassForName (
            const std::string & subclassName )  [inline], [virtual]
```

Definition at line 46 of file MineruleOptions.hpp.

```
00047       {
00048       throw MineruleException(MR_ERROR_OPTION_PARSING,
00049                               className()+" does not support sub class named:"+
00050                               subclassName);
00051     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions.hpp

6.101 mrc::Options Class Reference

```
#include <Options.hpp>
```

Data Structures

- class ListFormat

Public Types

- enum Command {
  NoCommand , ShowList , SearchQry , DeleteQry ,
  CheckCatalogue , InstallCatalogue , UninstallCatalogue , AddDerivedQuery }
- enum QryParams { QryName =0 , LastParam }

Public Member Functions

- Options ()
- void setListFormat (const char *format)
- void setShowList ()
- void setSearchQry ()
- void setDeleteQry ()
- void setCheckCatalogue ()
- void setInstallCatalogue ()
- void setUninstallCatalogue ()
- void setSearchParam (QryParams qryParam, const std::string &param)
- const std::string & getSearchParam (QryParams qryParam) const
- void setAddDerivedQuery (const std::string &original, const std::string &derived)
- const std::string & getOriginalQuery () const
- const std::string & getDerivedQuery () const
- Command getCommand () const
- const ListFormat & getListFormat () const

Protected Attributes

- Command command
- std::string searchParam [LastParam]
- std::string originalQuery
- std::string derivedQuery
- ListFormat listFormat

6.101.1 Detailed Description

Definition at line 24 of file Options.hpp.

6.101.2 Member Enumeration Documentation

6.101.2.1 Command

enum mrc::Options::Command

Enumerator

NoCommand	
ShowList	
SearchQry	
DeleteQry	
CheckCatalogue	
InstallCatalogue	
UninstallCatalogue	
AddDerivedQuery	

Definition at line 26 of file Options.hpp.

```
00026 {NoCommand,ShowList,SearchQry,DeleteQry, CheckCatalogue, InstallCatalogue, UninstallCatalogue,
       AddDerivedQuery} Command;
```

6.101.2.2 QryParams

```
enum mrc::Options::QryParams
```

Enumerator

QryName	
LastParam	

Definition at line 37 of file Options.hpp.

```
00037 {QryName=0,LastParam} QryParams;
```

6.101.3 Constructor & Destructor Documentation

6.101.3.1 Options()

```
mrc::Options::Options ( )  [inline]
```

Definition at line 49 of file Options.hpp.

```
00049 :       command(NoCommand)  {  }
```

6.101.4 Member Function Documentation

6.101.4.1 getCommand()

```
Command mrc::Options::getCommand ( ) const  [inline]
```

Definition at line 123 of file Options.hpp.

```
00123                                       {
00124                       return command;
00125               }
```

6.101.4.2 getDerivedQuery()

```
const std::string & mrc::Options::getDerivedQuery ( ) const  [inline]
```

Definition at line 119 of file Options.hpp.

```
00119                                                {
00120                       return derivedQuery;
00121               }
```

6.101.4.3 getListFormat()

```
const ListFormat & mrc::Options::getListFormat ( ) const  [inline]
```

Definition at line 127 of file Options.hpp.

```
00127                                              {
00128                       return listFormat;
00129               }
```

6.101.4.4 getOriginalQuery()

```
const std::string & mrc::Options::getOriginalQuery ( ) const  [inline]
```

Definition at line 115 of file Options.hpp.

```
00115                                               {
00116                       return originalQuery;
00117               }
```

6.101.4.5 getSearchParam()

```
const std::string & mrc::Options::getSearchParam (
            QryParams qryParam ) const  [inline]
```

Definition at line 105 of file Options.hpp.

```
00105                                                                    {
00106                       return searchParam[qryParam];
00107               }
```

6.101.4.6 setAddDerivedQuery()

```
void mrc::Options::setAddDerivedQuery (
            const std::string & original,
            const std::string & derived )  [inline]
```

Definition at line 109 of file Options.hpp.

```
00109                                                                                                   {
00110                         command = AddDerivedQuery;
00111                         originalQuery = original;
00112                         derivedQuery = derived;
00113                 }
```

6.101.4.7 setCheckCatalogue()

```
void mrc::Options::setCheckCatalogue ( )  [inline]
```

Definition at line 77 of file Options.hpp.

```
00077                                           {
00078                         if(command!=NoCommand && command != CheckCatalogue) {
00079                                 throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00080                         }
00081
00082                         command = CheckCatalogue;
00083                 }
```

6.101.4.8 setDeleteQry()

```
void mrc::Options::setDeleteQry ( )  [inline]
```

Definition at line 69 of file Options.hpp.

```
00069                                           {
00070                         if(command!=NoCommand && command!=DeleteQry) {
00071                                 throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00072                         }
00073
00074                         command = DeleteQry;
00075                 }
```

6.101.4.9 setInstallCatalogue()

```
void mrc::Options::setInstallCatalogue ( )  [inline]
```

Definition at line 85 of file Options.hpp.

```
00085                                           {
00086                         if(command!=NoCommand && command != InstallCatalogue) {
00087                                 throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00088                         }
00089
00090                         command = InstallCatalogue;
00091                 }
```

6.101.4.10 setListFormat()

```
void mrc::Options::setListFormat (
            const char * format )
```

Definition at line 21 of file Options.cpp.

```
00021                                                   {
00022                 if(format==NULL)
00023                         return;
00024
00025                 for(int i=0; format[i]!='\0'; i++) {
00026                         switch( format[i] ) {
00027                                 case 's':
00028                                         listFormat.size=true;
00029                                         break;
00030                                 case 't':
00031                                         listFormat.text = true;
00032                                         break;
00033                                 case 'r':
00034                                         listFormat.result = true;
00035                                         break;
00036                                 default:
00037                                         throw Exception(mrc::ERROR_OPTION_PARSING,
      std::string("Unknown flag:")+format[i]);
00038                         }
00039                 }
00040         }
```

6.101.4.11 setSearchParam()

```
void mrc::Options::setSearchParam (
            QryParams qryParam,
            const std::string & param )  [inline]
```

Definition at line 101 of file Options.hpp.

```
00101                                                                                   {
00102                         searchParam[qryParam]=param;
00103                 }
```

6.101.4.12 setSearchQry()

```
void mrc::Options::setSearchQry ( )  [inline]
```

Definition at line 61 of file Options.hpp.

```
00061                                   {
00062                         if(command!=NoCommand && command!=SearchQry) {
00063                                 throw Exception( mrc::ERROR_OPTION_PARSING, "You can specify either -l
      or -n option, but not both!" );
00064                         }
00065
00066                         command = SearchQry;
00067                 }
```

6.101.4.13 setShowList()

```
void mrc::Options::setShowList ( )  [inline]
```

Definition at line 53 of file Options.hpp.

```
00053                                   {
00054                         if(command!=NoCommand && command!=ShowList) {
00055                                 throw Exception( mrc::ERROR_OPTION_PARSING,    "You can specify either
      -l or -n option, but not both!" );
00056                         }
00057
00058                         command = ShowList;
00059                 }
```

6.101.4.14 setUninstallCatalogue()

```
void mrc::Options::setUninstallCatalogue ( )  [inline]
```

Definition at line 93 of file Options.hpp.

```
00093                                           {
00094                         if(command!=NoCommand && command != UninstallCatalogue) {
00095                                 throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00096                         }
00097
00098                         command = UninstallCatalogue;
00099                 }
```

6.101.5 Field Documentation

6.101.5.1 command

```
Command mrc::Options::command  [protected]
```

Definition at line 40 of file Options.hpp.

6.101.5.2 derivedQuery

```
std::string mrc::Options::derivedQuery  [protected]
```

Definition at line 44 of file Options.hpp.

6.101.5.3 listFormat

```
ListFormat mrc::Options::listFormat  [protected]
```

Definition at line 46 of file Options.hpp.

6.101.5.4 originalQuery

```
std::string mrc::Options::originalQuery  [protected]
```

Definition at line 43 of file Options.hpp.

6.101.5.5 searchParam

```
std::string mrc::Options::searchParam[LastParam]  [protected]
```

Definition at line 42 of file Options.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.hpp
- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Options.cpp

6.102 mrmatch::Options Class Reference

```
#include <Options.hpp>
```

Public Member Functions

- Options ()
- virtual ~Options ()
- void setQueryName (std::string name)
- void setQueryNumber (size_t number)
- void setTableName (const std::string &)
- const std::string & tableName () const
- std::string queryName () const
- MatcherSpecs matcherSpecs () const
- void setMatchKind (MatchKind kind)
- void setMatchOutput (MatchOutput out)
- void setMatchOutputTableName (const std::string &name)
- const std::string & getMatchOutputTableName () const
- bool initMineruleOptions () const
- void printUsage (int argc, char *const argv[])
- void parse (int argc, char *const argv[])

6.102.1 Detailed Description

Definition at line 25 of file Options.hpp.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 Options()

```
mrmatch::Options::Options ( )  [inline]
```

Definition at line 27 of file Options.hpp.

```
00027 : _queryNumber(0), _matchKind(RuleGids), _matchOutput(OutOnConsole) {};
```

6.102.2.2 ~Options()

```
virtual mrmatch::Options::~Options ( )  [inline], [virtual]
```

Definition at line 28 of file Options.hpp.

```
00028 {};
```

6.102.3 Member Function Documentation

6.102.3.1 getMatchOutputTableName()

```
const std::string & mrmatch::Options::getMatchOutputTableName ( ) const  [inline]
```

Definition at line 41 of file Options.hpp.

```
00041 { return _matchOutputTableName; }
```

6.102.3.2 initMineruleOptions()

```
bool mrmatch::Options::initMineruleOptions ( ) const
```

Definition at line 64 of file Options.cpp.

```
00064                                           {
00065                 bool ok=false;
00066                 MineruleOptions& options = MineruleOptions::getSharedOptions();
00067
00068                 if( !_mrOptionsFileName.empty() ) {
00069                         options.readFromFile(_mrOptionsFileName);
00070                         ok=true;
00071                 } else if( FileUtils::fileExists("options.txt") ) {
00072                         options.readFromFile("options.txt");
00073                         ok=true;
00074                 }
00075
00076                 std::vector<std::string>::const_iterator it;
00077                 for(it=_mrOptionsInline.begin(); it!=_mrOptionsInline.end(); ++it) {
00078                         options.readFromString(*it);
00079                         ok=true;
00080                 }
00081
00082                 return ok;
00083         }
```

6.102.3.3 matcherSpecs()

```
MatcherSpecs mrmatch::Options::matcherSpecs ( ) const  [inline]
```

Definition at line 36 of file Options.hpp.

```
00036 { return _matchKind | _matchOutput; }
```

6.102.3.4 parse()

```
void mrmatch::Options::parse (
            int argc,
            char *const argv[] )
```

Definition at line 110 of file Options.cpp.

```
00110                                                   {
00111                 char ch;
00112
00113                 while( (ch=getopt(argc, argv, "hcn:s:d:t:")) != -1 ) {
00114                         switch(ch) {
00115                                 case 'h':
00116                                         printUsage(argc, argv);
00117                                         break;
00118                                 case 'c':
00119                                         StringUtils::setColorsEnabled(false);
00120                                         break;
00121                                 case 'n':
00122                                         setQueryNumber(Converter(optarg).toLong());
00123                                         break;
00124                                 case 's':
00125                                         setMatchKind(stringToMatchKind(optarg));
00126                                         break;
00127                                 case 'd':
00128                                         setMatchOutputTableName(optarg);
00129                                         setMatchOutput(OutOnDB);
00130                                         break;
00131                                 case 't':
00132                                         setTableName(optarg);
00133                                         break;
00134                                 case '?':
00135                                 default:
00136                                         std::cout « StringUtils::toBold("Option not recognized:") «
      "-" « ch « std::endl;
00137                                         printUsage(argc, argv);
00138                                         exit(MRMATCH_OPTION_PARSING_ERROR);
00139                         }
00140                 }
00141
00142                 if( optind < argc-1 ) {
00143                         std::cout « StringUtils::toBold("Error:") « "some of the options were not
      recognized" « std::endl;
00144
00145                         printUsage(argc, argv);
00146                         exit(MRMATCH_OPTION_PARSING_ERROR);
00147                 }
00148
00149                 if( optind == argc-1 )
00150                         setQueryName(argv[argc-1]);
00151         }
```

6.102.3.5 printUsage()

```
void mrmatch::Options::printUsage (
            int argc,
            char *const argv[] )
```

Definition at line 26 of file Options.cpp.

```
00026                                                           {
00027                 char* progName = argv[0];
00028
00029                 std::cout « StringUtils::toBold("Usage:") « std::endl
00030                         « "  " « StringUtils::toBold(progName) « " [-h] [-c] [-n <num query>] [-O
      <optionlist>] [-f <optionfile>] [-d <outputtable>]  [-t <table name>] [-s <sort order>] [<query
      name>] " « std::endl « std::endl;
00031
00032                 std::cout
00033                         « StringUtils::toBold("   -h") « " - prints this message." « std::endl
00034                         « StringUtils::toBold("   -c") « " - suppress colors." « std::endl
00035                         « StringUtils::toBold("   -n") « " - specifies a query number (instead of a
      query name)." « std::endl
00036                         « StringUtils::toBold("   -O") « " - specifies a minerule option on the
      command line (overrides those read from file)." « std::endl
00037                         « StringUtils::toBold("   -f") « " - specify a file name containing the
      Options to be used." « std::endl
00038                         « "         default is 'optins.txt'"« std::endl
00039                         « StringUtils::toBold("   -d") « " - redirect the output onto the database.
      The output table must" «std::endl
00040                         « "         be specified as the argument to this option" « std::endl
00041                         « StringUtils::toBold("   -t") « " - specify a table name for the match (the
      table *must* have the" « std::endl
00042                         « "         same schema as the mining table used for the query)" « std::endl
00043                         « StringUtils::toBold("   -s") « " - sets the output sorting order. The
      parameter can be set to:" « std::endl
00044                         « StringUtils::toBold("         RuleGids") « ", to produce an output with the
      format: 'rule -> list of matching gids'" « std::endl
00045                         « StringUtils::toBold("         GidRules") « ", to produce an output with the
      format: 'gid -> list of matching rules'."  « std::endl
00046                         « "         default is 'RuleGids'. You can shorten the parameters as you like
      (e.g., 'G' for GidRules)." « std::endl;
00047
00048                 exit(SUCCESS);
00049         }
```

6.102.3.6 queryName()

```
std::string mrmatch::Options::queryName ( ) const
```

Definition at line 85 of file Options.cpp.

```
00085                                           {
00086                 if( _queryNumber != 0 && !_queryName.empty() ) {
00087                         std::cout « StringUtils::toBold("Error:") « "It appears that you specified
      both a query name and a query number" « std::endl;
00088                         exit(MRMATCH_OPTION_PARSING_ERROR);
00089                 }
00090
00091                 if( _queryNumber != 0 )
00092                         return OptimizerCatalogue::getMRQueryName(_queryNumber);
00093
00094                 if( !_queryName.empty() )
00095                         return _queryName;
00096
00097                 std::cout « StringUtils::toBold("Error:") « "It appears that you specified neither a
      query name nor a query number" « std::endl;
00098                 exit(MRMATCH_OPTION_PARSING_ERROR);
00099         }
```

6.102.3.7 setMatchKind()

```
void mrmatch::Options::setMatchKind (
            MatchKind kind )  [inline]
```

Definition at line 38 of file Options.hpp.

```
00038 { _matchKind = kind; }
```

6.102.3.8 setMatchOutput()

```
void mrmatch::Options::setMatchOutput (
            MatchOutput out )  [inline]
```

Definition at line 39 of file Options.hpp.

```
00039 { _matchOutput = out; }
```

6.102.3.9 setMatchOutputTableName()

```
void mrmatch::Options::setMatchOutputTableName (
            const std::string & name )  [inline]
```

Definition at line 40 of file Options.hpp.

```
00040 { _matchOutputTableName = name; }
```

6.102.3.10 setQueryName()

```
void mrmatch::Options::setQueryName (
            std::string name )  [inline]
```

Definition at line 30 of file Options.hpp.

```
00030 { _queryName   = name;          }
```

6.102.3.11 setQueryNumber()

```
void mrmatch::Options::setQueryNumber (
            size_t number )  [inline]
```

Definition at line 31 of file Options.hpp.

```
00031 { _queryNumber = number;  }
```

6.102.3.12 setTableName()

```
void mrmatch::Options::setTableName (
            const std::string & tableName )
```

Definition at line 105 of file Options.cpp.

```
00105                                                   {
00106                 _tableName = tableName;
00107         }
```

6.102.3.13 tableName()

```
const std::string & mrmatch::Options::tableName ( ) const
```

Definition at line 101 of file Options.cpp.

```
00101                                                {
00102                 return _tableName;
00103         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRMatch/Options.hpp
- /Users/esposito/Software/minerule/src/Programs/MRMatch/Options.cpp

6.103 mrprint::Options Class Reference

```
#include <Options.hpp>
```

Public Member Functions

- Options (int argc, char **argv)
- virtual ~Options ()
- void parse ()
- void printHelp () const
- std::string mroptFileName () const
- bool noLogArtifacts () const
- bool noLowConfidenceFilter () const
- std::string sortOrder () const
- size_t queryNumber () const
- std::string queryName () const

6.103.1 Detailed Description

Definition at line 27 of file Options.hpp.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 Options()

```
mrprint::Options::Options (
            int argc,
            char ** argv )  [inline]
```

Definition at line 29 of file Options.hpp.

```
00029                                    :
00030                                _argv(argv),
00031                                _argc(argc),
00032                                _mroptFileName(minerule::MineruleOptions::DEFAULT_FILE_NAME),
00033                                _noLogArtifacts(false),
00034                                _noLowConfidenceFilter(false),
00035                                _sortOrder("no"),
00036                                _queryNumber(0),
00037                                _queryName("") {};
```

6.103.2.2 ~Options()

```
virtual mrprint::Options::~Options ( )  [inline], [virtual]
```

Definition at line 38 of file Options.hpp.
```
00038 {};
```

6.103.3 Member Function Documentation

6.103.3.1 mroptFileName()

```
std::string mrprint::Options::mroptFileName ( ) const  [inline]
```

Definition at line 44 of file Options.hpp.
```
00044 { return _mroptFileName; }
```

6.103.3.2 noLogArtifacts()

```
bool mrprint::Options::noLogArtifacts ( ) const  [inline]
```

Definition at line 45 of file Options.hpp.
```
00045 { return _noLogArtifacts; }
```

6.103.3.3 noLowConfidenceFilter()

```
bool mrprint::Options::noLowConfidenceFilter ( ) const  [inline]
```

Definition at line 46 of file Options.hpp.
```
00046 { return _noLowConfidenceFilter; }
```

6.103.3.4 parse()

```
void mrprint::Options::parse ( )
```

Definition at line 55 of file Options.cpp.

```
00055                                {
00056                 const char* optstr = "hc0ln:s:";
00057
00058                 int opt;
00059                 while( (opt = getopt(_argc, _argv, optstr)) != -1 ) {
00060                         switch(opt) {
00061                                 case 'h':
00062                                 printHelp();
00063                                 break;
00064                                 case 'f':
00065                                 _mroptFileName = optarg;
00066                                 break;
00067                                 case 'c':
00068                                 minerule::StringUtils::setColorsEnabled(false);
00069                                 break;
00070                                 case '0':
00071                                 _noLogArtifacts=true;
00072                                 break;
00073                                 case 'l':
00074                                 _noLowConfidenceFilter=true;
00075                                 break;
00076                                 case 'n':
00077                                 _queryNumber = minerule::Converter(optarg).toLong();
00078                                 break;
00079                                 case 's':
00080                                 _sortOrder = optarg;
00081                                 break;
00082                                 default:
00083                                 std::cout « minerule::StringUtils::toRed("Error:") « " option " « opt
      « " not recognized." « std::endl;
00084                                 exit(MRPRINT_OPTION_PARSING_ERROR);
00085                         }
00086                 }
00087
00088                 if(optind == _argc-1) {
00089                         _queryName = _argv[optind];
00090                 }
00091
00092                 bool noQueryName = _queryName.empty() && _queryNumber == 0;
00093                 bool noOptionFile = !minerule::FileUtils::fileExists(_mroptFileName);
00094
00095                 if( noQueryName ) {
00096                         printError("no query name has been given. Please give one or use -n to specify
      a query number.");
00097                         exit(MRPRINT_OPTION_PARSING_ERROR);
00098                 }
00099
00100                if(noOptionFile) {
00101                        printError("option file " + _mroptFileName + " not found.");
00102                        exit(MRPRINT_OPTION_PARSING_ERROR);
00103                }
00104        }
```

6.103.3.5 printHelp()

```
void mrprint::Options::printHelp ( ) const
```

Definition at line 30 of file Options.cpp.

```
00030                                    {
00031                 std::cout « minerule::StringUtils::toBold("Usage:") « std::endl
00032                         « "    " « minerule::StringUtils::toBold(_argv[0]) « " [-h] [-f <optionfile>]
      [-0] [-c] [-n minerule-number] [-s <order>] resultsetname" « std::endl
00033                         « "The program allows printing results of minerule queries." « std::endl
00034                         « std::endl
00035                         « minerule::StringUtils::toBold("-h") « " - prints this message " « std::endl
00036                         « minerule::StringUtils::toBold("-f") « " - Specifies a minerule option file
      (default: '"« minerule::MineruleOptions::DEFAULT_FILE_NAME « "')" « std::endl
00037                         « minerule::StringUtils::toBold("-c") « " - suppress colors " « std::endl
00038                         « minerule::StringUtils::toBold("-0") « " - suppresses logging artifacts " «
      std::endl
```

```
00039                         « minerule::StringUtils::toBold("-l") « " - do not filter out rules having low
      confidence" « std::endl
00040                         « minerule::StringUtils::toBold("-n") « " - specifies a query number to be
      printed (this is an alternative to" « std::endl
00041                         « "     specifying the query name)" « std::endl
00042                         « minerule::StringUtils::toBold("-s") « " - sorts the rules in a specified
      order." « std::endl
00043                         « "   supported orders are: " « std::endl
00044                         « "     'no' -> no particular order (fastest display)" « std::endl
00045                         « "     'scbh' -> order is support, confidence, body, head"«std::endl
00046                         « "     'bhsc' -> order is body, head, supp, conf"«std::endl
00047                         « "     'hbsc' -> order is head, body, supp,  conf"«std::endl
00048                         « "     'csbh' -> order is conf, supp, body, head" « std::endl
00049                         « "     'cbhs' -> order is conf, body, head, supp" « std::endl
00050                         « "     'cbsh' -> order is conf, body, supp, head" « std::endl
00051                         « "   the default is 'no'" « std::endl
00052                         « std::endl « std::endl;
00053         }
```

6.103.3.6 queryName()

```
std::string mrprint::Options::queryName ( ) const  [inline]
```

Definition at line 49 of file Options.hpp.

```
00049 { return _queryName; }
```

6.103.3.7 queryNumber()

```
size_t mrprint::Options::queryNumber ( ) const  [inline]
```

Definition at line 48 of file Options.hpp.

```
00048 { return _queryNumber; }
```

6.103.3.8 sortOrder()

```
std::string mrprint::Options::sortOrder ( ) const  [inline]
```

Definition at line 47 of file Options.hpp.

```
00047 { return _sortOrder; }
```

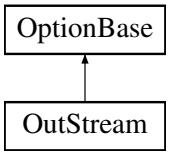
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.hpp
- /Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.cpp

6.104 OutStream Class Reference

```
#include <outstream.hpp>
```

Inheritance diagram for OutStream:



Public Member Functions

- OutStream (MineruleOptions *opt, const std::string &name)
- virtual ~OutStream ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- void setLogger (MRLogger &log)
- MRLogger & getLogger ()
- std::ostream & getStream () const
- void setLogLevel (size_t level)
- void disable ()
- void enable ()

6.104.1 Detailed Description

Definition at line 16 of file outstream.hpp.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 OutStream()

```
OutStream::OutStream (
            MineruleOptions * opt,
            const std::string & name )  [inline]
```

Definition at line 24 of file outstream.hpp.

```
00025         : logger(NULL), options(opt), streamName(name) {};
```

6.104.2.2 ~OutStream()

```
virtual OutStream::~OutStream ( )  [inline], [virtual]
```

Definition at line 26 of file outstream.hpp.

```
00026 {};
```

6.104.3 Member Function Documentation

6.104.3.1 className()

```
virtual std::string OutStream::className ( ) const  [inline], [virtual]
```

Definition at line 28 of file outstream.hpp.

```
00028                                       {
00029         return streamName;
00030   }
```

6.104.3.2 disable()

```
void OutStream::disable ( )  [inline]
```

Definition at line 54 of file outstream.hpp.

```
00054                  {
00055         logger->setLogLevel(0);
00056   }
```

6.104.3.3 enable()

```
void OutStream::enable ( )  [inline]
```

Definition at line 58 of file outstream.hpp.

```
00058                  {
00059         logger->setLogLevel(logLevel);
00060   }
```

6.104.3.4 getLogger()

```
MRLogger & OutStream::getLogger ( )  [inline]
```

Definition at line 40 of file outstream.hpp.

```
00040                          {
00041         return *logger;
00042   }
```

6.104.3.5 getStream()

```
std::ostream & OutStream::getStream ( ) const  [inline]
```

Definition at line 43 of file outstream.hpp.

```
00043                                  {
00044         assert(logger!=NULL);
00045         return logger->log();
00046   }
```

6.104.3.6 setLogger()

```
void OutStream::setLogger (
             MRLogger & log )  [inline]
```

Definition at line 35 of file outstream.hpp.

```
00035                                {
00036         logger=&log;
00037         logLevel = log.getLogLevel();
00038   }
```

6.104.3.7 setLogLevel()

```
void OutStream::setLogLevel (
             size_t level )  [inline]
```

Definition at line 48 of file outstream.hpp.

```
00048                                    {
00049         assert(logger!=NULL);
00050         logger->setLogLevel(level);
00051         logLevel=level;
00052   }
```

6.104.3.8 setOption()

```
virtual void OutStream::setOption (
             const std::string & name,
             const std::string & value )  [virtual]
```

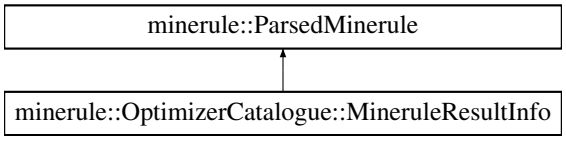
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/outstream.hpp

6.105 minerule::ParsedMinerule Class Reference

`#include <ParsedMinerule.hpp>`

Inheritance diagram for minerule::ParsedMinerule:



Public Types

- typedef std::vector< std::string > AttrVector
- typedef Bem_cond * bem_c

Public Member Functions

- ParsedMinerule ()
- ParsedMinerule (const std::string &minerule_text)
- ~ParsedMinerule ()
- ParsedMinerule (const ParsedMinerule &mr)
- void init (const std::string &minerule_text)
- bool requiresClusters () const
- bool hasCrossConditions (const list_OR_node *cond) const
- bool hasCrossConditions () const
- bool hasDisjuctionsInMC () const
- bool hasSameBodyHead () const
- std::string getMinesequenceText () const
- std::string getMineruleText () const
- std::string getMineitemsetsText () const
- std::string getText () const
- std::string getBEMText () const

Data Fields

- AttrVector ga
- AttrVector oa
- AttrVector ca
- AttrVector ra
- AttrVector ba
- AttrVector ha
- list_OR_node * mc
- list_OR_node * gc
- list_OR_node * cc
- AttrVector c_aggr_list
- float sup
- float conf
- MinMaxPair bodyCardinalities
- MinMaxPair headCardinalities
- std::string tab_source
- std::string tab_result
- bool tautologies
- bool body_coincident_head
- bool distinct
- MinMaxPair sequenceAllowedGaps
- MinMaxPair length
- std::vector< Dist_cond * > seq_dist_vect
- std::vector< Bem_cond * > seq_bem_vect
- std::string filter_condition
- MiningTasks miningTask

6.105.1 Detailed Description

Definition at line 136 of file ParsedMinerule.hpp.

6.105.2 Member Typedef Documentation

6.105.2.1 AttrVector

```
typedef std::vector<std::string> minerule::ParsedMinerule::AttrVector
```

Definition at line 139 of file ParsedMinerule.hpp.

6.105.2.2 bem_c

```
typedef Bem_cond* minerule::ParsedMinerule::bem_c
```

Definition at line 140 of file ParsedMinerule.hpp.

6.105.3 Constructor & Destructor Documentation

6.105.3.1 ParsedMinerule() [1/3]

```
minerule::ParsedMinerule::ParsedMinerule ( )  [inline]
```

Definition at line 191 of file ParsedMinerule.hpp.

```
00191                     :
00192     mc(NULL),
00193     gc(NULL),
00194     cc(NULL),
00195     sup(0.0),
00196     conf(0.0),
00197     bodyCardinalities( MinMaxPair(1,1000)),
00198     headCardinalities( MinMaxPair(1,1000)),
00199     sequenceAllowedGaps( MinMaxPair(0,0)),
00200     length(MinMaxPair(1,std::numeric_limits<int>::max())) {
00201   };
```

6.105.3.2 ParsedMinerule() [2/3]

```
minerule::ParsedMinerule::ParsedMinerule (
            const std::string & minerule_text )  [inline], [explicit]
```

Definition at line 203 of file ParsedMinerule.hpp.

```
00203                                                     :
00204     mc(NULL),
00205     gc(NULL),
00206     cc(NULL),
00207     sup(0.0),
00208     conf(0.0),
00209     bodyCardinalities(1,1000),
00210     headCardinalities(1,1000),
00211     sequenceAllowedGaps( MinMaxPair(0,0)),
00212     length(MinMaxPair(1,std::numeric_limits<int>::max())) {
00213     init(minerule_text);
00214   }
```

6.105.3.3 ~ParsedMinerule()

```
minerule::ParsedMinerule::~ParsedMinerule ( )  [inline]
```

Definition at line 216 of file ParsedMinerule.hpp.

```
00216                     {
00217 #ifdef MRUSERWARNING
00218 #warning CAPIRE PERCHE" DA SEGMENTATION FAULT NEL CASO LE SEGUENTI FREE SIANO DECOMMENTATE!
00219           // possible explanation: not all char* pointers in simple predicates have
00220           // been dynamically allocated...
00221 #endif
00222     //free_l_OR(mc);
00223     //free_l_OR(gc);
00224     //free_l_OR(cc);
00225   }
```

6.105.3.4 ParsedMinerule() [3/3]

```
minerule::ParsedMinerule::ParsedMinerule (
            const ParsedMinerule & mr )  [inline]
```

Definition at line 227 of file ParsedMinerule.hpp.

```
00227                                          :
00228       oa(mr.oa),
00229       filter_condition(mr.filter_condition),
00230       distinct(mr.distinct),
00231     ga(mr.ga),
00232     ca(mr.ca),
00233     ra(mr.ra),
00234     ba(mr.ba),
00235     ha(mr.ha),
00236     c_aggr_list(mr.c_aggr_list),
00237     sup(mr.sup),
00238     conf(mr.conf),
00239     bodyCardinalities( mr.bodyCardinalities ),
00240     headCardinalities( mr.headCardinalities ),
00241     tab_source(mr.tab_source),
00242     tab_result(mr.tab_result),
00243     tautologies(mr.tautologies),
00244     body_coincident_head(mr.body_coincident_head),
00245     sequenceAllowedGaps(mr.sequenceAllowedGaps),
00246     miningTask(mr.miningTask),
00247     length(mr.length){
00248     mc=clone_l_OR(mr.mc);
00249     gc=clone_l_OR(mr.gc);
00250     cc=clone_l_OR(mr.cc);
00251     seq_bem_vect= Bem_cond::copyBemCond(mr.seq_bem_vect);
00252     seq_dist_vect= Dist_cond::copyDistCond(mr.seq_dist_vect);
00253   }
```

6.105.4 Member Function Documentation

6.105.4.1 getBEMText()

```
std::string minerule::ParsedMinerule::getBEMText ( ) const
```

Definition at line 394 of file ParsedMinerule.cpp.

```
00394                                                {
00395           std::string ris = "";
00396
00397           for(int i = 0 ; i < seq_bem_vect.size(); ++i)
00398               ris += seq_bem_vect[i]->type +"."+seq_bem_vect[i]->attr+" "+seq_bem_vect[i]->op+"
      "+seq_bem_vect[i]->val+" ";
00399
00400           return ris;
00401       }
```

6.105.4.2 getMineitemsetsText()

```
std::string minerule::ParsedMinerule::getMineitemsetsText ( ) const
```

Definition at line 370 of file ParsedMinerule.cpp.

```
00370                                                                   {
00371                         std::string result;
00372                         result =
00373                                 "MINE ITEMSET " + tab_result + " AS " +
00374                                         "SELECT DISTINCT " +
00375                                                 getCardsText(bodyCardinalities) + " " +
      getAttrText(ba) + " AS BODY" +
00376                                                         ", SUPPORT ";
00377
00378                         if( mc!=NULL )
00379                                 result += "WHERE "+getCondText(mc) + " ";
00380
00381                         result +=
00382                                 "FROM "+tab_source + " "
00383                                         "GROUP BY " + getAttrText(ga) + " ";
00384
00385                         if( gc!=NULL )
00386                                 result += getCondText(gc) + " ";
00387
00388                         result +=
00389                                 "EXTRACTING ITEMSETS WITH SUPPORT:" + Converter(sup).toString();
00390
00391                         return result;
00392                 }
```

6.105.4.3 getMineruleText()

```
std::string minerule::ParsedMinerule::getMineruleText ( ) const
```

Definition at line 338 of file ParsedMinerule.cpp.

```
00338                                                                   {
00339                         std::string result;
00340                         result =
00341                                 "MINE RULE " + tab_result + " AS " +
00342                                 "SELECT DISTINCT " +
00343                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) + " AS BODY,"
    +
00344                                 getCardsText(headCardinalities) + " " + getAttrText(ha) + " AS HEAD"
00345                                 ", SUPPORT, CONFIDENCE ";
00346
00347                         if( mc!=NULL )
00348                                 result += "WHERE "+getCondText(mc) + " ";
00349
00350                         result +=
00351                                 "FROM "+tab_source + " "
00352                                         "GROUP BY " + getAttrText(ga) + " ";
00353
00354                         if( gc!=NULL )
00355                                 result += getCondText(gc) + " ";
00356
00357                         if( !ca.empty() ) {
00358                                 result += "CLUSTER BY " + getAttrText(ca) + " ";
00359
00360                                 if( cc!=NULL )
00361                                         result+= "HAVING " +getCondText(cc) + " ";
00362                         }
00363
00364                         result +=
00365                                 "EXTRACTING RULES WITH SUPPORT:" + Converter(sup).toString() + ","+
00366                                         "CONFIDENCE:"+Converter(conf).toString();
00367
00368                         return result;
00369                 }
```

6.105.4.4 getMinesequenceText()

```
std::string minerule::ParsedMinerule::getMinesequenceText ( ) const
```

Definition at line 308 of file ParsedMinerule.cpp.

```
00308                                                                   {
00309                         std::string result;
00310             std::string distinct= this->distinct? " DISTINCT " : " " ;
00311             std::string Bem_text= getBEMText();
00312             if(Bem_text.size()>0)
00313                 Bem_text= " HAVING " + Bem_text;
00314                         result =
00315                                 "MINE SEQUENCE " + tab_result + " AS " +
00316                 "SELECT " + distinct +
00317                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) +
00318                     ", SUPPORT FROM "+tab_source+" "+Bem_text+" GROUP BY "+getAttrText(ga)+
00319                                 " ORDER BY "+getAttrText(oa)+ " EXTRACTING SEQUENCES WITH SUPPORT:" +
00320                                 Converter(sup).toString();
00321
00322                         return result;
00323                 }
```

6.105.4.5 getText()

```
std::string minerule::ParsedMinerule::getText ( ) const
```

Definition at line 324 of file ParsedMinerule.cpp.

```
00324                                                   {
00325                         switch( miningTask ) {
00326                                 case MTMineRules:
00327                                 return getMineruleText();
00328                                 case MTMineSequences:
00329                                 return getMinesequenceText();
00330                                 case MTMineItemsets:
00331                                 return getMineitemsetsText();
00332                                 default:
00333                                 throw MineruleException( MR_ERROR_MINERULETEXT_PARSING,
00334                                         "Current query is not currently supported by the system "
00335                                                 "(i.e., it is not a MINE RULE nor a MINE SEQUENCE
      query" );
00336                         }
00337                 }
```

6.105.4.6 hasCrossConditions() [1/2]

```
bool minerule::ParsedMinerule::hasCrossConditions ( ) const  [inline]
```

Definition at line 264 of file ParsedMinerule.hpp.

```
00264                                   {
00265     return hasCrossConditions(mc) || hasCrossConditions(gc) || hasCrossConditions(cc);
00266   }
```

6.105.4.7 hasCrossConditions() [2/2]

```
bool minerule::ParsedMinerule::hasCrossConditions (
            const list_OR_node * cond ) const
```

Definition at line 403 of file ParsedMinerule.cpp.

```
00403                                                                                   {
00404                         MRLog()<<"Checking whether the minerule contains cross predicates" << std::endl;
00405
00406                         const list_OR_node* curr_OR;
00407                         for(curr_OR=cond; curr_OR!=NULL; curr_OR=curr_OR->next) {
00408                                 list_AND_node* curr_AND;
00409                                 for(curr_AND=curr_OR->l_and; curr_AND!=NULL; curr_AND=curr_AND->next)
      {
00410                                         assert(curr_AND->sp!=NULL);
00411                                         MRLog() << curr_AND->sp->val1 << " " << curr_AND->sp->val2 <<
      std::endl;
00412                                         bool hasHead =
00413                                                 strstr(curr_AND->sp->val1,
      "HEAD.")==curr_AND->sp->val1 ||
00414                                                         strstr(curr_AND->sp->val2,
      "HEAD.")==curr_AND->sp->val2;
00415                                         bool hasBody =
00416                                                 strstr(curr_AND->sp->val1,
      "BODY.")==curr_AND->sp->val1 ||
00417                                                         strstr(curr_AND->sp->val2,
      "BODY.")==curr_AND->sp->val2;
00418                                         if( hasHead && hasBody ) {
00419                                                 MRLog() << "Cross predicate found!" << std::endl;
00420                                                 return true;
00421                                         }
00422                                 }
00423                         }
00424
00425                         MRLog() << "Cross predicate not found!" << std::endl;
00426                         return false;
00427                 }
```

6.105.4.8 hasDisjuctionsInMC()

```
bool minerule::ParsedMinerule::hasDisjuctionsInMC ( ) const  [inline]
```

Definition at line 268 of file ParsedMinerule.hpp.

```
00268                                       {
00269     return mc!=NULL && mc->next!=NULL;
00270   }
```

6.105.4.9 hasSameBodyHead()

```
bool minerule::ParsedMinerule::hasSameBodyHead ( ) const  [inline]
```

Definition at line 272 of file ParsedMinerule.hpp.

```
00272                                     {
00273           return ba == ha;
00274   }
```

6.105.4.10 init()

```
void minerule::ParsedMinerule::init (
            const std::string & minerule_text )
```

Definition at line 239 of file ParsedMinerule.cpp.

```
00239                                                                   {
00240                       parseMinerule( minerule_text, *this );
00241               }
```

6.105.4.11 requiresClusters()

```
bool minerule::ParsedMinerule::requiresClusters ( ) const  [inline]
```

Definition at line 257 of file ParsedMinerule.hpp.

```
00257                                {
00258     return !ca.empty();
00259     //  return !(body_coincident_head && ca.empty());
00260   }
```

6.105.5 Field Documentation

6.105.5.1 ba

```
AttrVector minerule::ParsedMinerule::ba
```

Definition at line 158 of file ParsedMinerule.hpp.

6.105.5.2 body_coincident_head

```
bool minerule::ParsedMinerule::body_coincident_head
```

Definition at line 177 of file ParsedMinerule.hpp.

6.105.5.3 bodyCardinalities

```
MinMaxPair minerule::ParsedMinerule::bodyCardinalities
```

Definition at line 171 of file ParsedMinerule.hpp.

6.105.5.4 c_aggr_list

```
AttrVector minerule::ParsedMinerule::c_aggr_list
```

Definition at line 167 of file ParsedMinerule.hpp.

6.105.5.5 ca

```
AttrVector minerule::ParsedMinerule::ca
```

Definition at line 155 of file ParsedMinerule.hpp.

6.105.5.6 cc

```
list_OR_node* minerule::ParsedMinerule::cc
```

Definition at line 165 of file ParsedMinerule.hpp.

6.105.5.7 conf

```
float minerule::ParsedMinerule::conf
```

Definition at line 170 of file ParsedMinerule.hpp.

6.105.5.8 distinct

```
bool minerule::ParsedMinerule::distinct
```

Definition at line 178 of file ParsedMinerule.hpp.

6.105.5.9 filter_condition

```
std::string minerule::ParsedMinerule::filter_condition
```

Definition at line 185 of file ParsedMinerule.hpp.

6.105.5.10 ga

```
AttrVector minerule::ParsedMinerule::ga
```

Definition at line 153 of file ParsedMinerule.hpp.

6.105.5.11 gc

```
list_OR_node* minerule::ParsedMinerule::gc
```

Definition at line 164 of file ParsedMinerule.hpp.

6.105.5.12 ha

```
AttrVector minerule::ParsedMinerule::ha
```

Definition at line 159 of file ParsedMinerule.hpp.

6.105.5.13 headCardinalities

```
MinMaxPair minerule::ParsedMinerule::headCardinalities
```

Definition at line 172 of file ParsedMinerule.hpp.

6.105.5.14 length

```
MinMaxPair minerule::ParsedMinerule::length
```

Definition at line 182 of file ParsedMinerule.hpp.

6.105.5.15 mc

```
list_OR_node* minerule::ParsedMinerule::mc
```

Definition at line 163 of file ParsedMinerule.hpp.

6.105.5.16 miningTask

```
MiningTasks minerule::ParsedMinerule::miningTask
```

Definition at line 188 of file ParsedMinerule.hpp.

6.105.5.17 oa

```
AttrVector minerule::ParsedMinerule::oa
```

Definition at line 154 of file ParsedMinerule.hpp.

6.105.5.18 ra

```
AttrVector minerule::ParsedMinerule::ra
```

Definition at line 156 of file ParsedMinerule.hpp.

6.105.5.19 seq_bem_vect

```
std::vector<Bem_cond*> minerule::ParsedMinerule::seq_bem_vect
```

Definition at line 184 of file ParsedMinerule.hpp.

6.105.5.20 seq_dist_vect

```
std::vector<Dist_cond*> minerule::ParsedMinerule::seq_dist_vect
```

Definition at line 183 of file ParsedMinerule.hpp.

6.105.5.21 sequenceAllowedGaps

```
MinMaxPair minerule::ParsedMinerule::sequenceAllowedGaps
```

Definition at line 181 of file ParsedMinerule.hpp.

6.105.5.22 sup

```
float minerule::ParsedMinerule::sup
```

Definition at line 169 of file ParsedMinerule.hpp.

6.105.5.23 tab_result

```
std::string minerule::ParsedMinerule::tab_result
```

Definition at line 174 of file ParsedMinerule.hpp.

6.105.5.24 tab_source

```
std::string minerule::ParsedMinerule::tab_source
```

Definition at line 173 of file ParsedMinerule.hpp.

6.105.5.25 tautologies

```
bool minerule::ParsedMinerule::tautologies
```

Definition at line 176 of file ParsedMinerule.hpp.

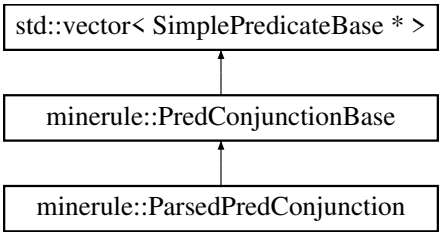
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedMinerule.hpp
- /Users/esposito/Software/minerule/src/Parsers/ParsedMinerule.cpp

6.106 minerule::ParsedPredConjunction Class Reference

```
#include <ParsedPredicate.hpp>
```

Inheritance diagram for minerule::ParsedPredConjunction:



Public Member Functions

- ParsedPredConjunction ()
- ParsedPredConjunction (const ParsedPredConjunction &rhs)
- virtual PredConjunctionBase * cloneInstance () const
- virtual PredConjunctionBase * newInstance () const
- PredicateBase * newPredicate () const
- PredConjunctionBase * newPredConjunction () const
- bool areAllBorH () const
- bool areAllAggr_f () const
- bool atLeastOneBorH () const
- PredConjunctionBase & operator&= (const PredConjunctionBase &p)
- PredicateBase & operator! () const
- bool find (SimplePredicateBase *sp) const
- virtual PredConjunctionBase * copy_and_append (SimplePredicateBase *sp)

Data Fields

- T elements

  STL member.

6.106.1 Detailed Description

This class represent a conjunction of SimplePredicate

Definition at line 108 of file ParsedPredicate.hpp.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 ParsedPredConjunction() [1/2]

```
minerule::ParsedPredConjunction::ParsedPredConjunction ( )  [inline]
```

Build a new ParsedPredConjunction that contain the old vector plus the new element and not modify the old vector The empty Constructor of the class

Definition at line 118 of file ParsedPredicate.hpp.

```
00118 : PredConjunctionBase() {};
```

6.106.2.2 ParsedPredConjunction() [2/2]

```
minerule::ParsedPredConjunction::ParsedPredConjunction (
            const ParsedPredConjunction & rhs )  [inline]
```

This is the copy constructor

Parameters

rhs	another object of this class

Definition at line 124 of file ParsedPredicate.hpp.

```
00124                                                                    {
00125       ParsedPredConjunction::const_iterator it;
00126       for (it=rhs.begin();it!=rhs.end();it++) {
00127         this->push_back( new ParsedSimplePredicate(dynamic_cast<const ParsedSimplePredicate&>(**it))
       );
00128       }
00129     }
```

6.106.3 Member Function Documentation

6.106.3.1 areAllAggr_f()

```
bool minerule::ParsedPredConjunction::areAllAggr_f ( ) const  [inline]
```

check if all the SimplePredicate are part of body or part of the head

Returns

true if only if all predicate are the aggregation function

Definition at line 169 of file ParsedPredicate.hpp.

```
00169                                {
00170       ParsedPredConjunction::const_iterator it;
00171       for (it=this->begin();
00172            it!=this->end() && dynamic_cast<const ParsedSimplePredicate&>(**it).isAggr_function();
00173            it++); /* note the empty body of the for loop */
00174       return it==this->end();
00175     }
```

6.106.3.2 areAllBorH()

```
bool minerule::ParsedPredConjunction::areAllBorH ( ) const  [inline]
```

check if all the SimplePredicate are part of body or part of the head

Returns

true if only if all predicate are part of body or head

Definition at line 156 of file ParsedPredicate.hpp.

```
00156                               {
00157       ParsedPredConjunction::const_iterator it;
00158       for (it=this->begin();
00159            it!=this->end() && dynamic_cast<const ParsedSimplePredicate&>(**it).isPartOfBorH();
00160            it++); /* note the empty body of the for loop */
00161
00162       return it==this->end();
00163     }
```

6.106.3.3 atLeastOneBorH()

```
bool minerule::ParsedPredConjunction::atLeastOneBorH ( ) const  [inline]
```

Check if at least One the SimplePredicate are part of body or part of the head

Returns

true at least One predicate is part of body or head

Definition at line 182 of file ParsedPredicate.hpp.

```
00182                                {
00183       ParsedPredConjunction::const_iterator it;
00184       for (it=this->begin();
00185            it!=this->end() && !dynamic_cast<const ParsedSimplePredicate&>(**it).isPartOfBorH();
00186            it++); /* note the empty body of the for loop */
00187
00188       return it!=this->end();
00189     }
```

6.106.3.4 cloneInstance()

```
virtual PredConjunctionBase * minerule::ParsedPredConjunction::cloneInstance ( ) const  [inline],
[virtual]
```

- this method is very important because permit to clone the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

  Returns

  PredConjunctioBase the object of super-class that incapsule a new object of this class

Reimplemented from minerule::PredConjunctionBase.

Definition at line 133 of file ParsedPredicate.hpp.

```
00133                                                {
00134       return new ParsedPredConjunction( *this );
00135     }
```

6.106.3.5 copy_and_append()

```
virtual PredConjunctionBase * minerule::PredConjunctionBase::copy_and_append (
             SimplePredicateBase * sp )  [inline], [virtual], [inherited]
```

Build a new PredConjunctionBaseParse that contain the old vector plus the new element and not modify the old vector

Parameters

*SimplePredicateBase**	sp the simple term that i want insert in the new PredConjunctionBase

Definition at line 213 of file PredicateBase.hpp.

```
00213                                                                    {
00214       PredConjunctionBase* conj =  this->cloneInstance();
00215       conj->push_back( sp->cloneInstance() );
00216       return conj;
00217     }
```

6.106.3.6 find()

```
bool minerule::PredConjunctionBase::find (
            SimplePredicateBase * sp ) const  [inherited]
```

Find if this PredConjunctionBase contains the SimplePredicateBase ∗sp .

Definition at line 84 of file PredicateBase.cpp.

```
00084                                                                {
00085     PredConjunctionBase::const_iterator it;
00086
00087     // Checking if sp is present in this PredConjunctionBase
00088     for (it=this->begin();
00089          it!=this->end() && **it != *sp ;
00090          it++); // note the body of the for is empty
00091
00092     return it!=this->end();
00093   }
```

6.106.3.7 newInstance()

```
virtual PredConjunctionBase * minerule::ParsedPredConjunction::newInstance ( ) const  [inline],
[virtual]
```

\ ∗ this method is very important because permit to create a new instance of the exact objcet in the hierachy of class \ ∗ because the RTTI mechanism use the right method with a specific type of object. if you want create a \ ∗ subclass of this class you must overriding this method for your class ∗

Returns

PredConjunctionBase the object of super-class that incapsule

Reimplemented from minerule::PredConjunctionBase.

Definition at line 138 of file ParsedPredicate.hpp.

```
00138                                                {
00139       return new ParsedPredConjunction( );
00140     }
```

6.106.3.8 newPredConjunction()

```
PredConjunctionBase * minerule::ParsedPredConjunction::newPredConjunction ( ) const  [virtual]
```

Reimplemented from minerule::PredConjunctionBase.

Definition at line 18 of file ParsedPredicate.cpp.

```
00018                                                            {
00019     return new ParsedPredConjunction();
00020   }
```

6.106.3.9 newPredicate()

```
PredicateBase * minerule::ParsedPredConjunction::newPredicate ( ) const  [virtual]
```

Reimplemented from minerule::PredConjunctionBase.

Definition at line 14 of file ParsedPredicate.cpp.

```
00014                                                 {
00015     return new ParsedPredicate();
00016   }
```

6.106.3.10 operator"!()

```
PredicateBase & minerule::PredConjunctionBase::operator!  ( ) const  [inherited]
```

Builds the negation of this PredConjunctionBase and return a PredicateBase object (in form disjunctive)

Returns

PredicateBase& the negation of this conjunction predicate

Definition at line 109 of file PredicateBase.cpp.

```
00109                                        {
00110     PredicateBase* result = newPredicate();
00111     PredConjunctionBase::const_iterator it;
00112     for(it=this->begin(); it!=this->end(); ++it) {
00113       (*result) |= !(**it);
00114     }
00115
00116     return *result;
00117   }
```

6.106.3.11 operator&=()

```
PredConjunctionBase & minerule::PredConjunctionBase::operator&= (
            const PredConjunctionBase & p )  [inherited]
```

Builds the logical and of *this and p and return *this.

Definition at line 97 of file PredicateBase.cpp.

```
00097                                                          {
00098     PredConjunctionBase::const_iterator it;
00099     for(it=rhs.begin(); it!=rhs.end(); ++it) {
00100       if( !this->find(*it) )
00101         this->push_back( (*it)->cloneInstance() );
00102     }
00103
00104     return *this;
00105   }
```

6.106.4 Field Documentation

6.106.4.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

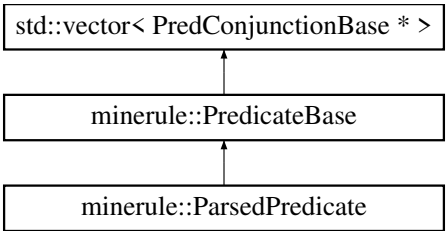
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedPredicate.hpp
- /Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp

6.107 minerule::ParsedPredicate Class Reference

```
#include <ParsedPredicate.hpp>
```

Inheritance diagram for minerule::ParsedPredicate:



Public Member Functions

- ParsedPredicate ()
- ParsedPredicate (const ParsedPredicate &rhs)
- virtual ParsedPredicate * cloneInstance () const
- virtual ParsedPredicate * newInstance () const
- PredicateBase * newPredicate () const
- PredConjunctionBase * newPredConjunction () const
- bool areAllBorH () const
- bool areAllAggr_f () const
- bool atLeastOneBorH () const
- list_OR_node * convert () const
- list_AND_node * convert_and_list (const PredConjunctionBase &) const
- PredicateBase & operator&= (const PredicateBase &p)
- PredicateBase & operator|= (const PredicateBase &p)
- void merge (PredicateBase *l2)
- PredicateBase & operator! () const
- void stamp ()

Data Fields

- T elements

  STL member.

6.107.1 Detailed Description

This class represent a disjunction of PredConjunction, the prefer form of rappresentation is the disjunction form

Definition at line 201 of file ParsedPredicate.hpp.

6.107.2 Constructor & Destructor Documentation

6.107.2.1 ParsedPredicate() [1/2]

```
minerule::ParsedPredicate::ParsedPredicate ( )  [inline]
```

The empty constructor

Definition at line 206 of file ParsedPredicate.hpp.
```
00206 : PredicateBase() {}
```

6.107.2.2 ParsedPredicate() [2/2]

```
minerule::ParsedPredicate::ParsedPredicate (
            const ParsedPredicate & rhs )  [inline]
```

The copy constructor

Parameters

rhs	another object of this class

Definition at line 212 of file ParsedPredicate.hpp.
```
00212 : PredicateBase(rhs) {}
```

6.107.3 Member Function Documentation

6.107.3.1 areAllAggr_f()

```
bool minerule::ParsedPredicate::areAllAggr_f ( ) const  [inline]
```

Check if all the SimplePredicate are part of body or part of the head

Returns

true if only if all predicate are aggregate function

Definition at line 255 of file ParsedPredicate.hpp.

```
00255                                {
00256       ParsedPredicate::const_iterator it;
00257       for (it=this->begin();
00258            it!=this->end() && dynamic_cast<const ParsedPredConjunction&>(**it).areAllAggr_f();
00259            it++); /* note the empty body of the for loop */
00260       return it==this->end();
00261     }
```

6.107.3.2 areAllBorH()

```
bool minerule::ParsedPredicate::areAllBorH ( ) const  [inline]
```

Check if all the SimplePredicate are part of body or part of the head

Returns

true if only if all predicate are part of body or head

Definition at line 240 of file ParsedPredicate.hpp.

```
00240                               {
00241       ParsedPredicate::const_iterator it;
00242
00243       for (it=this->begin();
00244            it!=this->end() && dynamic_cast<const ParsedPredConjunction&>(**it).areAllBorH();
00245            it++); /* note the empty body of the for loop */
00246
00247       return it==this->end();
00248     }
```

6.107.3.3 atLeastOneBorH()

```
bool minerule::ParsedPredicate::atLeastOneBorH ( ) const  [inline]
```

Check if at least One the SimplePredicate are part of body or part of the head

Returns

true at least One predicate is part of body or head

Definition at line 268 of file ParsedPredicate.hpp.

```
00268                                    {
00269       ParsedPredicate::const_iterator it;
00270
00271       for (it=this->begin();
00272            it!=this->end() && !dynamic_cast<const ParsedPredConjunction&>(**it).atLeastOneBorH();
00273            it++); /* note the empty body of the for loop */
00274
00275       return it!=this->end();
00276     }
```

6.107.3.4 cloneInstance()

```
virtual ParsedPredicate * minerule::ParsedPredicate::cloneInstance ( ) const  [inline], [virtual]
```

- this method is very important because permit to clone the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. *

Returns

PredicateBase the object of super-class that incapsule a new object of this class

Reimplemented from minerule::PredicateBase.

Definition at line 216 of file ParsedPredicate.hpp.

```
00216                                           {
00217       return new ParsedPredicate(*this);
00218     }
```

6.107.3.5 convert()

```
list_OR_node * minerule::ParsedPredicate::convert ( ) const
```

Convert the ParsedPredicate to the "equivalent" c-structure.

Definition at line 61 of file ParsedPredicate.cpp.

```
00061                                          {
00062     PredicateBase::const_iterator it_or;
00063     list_OR_node* list_or = NULL;
00064     list_OR_node* or_head = NULL;
00065 
00066     for( it_or = this->begin(); it_or!=this->end(); ++it_or ) {
00067       if(list_or==NULL) {
00068         list_or = (list_OR_node*) malloc( sizeof(list_OR_node) );
00069         list_or->next=NULL;
00070         or_head=list_or;
00071       } else {
00072         list_or->next = (list_OR_node*) malloc( sizeof(list_OR_node) );
00073         list_or = list_or->next;
00074         list_or->next=NULL;
00075       }
00076 
00077       list_or->l_and = convert_and_list(**it_or);
00078     }
00079 
00080     return or_head;
00081   }
```

6.107.3.6 convert_and_list()

```
list_AND_node * minerule::ParsedPredicate::convert_and_list (
            const PredConjunctionBase & conj ) const
```

Definition at line 35 of file ParsedPredicate.cpp.

```
00035                                                                                {
00036     PredConjunctionBase::const_iterator it_and;
00037     list_AND_node* list_and = NULL;
00038     list_AND_node* and_head = NULL;
00039
00040     for( it_and = conj.begin(); it_and!=conj.end(); ++it_and ) {
00041       if(list_and==NULL) {
00042         list_and = (list_AND_node*)malloc( sizeof(list_AND_node) );
00043         list_and->next = NULL;
00044         and_head = list_and;
00045       } else {
00046         list_and->next = (list_AND_node*) malloc( sizeof( list_AND_node ) );
00047         list_and = list_and->next;
00048         list_and->next = NULL;
00049       }
00050
00051       list_and->sp = (simple_pred*) malloc(sizeof(simple_pred));
00052       list_and->sp->val1=strdup((*it_and)->getVal1().c_str());
00053       list_and->sp->op=strdup((*it_and)->getOp().c_str());
00054       list_and->sp->val2=strdup((*it_and)->getVal2().c_str());
00055     }
00056
00057     return and_head;
00058   }
```

6.107.3.7 merge()

```
void minerule::PredicateBase::merge (
            PredicateBase * l2 )  [inline], [inherited]
```

merge unordered the two vector into the first and delete all the element of the vector pass as parameter

Definition at line 286 of file PredicateBase.hpp.

```
00286                                  {
00287       copy( l2->begin(), l2->end(), std::back_insert_iterator<PredicateBase>(*this));
00288       erase(l2->begin(), l2->end());
00289
00290       /*    Predicate::iterator it_and;
00291             it_and=l2->begin();
00292             while (it_and!=l2->end()){
00293             this->push_back(*it_and);
00294             *it_and=NULL;
00295             it_and++;
00296             }*/
00297     }
```

6.107.3.8 newInstance()

```
virtual ParsedPredicate * minerule::ParsedPredicate::newInstance ( ) const  [inline], [virtual]
```

- this method is very important because permit to create a new instance of the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. *

  Returns

  PredidcateBase the object of super-class that incapsule

Reimplemented from minerule::PredicateBase.

Definition at line 222 of file ParsedPredicate.hpp.

```
00222                                  {
00223       return new ParsedPredicate();
00224     }
```

6.107.3.9 newPredConjunction()

```
PredConjunctionBase * minerule::ParsedPredicate::newPredConjunction ( ) const  [virtual]
```

Reimplemented from minerule::PredicateBase.

Definition at line 26 of file ParsedPredicate.cpp.

```
00026                                                    {
00027     return new ParsedPredConjunction();
00028   }
```

6.107.3.10 newPredicate()

```
PredicateBase * minerule::ParsedPredicate::newPredicate ( ) const  [virtual]
```

Reimplemented from minerule::PredicateBase.

Definition at line 22 of file ParsedPredicate.cpp.

```
00022                                                {
00023     return new ParsedPredicate();
00024   }
```

6.107.3.11 operator"!()

```
PredicateBase & minerule::PredicateBase::operator!  ( ) const  [inherited]
```

- Builds the logical negation but i return another object of this class (in disjunction form)

Definition at line 185 of file PredicateBase.cpp.

```
00185                                  {
00186     assert( this->size() != 0 );
00187
00188     PredicateBase* result = newPredicate();
00189     PredicateBase::const_iterator it=this->begin();
00190
00191     (*result) |= !(**it);
00192     ++it;
00193
00194     for(; it!=this->end(); ++it ) {
00195       (*result) &= !(**it);
00196     }
00197
00198     return *result;
00199   }
```

6.107.3.12 operator&=()

```
PredicateBase & minerule::PredicateBase::operator&= (
            const PredicateBase & p )  [inherited]
```

Builds the logical and of ∗this and p and returns ∗this

Definition at line 151 of file PredicateBase.cpp.

```
00151                                                                {
00152     size_t oldSize = this->size();
00153     // making room for the new OR predicates
00154     PredicateBase* thisCopy = this->cloneInstance();
00155     for(size_t i=0; i<p.size()-1; i++) {
00156       *this |= *thisCopy;
00157     }
00158
00159     // building the conjunctions (now there are oldSize*p.size() conj
00160     // in this predicate. The first oldSize ones need to be conjuncted
00161     // with p[0], the second oldSize ones with p[1]... the p.size()'th
00162     // oldSize ones with p[p.size()-1].
00163     for(size_t i=0; i<p.size(); i++) {
00164       for(size_t j=0; j<oldSize; j++) {
00165         assert(j+i*oldSize < this->size() );
00166         //      std::cout « "Pred("«i«","«j«"):" « *(*this)[j+i*oldSize] « std::endl;
00167         *(*this)[j+i*oldSize] &= *p[i];
00168       }
00169     }
00170
00171     return *this;
00172   }
```

6.107.3.13 operator"|=()

```
PredicateBase & minerule::PredicateBase::operator|= (
            const PredicateBase & p )  [inherited]
```

Builds the logical or of ∗this and p and returns ∗this

Definition at line 175 of file PredicateBase.cpp.

```
00175                                                    {
00176     for(PredicateBase::const_iterator it=p.begin(); it!=p.end(); it++) {
00177       push_back( (*it)->cloneInstance() );
00178     }
00179
00180     return *this;
00181   }
```

6.107.3.14 stamp()

```
void minerule::PredicateBase::stamp ( )  [inline], [inherited]
```

This is a simple function that print a PredicateBase object

Definition at line 305 of file PredicateBase.hpp.

```
00305                {
00306       if (this!=NULL){
00307         int c_and=0;
00308         int c_or=0;
00309         PredicateBase::iterator it_and = this->begin();
00310         PredConjunctionBase::iterator it_sp;
00311         while (it_and!=this->end()){
00312           if (c_or) std::cout«"\n OR \n";
00313           it_sp=(*it_and)->begin();
00314           c_and=0;
```

```
00315           while(it_sp!=(*it_and)->end()){
00316             if (c_and) std::cout«" AND ";
00317             std::cout«(*it_sp)->getVal1()«" ";
00318             std::cout«(*it_sp)->getOp()«" ";
00319             std::cout«(*it_sp)->getVal2();
00320             it_sp++;
00321             c_and++;
00322           }
00323           //delete *it_and;
00324           it_and++;
00325           c_or++;
00326         }
00327         std::cout«"\n\n\n\n";
00328       }
00329
00330     }
```

6.107.4 Field Documentation

6.107.4.1 elements

`T std::vector< T >::elements [inherited]`

STL member.

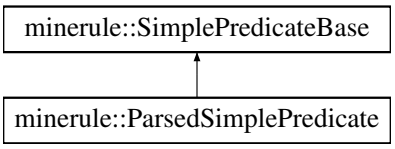
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedPredicate.hpp
- /Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp

6.108 minerule::ParsedSimplePredicate Class Reference

`#include <ParsedPredicate.hpp>`

Inheritance diagram for minerule::ParsedSimplePredicate:



Public Member Functions

- ParsedSimplePredicate ()
- ParsedSimplePredicate (const std::string &val1, const std::string &op, const std::string &val2, bool BorH, bool aggr)
- ParsedSimplePredicate (const ParsedSimplePredicate &rhs)
- virtual SimplePredicateBase * cloneInstance () const
- virtual SimplePredicateBase * newInstance () const
- PredicateBase * newPredicate () const
- PredConjunctionBase * newPredConjunction () const
- bool isPartOfBorH () const
- bool isAggr_function () const
- const std::string & getVal1 () const
- const std::string & getVal2 () const
- const std::string & getOp () const
- virtual bool operator== (const SimplePredicateBase &rhs) const
- virtual bool operator!= (const SimplePredicateBase &rhs) const
- virtual bool operator< (const SimplePredicateBase &rhs) const
- virtual PredicateBase & operator! () const

6.108.1 Detailed Description

This class represent a single comparison with two terms the attribute aggr_f is marked as true if this predicate contains an aggregation functions for example SUM, AVG, STD, ...

Definition at line 24 of file ParsedPredicate.hpp.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 ParsedSimplePredicate() [1/3]

```
minerule::ParsedSimplePredicate::ParsedSimplePredicate ( )  [inline]
```

The constructor of the class without parameter

Definition at line 32 of file ParsedPredicate.hpp.

```
00032 : SimplePredicateBase(), partOfBorH(false), aggr_f(false) {}
```

6.108.2.2 ParsedSimplePredicate() [2/3]

```
minerule::ParsedSimplePredicate::ParsedSimplePredicate (
            const std::string & val1,
            const std::string & op,
            const std::string & val2,
            bool BorH,
            bool aggr )  [inline]
```

This constructor take three parameter, two terms and an operator of comparison

Parameters

val1	the first term
op	the comparison operator
val2	the second term

Definition at line 41 of file ParsedPredicate.hpp.

```
00041
                :
00042       SimplePredicateBase(val1, op, val2), partOfBorH(BorH), aggr_f(aggr){}
```

6.108.2.3 ParsedSimplePredicate() [3/3]

```
minerule::ParsedSimplePredicate::ParsedSimplePredicate (
            const ParsedSimplePredicate & rhs )  [inline]
```

This is the copy constructor

Parameters

rhs	another ParsedSimplePredicate obj

Definition at line 49 of file ParsedPredicate.hpp.

```
00049                                                   :
00050       SimplePredicateBase(rhs), partOfBorH(rhs.partOfBorH), aggr_f(rhs.aggr_f) {}
```

6.108.3 Member Function Documentation

6.108.3.1 cloneInstance()

```
virtual SimplePredicateBase * minerule::ParsedSimplePredicate::cloneInstance ( ) const  [inline],
[virtual]
```

this method is very important because permit to clone the exact objcet in the hierachy of class because the RTTI mechanism use the right method with a specific type of object

Returns

SimplePredicateBase the object of super-class that incapsule a new object of this class

Reimplemented from minerule::SimplePredicateBase.

Definition at line 58 of file ParsedPredicate.hpp.

```
00058                                           {
00059       return new ParsedSimplePredicate(*this);
00060     }
```

6.108.3.2 getOp()

```
const std::string & minerule::SimplePredicateBase::getOp ( ) const  [inline], [inherited]
```

This method return a pointer to comparison operator of Simple Predicate

Returns

std::string& op the comparison operator

Definition at line 103 of file PredicateBase.hpp.

```
00103                                       {
00104       return op;
00105     }
```

6.108.3.3 getVal1()

```
const std::string & minerule::SimplePredicateBase::getVal1 ( ) const  [inline], [inherited]
```

This method return a pointer to first term of Simple Predicate

Returns

std::string& val1 the firt term

Definition at line 87 of file PredicateBase.hpp.

```
00087                                       {
00088       return val1;
00089     }
```

6.108.3.4 getVal2()

```
const std::string & minerule::SimplePredicateBase::getVal2 ( ) const  [inline], [inherited]
```

This method return a pointer to second term of Simple Predicate

Returns

std::string& val2 the second term

Definition at line 95 of file PredicateBase.hpp.

```
00095                                       {
00096       return val2;
00097     }
```

6.108.3.5 isAggr_function()

```
bool minerule::ParsedSimplePredicate::isAggr_function ( ) const  [inline]
```

this method control if the simplePredicate contains a BODY or HEAD attribute

Returns

true if this object represent an aggregation function as SUM, STD, AVG, ecc..

Definition at line 95 of file ParsedPredicate.hpp.

```
00095                                  {
00096       return aggr_f;
00097     }
```

6.108.3.6 isPartOfBorH()

```
bool minerule::ParsedSimplePredicate::isPartOfBorH ( ) const  [inline]
```

this method control if the simplePredicate contains a BODY or HEAD attribute

Returns

true if this object contains a BODY or HEAD attribute, false in other situation

Definition at line 87 of file ParsedPredicate.hpp.

```
00087                                 {
00088       return partOfBorH;
00089     }
```

6.108.3.7 newInstance()

```
virtual SimplePredicateBase * minerule::ParsedSimplePredicate::newInstance ( ) const  [inline],
[virtual]
```

this method is very important because permit to create a new instance of the exact objcet in the hierachy of class because the RTTI mechanism use the right method with a specific type of object

Returns

SimplePredicateBase the object of super-class that incapsule

Reimplemented from minerule::SimplePredicateBase.

Definition at line 68 of file ParsedPredicate.hpp.

```
00068                                          {
00069         return new ParsedSimplePredicate();
00070     }
```

6.108.3.8 newPredConjunction()

```
PredConjunctionBase * minerule::ParsedSimplePredicate::newPredConjunction ( ) const  [virtual]
```

Reimplemented from minerule::SimplePredicateBase.

Definition at line 10 of file ParsedPredicate.cpp.

```
00010                                                          {
00011     return new ParsedPredConjunction();
00012   }
```

6.108.3.9 newPredicate()

```
PredicateBase * minerule::ParsedSimplePredicate::newPredicate ( ) const  [virtual]
```

Reimplemented from minerule::SimplePredicateBase.

Definition at line 6 of file ParsedPredicate.cpp.

```
00006                                                  {
00007     return new ParsedPredicate();
00008   }
```

6.108.3.10 operator"!()

```
PredicateBase & minerule::SimplePredicateBase::operator!  ( ) const  [virtual], [inherited]
```

Define a method that return a new PredicateBase& that contains the negation of this Simple Predicate

Returns

PredicateBase& that contains the negation of this Simple Predicate

Definition at line 37 of file PredicateBase.cpp.

```
00037                                          {
00038    std::string nop;
00039     if (op==">") { nop="<="; }
00040     else if (op==">=") { nop="<"; }
00041     else if (op=="<") { nop=">="; }
00042     else if (op=="<=") { nop=">"; }
00043     else if (op=="=") { nop="<>"; }
00044     else if (op=="<>") { nop="="; }
00045     SimplePredicateBase* sp= this->cloneInstance();
00046     PredConjunctionBase* pc= this->newPredConjunction();
00047     PredicateBase*       pb= this->newPredicate();
00048
00049     sp->op = nop;
00050     pc->push_back(sp);
00051     pb->push_back(pc);
00052
00053     return *pb;
00054   }
```

6.108.3.11 operator"!=()

```
virtual bool minerule::SimplePredicateBase::operator!= (
            const SimplePredicateBase & rhs ) const  [inline], [virtual], [inherited]
```

Redefine the operator !=

Returns

true if at least one of the part of Simple Predicate ar different

Definition at line 122 of file PredicateBase.hpp.

```
00122                                                                    {
00123       return
00124         val1!=rhs.val1 ||
00125         val2!=rhs.val2 ||
00126         op!=rhs.op;
00127     }
```

6.108.3.12 operator<()

```
virtual bool minerule::SimplePredicateBase::operator< (
            const SimplePredicateBase & rhs ) const  [inline], [virtual], [inherited]
```

Redefine the < operator

Returns

true if the first term is lesser than second

Definition at line 133 of file PredicateBase.hpp.

```
00133                                                                {
00134       return val1+op+val2 < rhs.val1+rhs.op+rhs.val2;
00135     }
```

6.108.3.13 operator==()

```
virtual bool minerule::SimplePredicateBase::operator== (
            const SimplePredicateBase & rhs ) const  [inline], [virtual], [inherited]
```

Redefine the operator ==

Returns

true if and only if all the part of Simple Predicate are equals

Definition at line 111 of file PredicateBase.hpp.

```
00111                                                                    {
00112       return
00113         val1==rhs.val1 &&
00114         val2==rhs.val2 &&
00115         op==rhs.op;
00116     }
```

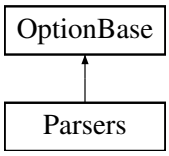
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/ParsedPredicate.hpp
- /Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp

6.109 Parsers Class Reference

`#include <parsers.hpp>`

Inheritance diagram for Parsers:



Public Member Functions

- Parsers ()
- virtual ~Parsers ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- void setLogFILE (const std::string &fname)
- void setLogOnStdout ()
- void setLogOnStderr ()
- const FILE * getLogFILE () const
- void setMinBodyElems (int m)
- void setMaxBodyElems (int M)
- void setMinHeadElems (int m)
- void setMaxHeadElems (int M)
- const MinMaxPair & getBodyCardinalities () const
- const MinMaxPair & getHeadCardinalities () const

6.109.1 Detailed Description

Definition at line 16 of file parsers.hpp.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 Parsers()

`Parsers::Parsers ( ) [inline]`

Definition at line 28 of file parsers.hpp.

```
00028             : logfile(NULL),
00029         bodyCardinalities(MinMaxPair(1,1000)),
00030         headCardinalities(MinMaxPair(1,1000)) {};
```

6.109.2.2 ~Parsers()

```
virtual Parsers::~Parsers ( )  [inline], [virtual]
```

Definition at line 32 of file parsers.hpp.

```
00032                       {
00033         clearStream();
00034     }
```

6.109.3 Member Function Documentation

6.109.3.1 className()

```
virtual std::string Parsers::className ( ) const  [inline], [virtual]
```

Definition at line 36 of file parsers.hpp.

```
00036                                       {
00037         return "parsers";
00038     }
```

6.109.3.2 getBodyCardinalities()

```
const MinMaxPair & Parsers::getBodyCardinalities ( ) const  [inline]
```

Definition at line 68 of file parsers.hpp.

```
00068                                                  {
00069         return bodyCardinalities;
00070     }
```

6.109.3.3 getHeadCardinalities()

```
const MinMaxPair & Parsers::getHeadCardinalities ( ) const  [inline]
```

Definition at line 72 of file parsers.hpp.

```
00072                                                  {
00073         return headCardinalities;
00074     }
```

6.109.3.4 getLogFILE()

```
const FILE * Parsers::getLogFILE ( ) const  [inline]
```

Definition at line 47 of file parsers.hpp.

```
00047                                    {
00048         assert( logfile!=NULL );
00049         return logfile;
00050     }
```

6.109.3.5 setLogFILE()

```
void Parsers::setLogFILE (
            const std::string & fname )
```

6.109.3.6 setLogOnStderr()

```
void Parsers::setLogOnStderr ( )
```

6.109.3.7 setLogOnStdout()

```
void Parsers::setLogOnStdout ( )
```

6.109.3.8 setMaxBodyElems()

```
void Parsers::setMaxBodyElems (
            int M )  [inline]
```

Definition at line 56 of file parsers.hpp.

```
00056                                {
00057         bodyCardinalities.setMax(M);
00058     }
```

6.109.3.9 setMaxHeadElems()

```
void Parsers::setMaxHeadElems (
            int M )  [inline]
```

Definition at line 64 of file parsers.hpp.

```
00064                                {
00065         headCardinalities.setMax(M);
00066     }
```

6.109.3.10 setMinBodyElems()

```
void Parsers::setMinBodyElems (
            int m )  [inline]
```

Definition at line 52 of file parsers.hpp.

```
00052                                {
00053         bodyCardinalities.setMin(m);
00054     }
```

6.109.3.11 setMinHeadElems()

```
void Parsers::setMinHeadElems (
            int m )  [inline]
```

Definition at line 60 of file parsers.hpp.

```
00060                                {
00061         headCardinalities.setMin(m);
00062     }
```

6.109.3.12 setOption()

```
virtual void Parsers::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

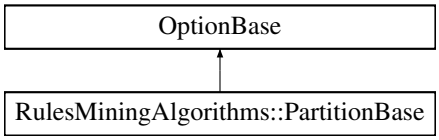
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/parsers.hpp

6.110 RulesMiningAlgorithms::PartitionBase Class Reference

```
#include <rulemining.hpp>
```

Inheritance diagram for RulesMiningAlgorithms::PartitionBase:



Public Member Functions

- virtual ~PartitionBase ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- unsigned int getRowsPerPartition () const
- void setRowsPerPartition (unsigned int num)

6.110.1 Detailed Description

Definition at line 18 of file rulemining.hpp.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 ~PartitionBase()

```
virtual RulesMiningAlgorithms::PartitionBase::~PartitionBase ( )  [inline], [virtual]
```

Definition at line 21 of file rulemining.hpp.

```
00021 {};
```

6.110.3 Member Function Documentation

6.110.3.1 className()

```
virtual std::string RulesMiningAlgorithms::PartitionBase::className ( ) const  [inline], [virtual]
```

Definition at line 23 of file rulemining.hpp.

```
00023                                                {
00024             return "partitionbase";
00025           }
```

6.110.3.2 getRowsPerPartition()

```
unsigned int RulesMiningAlgorithms::PartitionBase::getRowsPerPartition ( ) const  [inline]
```

Definition at line 29 of file rulemining.hpp.

```
00029                                                  {
00030             return rowsPerPartition;
00031           }
```

6.110.3.3 setOption()

```
virtual void RulesMiningAlgorithms::PartitionBase::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.110.3.4 setRowsPerPartition()

```
void RulesMiningAlgorithms::PartitionBase::setRowsPerPartition (
            unsigned int num )  [inline]
```

Definition at line 32 of file rulemining.hpp.

```
00032                                                    {
00033             rowsPerPartition = num;
00034           }
```

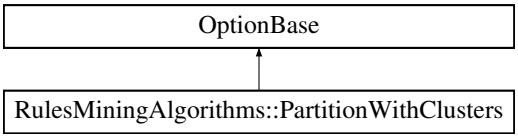
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp

6.111 RulesMiningAlgorithms::PartitionWithClusters Class Reference

`#include <rulemining.hpp>`

Inheritance diagram for RulesMiningAlgorithms::PartitionWithClusters:



Public Member Functions

- virtual ~PartitionWithClusters ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- unsigned int getRowsPerPartition () const
- void setRowsPerPartition (unsigned int num)

6.111.1 Detailed Description

Definition at line 37 of file rulemining.hpp.

6.111.2 Constructor & Destructor Documentation

6.111.2.1 ~PartitionWithClusters()

`virtual RulesMiningAlgorithms::PartitionWithClusters::~PartitionWithClusters ( ) [inline], [virtual]`

Definition at line 40 of file rulemining.hpp.

```
00040 {};
```

6.111.3 Member Function Documentation

6.111.3.1 className()

`virtual std::string RulesMiningAlgorithms::PartitionWithClusters::className ( ) const [inline], [virtual]`

Definition at line 42 of file rulemining.hpp.

```
00042                                          {
00043             return "partitionwithclusters";
00044           }
```

6.111.3.2 getRowsPerPartition()

```
unsigned int RulesMiningAlgorithms::PartitionWithClusters::getRowsPerPartition ( ) const  [inline]
```

Definition at line 48 of file rulemining.hpp.

```
00048                                                {
00049             return rowsPerPartition;
00050           }
```

6.111.3.3 setOption()

```
virtual void RulesMiningAlgorithms::PartitionWithClusters::setOption (
             const std::string & name,
             const std::string & value )  [virtual]
```

6.111.3.4 setRowsPerPartition()

```
void RulesMiningAlgorithms::PartitionWithClusters::setRowsPerPartition (
             unsigned int num )  [inline]
```

Definition at line 51 of file rulemining.hpp.

```
00051                                                 {
00052             rowsPerPartition = num;
00053           }
```

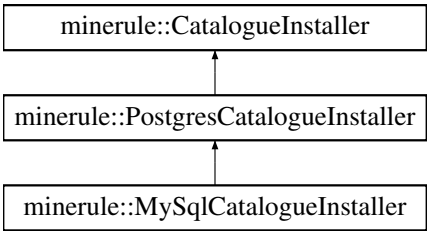
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp

6.112 minerule::PostgresCatalogueInstaller Class Reference

```
#include <PostgresCatalogueInstaller.hpp>
```

Inheritance diagram for minerule::PostgresCatalogueInstaller:



Public Types

- enum SupportedDbms { MySql , Postgres }

Public Member Functions

- PostgresCatalogueInstaller ()
- virtual ~PostgresCatalogueInstaller ()
- virtual void installMRQuery ()
- virtual void installMRAttList ()
- virtual void installMREqKeys ()
- virtual void installMREqKeysCol ()
- virtual void installMRDepFun ()
- virtual void installMRDepFunCol ()
- virtual void installMRAutoincrement ()
- virtual void initializeAutoincrement ()
- virtual void dropMRQuery ()
- virtual void dropMRAttList ()
- virtual void dropMREqKeys ()
- virtual void dropMREqKeysCol ()
- virtual void dropMRDepFun ()
- virtual void dropMRDepFunCol ()
- virtual void dropMRAutoincrement ()
- virtual void install ()
- virtual void uninstall ()

Static Public Member Functions

- static CatalogueInstaller * newInstaller (SupportedDbms dbms)
- static CatalogueInstaller * newInstaller ()

Protected Attributes

- mrdb::Connection * _connection
- mrdb::Statement * _statement

6.112.1 Detailed Description

Definition at line 22 of file PostgresCatalogueInstaller.hpp.

6.112.2 Member Enumeration Documentation

6.112.2.1 SupportedDbms

```
enum minerule::CatalogueInstaller::SupportedDbms  [inherited]
```

Enumerator

MySql	
Postgres	

Definition at line 29 of file CatalogueInstaller.hpp.
```
00029 { MySql, Postgres } SupportedDbms;
```

6.112.3 Constructor & Destructor Documentation

6.112.3.1 PostgresCatalogueInstaller()

```
minerule::PostgresCatalogueInstaller::PostgresCatalogueInstaller ( )  [inline]
```

Definition at line 24 of file PostgresCatalogueInstaller.hpp.
```
00024 : CatalogueInstaller() {}
```

6.112.3.2 ~PostgresCatalogueInstaller()

```
virtual minerule::PostgresCatalogueInstaller::~PostgresCatalogueInstaller ( )  [inline], [virtual]
```

Definition at line 25 of file PostgresCatalogueInstaller.hpp.
```
00025 {};
```

6.112.4 Member Function Documentation

6.112.4.1 dropMRAttList()

```
void minerule::PostgresCatalogueInstaller::dropMRAttList ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 24 of file PostgresCatalogueInstaller.cpp.
```
00024                                                   {
00025                 _statement->execute("DROP TABLE IF EXISTS mr_att_lists");
00026         }
```

6.112.4.2 dropMRAutoincrement()

```
void minerule::PostgresCatalogueInstaller::dropMRAutoincrement ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 44 of file PostgresCatalogueInstaller.cpp.
```
00044                                                       {
00045                 _statement->execute("DROP TABLE IF EXISTS mr_autoincrement");
00046         }
```

6.112.4.3 dropMRDepFun()

```
void minerule::PostgresCatalogueInstaller::dropMRDepFun ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 36 of file PostgresCatalogueInstaller.cpp.

```
00036                                                       {
00037                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun");
00038         }
```

6.112.4.4 dropMRDepFunCol()

```
void minerule::PostgresCatalogueInstaller::dropMRDepFunCol ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 40 of file PostgresCatalogueInstaller.cpp.

```
00040                                                          {
00041                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun_col");
00042         }
```

6.112.4.5 dropMREqKeys()

```
void minerule::PostgresCatalogueInstaller::dropMREqKeys ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 28 of file PostgresCatalogueInstaller.cpp.

```
00028                                                       {
00029                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys");
00030         }
```

6.112.4.6 dropMREqKeysCol()

```
void minerule::PostgresCatalogueInstaller::dropMREqKeysCol ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 32 of file PostgresCatalogueInstaller.cpp.

```
00032                                                          {
00033                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys_col");
00034         }
```

6.112.4.7 dropMRQuery()

```
void minerule::PostgresCatalogueInstaller::dropMRQuery ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 20 of file PostgresCatalogueInstaller.cpp.

```
00020                                                       {
00021                 _statement->execute("DROP TABLE IF EXISTS mr_query");
00022         }
```

6.112.4.8 initializeAutoincrement()

```
void minerule::PostgresCatalogueInstaller::initializeAutoincrement ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 118 of file PostgresCatalogueInstaller.cpp.

```
00118                                                                {
00119                 _statement->execute("INSERT INTO mr_autoincrement (table_name, current_id) VALUES
      ('mr_att_lists',0),('mr_eq_keys',0),('mr_query',0)");
00120         }
```

6.112.4.9 install()

```
virtual void minerule::CatalogueInstaller::install ( )  [inline], [virtual], [inherited]
```

Definition at line 57 of file CatalogueInstaller.hpp.

```
00057                                {
00058                         uninstall();
00059
00060                         installMRQuery();
00061                         installMRAttList();
00062                         installMREqKeys();
00063                         installMREqKeysCol();
00064                         installMRDepFun();
00065                         installMRDepFunCol();
00066                         installMRAutoincrement();
00067                         initializeAutoincrement();
00068                 }
```

6.112.4.10 installMRAttList()

```
void minerule::PostgresCatalogueInstaller::installMRAttList ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 66 of file PostgresCatalogueInstaller.cpp.

```
00066                                                       {
00067                 _statement->execute("CREATE TABLE mr_att_lists (att_list_id  integer NOT NULL,col_name
      varchar(128) NOT NULL)");
00068                 _statement->execute("CREATE INDEX mr_att_lists_att_list_id_index ON
      mr_att_lists(att_list_id)");
00069         }
```

6.112.4.11 installMRAutoincrement()

```
void minerule::PostgresCatalogueInstaller::installMRAutoincrement ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 109 of file PostgresCatalogueInstaller.cpp.

```
00109                                                       {
00110                 _statement->execute("CREATE TABLE mr_autoincrement (table_name varchar(128) NOT NULL,"
00111                                                     "current_id integer NOT NULL,"
00112                                                     "CONSTRAINT mr_autoincrement_primary PRIMARY
      KEY (table_name))");
00113
00114         }
```

6.112.4.12 installMRDepFun()

```
void minerule::PostgresCatalogueInstaller::installMRDepFun ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 91 of file PostgresCatalogueInstaller.cpp.

```
00091                                                   {
00092                 _statement->execute("CREATE TABLE mr_dep_fun (lhs_tab_name  varchar(128) NOT NULL,"
00093                                                         "lhs_att_list_id  integer NOT NULL,"
00094                                                         "rhs_tab_name  varchar(128) NOT NULL,"
00095                                                         "rhs_att_list_id  integer NOT NULL,"
00096                                                         "order_type char(1),"
00097                                                         "CONSTRAINT mr_dep_fun_primary PRIMARY
      KEY(lhs_tab_name,lhs_att_list_id,rhs_tab_name, rhs_att_list_id))");
00098         }
```

6.112.4.13 installMRDepFunCol()

```
void minerule::PostgresCatalogueInstaller::installMRDepFunCol ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 102 of file PostgresCatalogueInstaller.cpp.

```
00102                                                       {
00103                 _statement->execute("CREATE TABLE mr_dep_fun_col (col_id  integer NOT NULL, col_name
      varchar(128) NOT NULL)");
00104                 _statement->execute("CREATE INDEX mr_dep_fun_col_col_id_index ON
      mr_dep_fun_col(col_id)");
00105         }
```

6.112.4.14 installMREqKeys()

```
void minerule::PostgresCatalogueInstaller::installMREqKeys ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 73 of file PostgresCatalogueInstaller.cpp.

```
00073                                                   {
00074                 _statement->execute("CREATE TABLE mr_eq_keys (tab_name  varchar(128) NOT NULL,"
00075                                                         "key_id  integer NOT NULL,"
00076                                                         "ref_key_id  integer NOT NULL,"
00077                                                         "order_type char(1),"
00078                                                         "CONSTRAINT mr_eq_keys_primary PRIMARY
      KEY(tab_name,key_id,ref_key_id))");
00079         }
```

6.112.4.15 installMREqKeysCol()

```
void minerule::PostgresCatalogueInstaller::installMREqKeysCol ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 83 of file PostgresCatalogueInstaller.cpp.

```
00083                                                    {
00084                 _statement->execute("CREATE TABLE mr_eq_keys_col (key_id  integer NOT NULL,"
00085                                                          "col_name  varchar(128) NOT NULL)");
00086                 _statement->execute("CREATE INDEX mr_eq_keys_col_key_id_index ON
      mr_eq_keys_col(key_id)");
00087         }
```

6.112.4.16 installMRQuery()

```
void minerule::PostgresCatalogueInstaller::installMRQuery ( )  [virtual]
```

Implements minerule::CatalogueInstaller.

Definition at line 50 of file PostgresCatalogueInstaller.cpp.

```
00050                                                    {
00051                 _statement->execute("CREATE TABLE mr_query (query_id  integer NOT NULL PRIMARY KEY,"
00052                                                          "query_text  text NOT NULL,"
00053                                                          "query_name  varchar(128) NOT NULL,     "
00054                                                          "tab_results_name varchar(128) NOT NULL,"
00055                                                          "source_tab_name varchar(128) NOT NULL,"
00056                                                          "gal  integer NOT NULL,"
00057                                                          "ral  integer NOT NULL,"
00058                                                                  "cal  integer)");
00059                 _statement->execute("CREATE INDEX mr_query_tab_results_name_index ON
      mr_query(tab_results_name)");
00060                 _statement->execute("CREATE INDEX mr_query_query_name_index ON mr_query
      (query_name)");
00061                 _statement->execute("CREATE INDEX mr_query_source_tab_name_index ON mr_query
      (source_tab_name)");
00062         }
```

6.112.4.17 newInstaller() [1/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller ( )  [static], [inherited]
```

Definition at line 34 of file CatalogueInstaller.cpp.

```
00034                                                    {
00035                 std::string dbmsStr = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00036
00037                 if(dbmsStr == "mysql") return newInstaller(MySql);
00038                 if(dbmsStr == "postgres") return newInstaller(Postgres);
00039
00040                 throw MineruleException(MR_ERROR_OPTION_CONFIGURATION,std::string("Option mrdb::dbms
      is set to an unsupported value.") +
00041                         " Current value is:"+dbmsStr+ ". Supported values are 'postgres' and
      'mysql'.");
00042         }
```

6.112.4.18 newInstaller() [2/2]

```
CatalogueInstaller * minerule::CatalogueInstaller::newInstaller (
            SupportedDbms dbms )  [static], [inherited]
```

Definition at line 25 of file CatalogueInstaller.cpp.

```
00025                                                                   {
00026                 switch(dbms) {
00027                 case MySql:
00028                         return new MySqlCatalogueInstaller();
00029                 case Postgres:
00030                         return new PostgresCatalogueInstaller();
00031                 }
00032         }
```

6.112.4.19 uninstall()

```
virtual void minerule::CatalogueInstaller::uninstall ( )  [inline], [virtual], [inherited]
```

Definition at line 70 of file CatalogueInstaller.hpp.

```
00070                                 {
00071                         dropMRQuery();
00072                         dropMRAttList();
00073                         dropMREqKeys();
00074                         dropMREqKeysCol();
00075                         dropMRDepFun();
00076                         dropMRDepFunCol();
00077                         dropMRAutoincrement();
00078                 }
```

6.112.5 Field Documentation

6.112.5.1 _connection

```
mrdb::Connection* minerule::CatalogueInstaller::_connection  [protected], [inherited]
```

Definition at line 26 of file CatalogueInstaller.hpp.

6.112.5.2 _statement

```
mrdb::Statement* minerule::CatalogueInstaller::_statement  [protected], [inherited]
```

Definition at line 27 of file CatalogueInstaller.hpp.

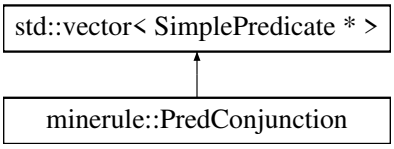
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp
- /Users/esposito/Software/minerule/src/Optimizer/Installers/PostgresCatalogueInstaller.cpp

6.113 minerule::PredConjunction Class Reference

`#include <Predicate.hpp>`

Inheritance diagram for minerule::PredConjunction:



Public Member Functions

- PredConjunction ()
- PredConjunction (const list_AND_node *)
- PredConjunction (const PredConjunction &)
- ~PredConjunction ()
- bool evaluate (const VarSet &) const
- PredConjunction & operator&= (const PredConjunction &p)

Data Fields

- T elements

  STL member.

6.113.1 Detailed Description

Definition at line 138 of file Predicate.hpp.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 PredConjunction() [1/3]

`minerule::PredConjunction::PredConjunction ( ) [inline]`

Definition at line 140 of file Predicate.hpp.

```
00140 {};
```

6.113.2.2 PredConjunction() [2/3]

```
minerule::PredConjunction::PredConjunction (
            const list_AND_node * land )
```

Definition at line 75 of file Predicate.cpp.

```
00075                                                                {
00076     while(land!=NULL) {
00077       push_back(&SimplePredicate::newSimplePredicate(land->sp));
00078       land = land->next;
00079     }
00080   }
```

6.113.2.3 PredConjunction() [3/3]

```
minerule::PredConjunction::PredConjunction (
            const PredConjunction & rhs )
```

Definition at line 82 of file Predicate.cpp.

```
00082                                                                {
00083     *this = rhs;
00084   }
```

6.113.2.4 ~PredConjunction()

```
minerule::PredConjunction::~PredConjunction ( )  [inline]
```

Definition at line 143 of file Predicate.hpp.

```
00143 { }
```

6.113.3 Member Function Documentation

6.113.3.1 evaluate()

```
bool minerule::PredConjunction::evaluate (
            const VarSet & vset ) const
```

Definition at line 88 of file Predicate.cpp.

```
00088                                                          {
00089     for(const_iterator it=begin(); it!=end(); it++) {
00090       if( vset.getVar((*it)->getVarId())==false )
00091         return false;
00092     }
00093
00094     return true;
00095   }
```

6.113.3.2 operator&=()

```
PredConjunction & minerule::PredConjunction::operator&= (
            const PredConjunction & p )
```

Builds the logical and of ∗this and p and return ∗this.

Definition at line 98 of file Predicate.cpp.

```
00098                                                    {
00099     insert( end(), p.begin(), p.end() );
00100     return *this;
00101   }
```

6.113.4 Field Documentation

6.113.4.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

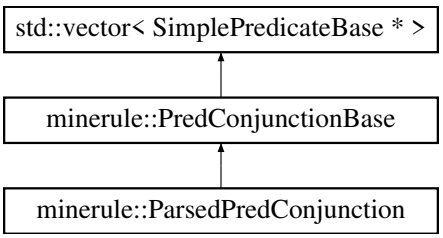
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/Predicate.cpp

6.114 minerule::PredConjunctionBase Class Reference

```
#include <PredicateBase.hpp>
```

Inheritance diagram for minerule::PredConjunctionBase:



Public Member Functions

- PredConjunctionBase ()
- PredConjunctionBase (const PredConjunctionBase &)
- virtual ∼PredConjunctionBase ()
- virtual PredicateBase ∗ newPredicate () const
- virtual PredConjunctionBase ∗ newPredConjunction () const
- virtual PredConjunctionBase ∗ cloneInstance () const
- virtual PredConjunctionBase ∗ newInstance () const
- PredConjunctionBase & operator&= (const PredConjunctionBase &p)
- PredicateBase & operator! () const
- bool find (SimplePredicateBase ∗sp) const
- virtual PredConjunctionBase ∗ copy_and_append (SimplePredicateBase ∗sp)

Data Fields

- T elements

  STL member.

6.114.1 Detailed Description

this class represent a conjunction of simplePredicate

Definition at line 151 of file PredicateBase.hpp.

6.114.2 Constructor & Destructor Documentation

6.114.2.1 PredConjunctionBase() [1/2]

```
minerule::PredConjunctionBase::PredConjunctionBase ( )  [inline]
```

The empty constructor

Definition at line 156 of file PredicateBase.hpp.

```
00156 {};
```

6.114.2.2 PredConjunctionBase() [2/2]

```
minerule::PredConjunctionBase::PredConjunctionBase (
            const PredConjunctionBase & rhs )
```

The copy constructor

Definition at line 62 of file PredicateBase.cpp.

```
00062                                                                    {
00063     PredConjunctionBase::const_iterator it;
00064     for(it=rhs.begin(); it!=rhs.end(); ++it) {
00065       this->push_back( (*it)->cloneInstance() );
00066     }
00067   }
```

6.114.2.3 ~PredConjunctionBase()

```
minerule::PredConjunctionBase::~PredConjunctionBase ( )  [virtual]
```

The distructor of this class

Definition at line 73 of file PredicateBase.cpp.

```
00073                                          {
00074         for (iterator it=begin();it!=end();++it){
00075                   delete *it;
00076         }
00077
00078   }
```

6.114.3 Member Function Documentation

6.114.3.1 cloneInstance()

```
virtual PredConjunctionBase * minerule::PredConjunctionBase::cloneInstance ( ) const  [inline],
[virtual]
```

- this method is very important because permit to clone the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

Returns

PredConjunctioBase a new object that was a copy of this object

Reimplemented in minerule::ParsedPredConjunction.

Definition at line 181 of file PredicateBase.hpp.

```
00181                                                {
00182       return new PredConjunctionBase(*this);
00183     }
```

6.114.3.2 copy_and_append()

```
virtual PredConjunctionBase * minerule::PredConjunctionBase::copy_and_append (
             SimplePredicateBase * sp )  [inline], [virtual]
```

Build a new PredConjunctionBaseParse that contain the old vector plus the new element and not modify the old vector

Parameters

*SimplePredicateBase**	sp the simple term that i want insert in the new PredConjunctionBase

Definition at line 213 of file PredicateBase.hpp.

```
00213                                                                                {
00214       PredConjunctionBase* conj =  this->cloneInstance();
00215       conj->push_back( sp->cloneInstance() );
00216       return conj;
00217     }
```

6.114.3.3 find()

```
bool minerule::PredConjunctionBase::find (
             SimplePredicateBase * sp ) const
```

Find if this PredConjunctionBase contains the SimplePredicateBase *sp .

Definition at line 84 of file PredicateBase.cpp.

```
00084                                                                    {
00085     PredConjunctionBase::const_iterator it;
00086
00087     // Checking if sp is present in this PredConjunctionBase
00088     for (it=this->begin();
00089          it!=this->end() && **it != *sp ;
00090          it++); // note the body of the for is empty
00091
00092     return it!=this->end();
00093   }
```

6.114.3.4 newInstance()

```
virtual PredConjunctionBase * minerule::PredConjunctionBase::newInstance ( ) const  [inline],
[virtual]
```

- this method is very important because permit to create a new instance of the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

  Returns

  PredConjunctioBase a new object of this class

Reimplemented in minerule::ParsedPredConjunction.

Definition at line 186 of file PredicateBase.hpp.

```
00186                                                {
00187       return new PredConjunctionBase();
00188     }
```

6.114.3.5 newPredConjunction()

```
PredConjunctionBase * minerule::PredConjunctionBase::newPredConjunction ( ) const  [virtual]
```

Reimplemented in minerule::ParsedPredConjunction.

Definition at line 22 of file PredicateBase.cpp.

```
00022                                                                    {
00023     return new PredConjunctionBase();
00024   }
```

6.114.3.6 newPredicate()

```
PredicateBase * minerule::PredConjunctionBase::newPredicate ( ) const  [virtual]
```

Reimplemented in minerule::ParsedPredConjunction.

Definition at line 18 of file PredicateBase.cpp.

```
00018                                                {
00019     return new PredicateBase();
00020   }
```

6.114.3.7 operator"!()

```
PredicateBase & minerule::PredConjunctionBase::operator!  ( ) const
```

Builds the negation of this PredConjunctionBase and return a PredicateBase object (in form disjunctive)

Returns

PredicateBase& the negation of this conjunction predicate

Definition at line 109 of file PredicateBase.cpp.

```
00109                                       {
00110     PredicateBase* result = newPredicate();
00111     PredConjunctionBase::const_iterator it;
00112     for(it=this->begin(); it!=this->end(); ++it) {
00113       (*result) |= !(**it);
00114     }
00115
00116     return *result;
00117   }
```

6.114.3.8 operator&=()

```
PredConjunctionBase & minerule::PredConjunctionBase::operator&= (
            const PredConjunctionBase & p )
```

Builds the logical and of ∗this and p and return ∗this.

Definition at line 97 of file PredicateBase.cpp.

```
00097                                                                   {
00098     PredConjunctionBase::const_iterator it;
00099     for(it=rhs.begin(); it!=rhs.end(); ++it) {
00100       if( !this->find(*it) )
00101         this->push_back( (*it)->cloneInstance() );
00102     }
00103
00104     return *this;
00105   }
```

6.114.4 Field Documentation

6.114.4.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

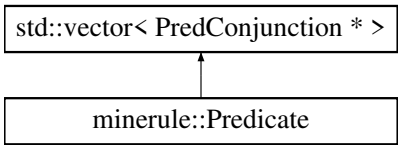
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/PredicateBase.hpp
- /Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp

6.115 minerule::Predicate Class Reference

`#include <Predicate.hpp>`

Inheritance diagram for minerule::Predicate:



Public Member Functions

- Predicate ()
- Predicate (const Predicate &)
- Predicate (const list_OR_node *node)
- ~Predicate ()
- Predicate & operator= (const Predicate &rhs)
- Predicate & operator&= (const Predicate &p)
- Predicate & operator|= (const Predicate &p)
- std::set< SimplePredicate *, PtrSimplePredComp > & getPredicateList (bool rebuildSet=false)
- bool evaluate (const VarSet &) const
- void setNumVariables (size_t n)
- size_t getNumVariables () const

Data Fields

- T elements

  STL member.

6.115.1 Detailed Description

Definition at line 158 of file Predicate.hpp.

6.115.2 Constructor & Destructor Documentation

6.115.2.1 Predicate() [1/3]

`minerule::Predicate::Predicate ( )  [inline]`

Definition at line 164 of file Predicate.hpp.

```
00164                     :
00165       predList(NULL),
00166       numVariables(0)  {};
```

6.115.2.2 Predicate() [2/3]

```
minerule::Predicate::Predicate (
            const Predicate & rhs )
```

Definition at line 116 of file Predicate.cpp.

```
00116                                           :
00117     predList(NULL),
00118     numVariables(0) {
00119     Predicate::const_iterator it;
00120     for(it=rhs.begin(); it!=rhs.end(); it++) {
00121       push_back( new PredConjunction(**it) );
00122     }
00123   }
```

6.115.2.3 Predicate() [3/3]

```
minerule::Predicate::Predicate (
            const list_OR_node * node )
```

Definition at line 107 of file Predicate.cpp.

```
00107                                           :
00108     predList(NULL),
00109     numVariables(0) {
00110     while( lor!=NULL ) {
00111       push_back( new PredConjunction(lor->l_and) );
00112       lor=lor->next;
00113     }
00114   }
```

6.115.2.4 ~Predicate()

```
minerule::Predicate::~Predicate ( )
```

Definition at line 128 of file Predicate.cpp.

```
00128                                {
00129     for( iterator it=begin(); it!=end(); it++ ) {
00130       delete *it;
00131     }
00132   }
```

6.115.3 Member Function Documentation

6.115.3.1 evaluate()

```
bool minerule::Predicate::evaluate (
            const VarSet & vset ) const
```

Definition at line 195 of file Predicate.cpp.

```
00195                                           {
00196     if(empty())
00197       return true;
00198
00199     for( const_iterator it=begin(); it!=end(); it++ ) {
00200       if( (*it)->evaluate(vset)==true )
00201         return true;
00202     }
00203
00204     return false;
00205   }
```

6.115.3.2 getNumVariables()

```
size_t minerule::Predicate::getNumVariables ( ) const  [inline]
```

Definition at line 216 of file Predicate.hpp.

```
00216                                    {
00217       return numVariables;
00218     }
```

6.115.3.3 getPredicateList()

```
std::set< SimplePredicate *, PtrSimplePredComp > & minerule::Predicate::getPredicateList (
             bool rebuildSet = false )
```

Definition at line 170 of file Predicate.cpp.

```
00170                                                {
00171     if(rebuildSet && predList!=NULL) {
00172       delete predList;
00173       predList=NULL;
00174     }
00175
00176     if( predList!=NULL )
00177       return *predList;
00178
00179     predList = new std::set<SimplePredicate*,PtrSimplePredComp>();
00180
00181     iterator it;
00182     for( it=begin(); it!=end(); it++ ) {
00183       PredConjunction::iterator cit;
00184       for( cit=(*it)->begin(); cit!=(*it)->end(); cit++ ) {
00185         predList->insert(*cit);
00186       }
00187     }
00188
00189     return *predList;
00190   }
```

6.115.3.4 operator&=()

```
Predicate & minerule::Predicate::operator&= (
             const Predicate & p )
```

Builds the logical and of *this and p and returns *this

Definition at line 134 of file Predicate.cpp.

```
00134                                                    {
00135     size_t oldSize = this->size();
00136     // making room for the new OR predicates
00137     Predicate thisCopy(*this);
00138     for(size_t i=0; i<p.size()-1; i++) {
00139       *this |= thisCopy;
00140     }
00141
00142     //    std::cout << "This copy:" << thisCopy << std::endl;
00143     //    std::cout << "P:" << p << std::endl;
00144
00145     // building the conjunctions (now there are oldSize*p.size() conj
00146     // in this predicate. The first oldSize ones need to be conjuncted
00147     // with p[0], the second oldSize ones with p[1]... the p.size()'th
00148     // oldSize ones with p[p.size()-1].
00149     for(size_t i=0; i<p.size(); i++) {
00150       for(size_t j=0; j<oldSize; j++) {
00151         assert(j+i*oldSize < this->size() );
00152         //      std::cout << "Pred("<<i<<","<<j<<"):" << *(*this)[j+i*oldSize] << std::endl;
00153         *(*this)[j+i*oldSize] &= *p[i];
00154       }
00155     }
00156
00157     return *this;
00158   }
```

6.115.3.5 operator=()

```
Predicate & minerule::Predicate::operator= (
            const Predicate & rhs )  [inline]
```

Definition at line 173 of file Predicate.hpp.

```
00173                                                {
00174       //freeing used memory
00175       for( iterator it=begin(); it!=end(); it++ ) {
00176         delete *it;
00177       }
00178
00179       // initializing variables
00180       predList=NULL;
00181       numVariables=0;
00182       // copying rhs contents
00183       for(Predicate::const_iterator it=rhs.begin(); it!=rhs.end(); it++) {
00184         push_back( new PredConjunction(**it) );
00185       }
00186
00187       return *this;
00188     }
```

6.115.3.6 operator"|=()

```
Predicate & minerule::Predicate::operator|= (
            const Predicate & p )
```

Builds the logical or of ∗this and p and returns ∗this

Definition at line 160 of file Predicate.cpp.

```
00160                                                    {
00161     for(Predicate::const_iterator it=p.begin(); it!=p.end(); it++) {
00162       push_back( new PredConjunction( **it ) );
00163     }
00164
00165     return *this;
00166   }
```

6.115.3.7 setNumVariables()

```
void minerule::Predicate::setNumVariables (
            size_t n )  [inline]
```

Definition at line 213 of file Predicate.hpp.

```
00213                                    {
00214       numVariables=n;
00215     }
```

6.115.4 Field Documentation

6.115.4.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

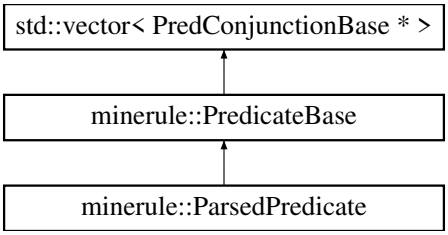
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/Predicate.cpp

6.116 minerule::PredicateBase Class Reference

```
#include <PredicateBase.hpp>
```

Inheritance diagram for minerule::PredicateBase:



Public Member Functions

- PredicateBase ()
- PredicateBase (const PredicateBase &)
- virtual PredicateBase * cloneInstance () const
- virtual PredicateBase * newInstance () const
- virtual ~PredicateBase ()
- virtual PredicateBase * newPredicate () const
- virtual PredConjunctionBase * newPredConjunction () const
- virtual PredicateBase & operator= (const PredicateBase &rhs)
- PredicateBase & operator&= (const PredicateBase &p)
- PredicateBase & operator|= (const PredicateBase &p)
- void merge (PredicateBase *l2)
- PredicateBase & operator! () const
- void stamp ()

Data Fields

- T elements

  STL member.

6.116.1 Detailed Description

This class represent a disjunction of conjunction(PredConjunctionBase) of condition (simplePredicate)

Definition at line 228 of file PredicateBase.hpp.

6.116.2 Constructor & Destructor Documentation

6.116.2.1 PredicateBase() [1/2]

```
minerule::PredicateBase::PredicateBase ( )  [inline]
```

The empty constructor

Definition at line 233 of file PredicateBase.hpp.
```
00233 {};
```

6.116.2.2 PredicateBase() [2/2]

```
minerule::PredicateBase::PredicateBase (
            const PredicateBase & rhs )
```

The copy constructor

Definition at line 138 of file PredicateBase.cpp.
```
00138                                                          {
00139     PredicateBase::const_iterator it;
00140     for(it=rhs.begin(); it!=rhs.end(); it++) {
00141       push_back( (*it)->cloneInstance() );
00142     }
00143   }
```

6.116.2.3 ~PredicateBase()

```
minerule::PredicateBase::~PredicateBase ( )  [virtual]
```

This is the descructor of this class

Definition at line 145 of file PredicateBase.cpp.
```
00145                                {
00146     for( iterator it=begin(); it!=end(); it++ ) {
00147       delete *it;
00148     }
00149   }
```

6.116.3 Member Function Documentation

6.116.3.1 cloneInstance()

```
virtual PredicateBase * minerule::PredicateBase::cloneInstance ( ) const  [inline], [virtual]
```

- this method is very important because permit to clone the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

Returns

PredicateBase a copy of this object

Reimplemented in minerule::ParsedPredicate.

Definition at line 242 of file PredicateBase.hpp.

```
00242                                                {
00243       return new PredicateBase(*this);
00244     }
```

6.116.3.2 merge()

```
void minerule::PredicateBase::merge (
            PredicateBase * l2 )  [inline]
```

merge unordered the two vector into the first and delete all the element of the vector pass as parameter

Definition at line 286 of file PredicateBase.hpp.

```
00286                                        {
00287       copy( l2->begin(), l2->end(), std::back_insert_iterator<PredicateBase>(*this));
00288       erase(l2->begin(), l2->end());
00289
00290       /*    Predicate::iterator it_and;
00291             it_and=l2->begin();
00292             while (it_and!=l2->end()){
00293             this->push_back(*it_and);
00294             *it_and=NULL;
00295             it_and++;
00296             }*/
00297     }
```

6.116.3.3 newInstance()

```
virtual PredicateBase * minerule::PredicateBase::newInstance ( ) const  [inline], [virtual]
```

- this method is very important because permit to create a new instance of the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

Returns

PredidcateBase a new instance of this class

Reimplemented in minerule::ParsedPredicate.

Definition at line 248 of file PredicateBase.hpp.

```
00248                                                {
00249       return new PredicateBase();
00250     }
```

6.116.3.4 newPredConjunction()

```
PredConjunctionBase * minerule::PredicateBase::newPredConjunction ( ) const  [virtual]
```

Reimplemented in minerule::ParsedPredicate.

Definition at line 30 of file PredicateBase.cpp.

```
00030                                                    {
00031     return new PredConjunctionBase();
00032   }
```

6.116.3.5 newPredicate()

```
PredicateBase * minerule::PredicateBase::newPredicate ( ) const  [virtual]
```

Reimplemented in minerule::ParsedPredicate.

Definition at line 26 of file PredicateBase.cpp.

```
00026                                                {
00027     return new PredicateBase();
00028   }
```

6.116.3.6 operator"!()

```
PredicateBase & minerule::PredicateBase::operator!  ( ) const
```

- Builds the logical negation but i return another object of this class (in disjunction form)

Definition at line 185 of file PredicateBase.cpp.

```
00185                                  {
00186     assert( this->size() != 0 );
00187
00188     PredicateBase* result = newPredicate();
00189     PredicateBase::const_iterator it=this->begin();
00190
00191     (*result) |= !(**it);
00192     ++it;
00193
00194     for(; it!=this->end(); ++it ) {
00195       (*result) &= !(**it);
00196     }
00197
00198     return *result;
00199   }
```

6.116.3.7 operator&=()

```
PredicateBase & minerule::PredicateBase::operator&= (
            const PredicateBase & p )
```

Builds the logical and of ∗this and p and returns ∗this

Definition at line 151 of file PredicateBase.cpp.

```
00151                                                                {
00152     size_t oldSize = this->size();
00153     // making room for the new OR predicates
00154     PredicateBase* thisCopy = this->cloneInstance();
00155     for(size_t i=0; i<p.size()-1; i++) {
00156       *this |= *thisCopy;
00157     }
00158
00159     // building the conjunctions (now there are oldSize*p.size() conj
00160     // in this predicate. The first oldSize ones need to be conjuncted
00161     // with p[0], the second oldSize ones with p[1]... the p.size()'th
00162     // oldSize ones with p[p.size()-1].
00163     for(size_t i=0; i<p.size(); i++) {
00164       for(size_t j=0; j<oldSize; j++) {
00165         assert(j+i*oldSize < this->size() );
00166         //      std::cout << "Pred("<<i<<","<<j<<"):" << *(*this)[j+i*oldSize] << std::endl;
00167         *(*this)[j+i*oldSize] &= *p[i];
00168       }
00169     }
00170
00171     return *this;
00172   }
```

6.116.3.8 operator=()

```
PredicateBase & minerule::PredicateBase::operator= (
            const PredicateBase & rhs )  [virtual]
```

Redefine the assigment operation

Definition at line 124 of file PredicateBase.cpp.

```
00124                                                                {
00125     //freeing used memory
00126     for( iterator it=begin(); it!=end(); it++ ) {
00127       delete *it;
00128     }
00129
00130     // copying rhs contents
00131     for(PredicateBase::const_iterator it=rhs.begin(); it!=rhs.end(); it++) {
00132       push_back( (*it)->cloneInstance() );
00133     }
00134
00135     return *this;
00136   }
```

6.116.3.9 operator"|=()

```
PredicateBase & minerule::PredicateBase::operator|= (
            const PredicateBase & p )
```

Builds the logical or of ∗this and p and returns ∗this

Definition at line 175 of file PredicateBase.cpp.

```
00175                                                                {
00176     for(PredicateBase::const_iterator it=p.begin(); it!=p.end(); it++) {
00177       push_back( (*it)->cloneInstance() );
00178     }
00179
00180     return *this;
00181   }
```

6.116.3.10 stamp()

```
void minerule::PredicateBase::stamp ( )  [inline]
```

This is a simple function that print a PredicateBase object

Definition at line 305 of file PredicateBase.hpp.

```
00305                 {
00306       if (this!=NULL){
00307         int c_and=0;
00308         int c_or=0;
00309         PredicateBase::iterator it_and = this->begin();
00310         PredConjunctionBase::iterator it_sp;
00311         while (it_and!=this->end()){
00312           if (c_or) std::cout«"\n OR \n";
00313           it_sp=(*it_and)->begin();
00314           c_and=0;
00315           while(it_sp!=(*it_and)->end()){
00316             if (c_and) std::cout«" AND ";
00317             std::cout«(*it_sp)->getVal1()«" ";
00318             std::cout«(*it_sp)->getOp()«" ";
00319             std::cout«(*it_sp)->getVal2();
00320             it_sp++;
00321             c_and++;
00322           }
00323           //delete *it_and;
00324           it_and++;
00325           c_or++;
00326         }
00327         std::cout«"\n\n\n\n";
00328       }
00329
00330     }
```

6.116.4 Field Documentation

6.116.4.1 elements

```
T std::vector< T >::elements  [inherited]
```

STL member.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/PredicateBase.hpp
- /Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp

6.117 minerule::PredicateUtils Class Reference

```
#include <PredicateUtils.hpp>
```

Public Types

- typedef EncodedNF::CodesRelationship PredicateRelationship

Static Public Member Functions

- static bool predicatesAreEquivalent (Predicate &p1, Predicate &p2, const std::string &tab_source)
- static PredicateRelationship getPredicateRelationship (Predicate &p1, Predicate &p2, const std::string &tab↩ _source)

6.117.1 Detailed Description

Definition at line 93 of file PredicateUtils.hpp.

6.117.2 Member Typedef Documentation

6.117.2.1 PredicateRelationship

```
typedef EncodedNF::CodesRelationship minerule::PredicateUtils::PredicateRelationship
```

Definition at line 95 of file PredicateUtils.hpp.

6.117.3 Member Function Documentation

6.117.3.1 getPredicateRelationship()

```
PredicateUtils::PredicateRelationship minerule::PredicateUtils::getPredicateRelationship (
            Predicate & p1,
            Predicate & p2,
            const std::string & tab_source )  [static]
```

Definition at line 75 of file PredicateUtils.cpp.

```
00077                                                                   {
00078     // The following two lines assign unique variables id
00079     // to elements in p1,p2. It do so setting the variable id
00080     // for each element of the union (through ci) to the position
00081     // in which the predicate appear in the union itself. Note
00082     // that this work only because two equivalent predicates are
00083     // indeed two references to the same object (look at SimplePredicate
00084     // class to understand why this is true).
00085     size_t counter=0;
00086     list_AND_node* allPreds=NULL;
00087
00088     CountingIterator ci(counter,allPreds);
00089     std::set<SimplePredicate*, PtrSimplePredComp>& sp1 =
00090       p1.getPredicateList();
00091     std::set<SimplePredicate*, PtrSimplePredComp>& sp2 =
00092       p2.getPredicateList();
00093
00094     std::set_union( sp1.begin(), sp1.end(),
00095                     sp2.begin(), sp2.end(),
00096                     ci );
00097
00098
00099     p1.setNumVariables(counter);
00100     p2.setNumVariables(counter);
00101
00102     InvalidConfigurationFilter filter( tab_source, allPreds );
00103
00104     ExpressionNFCoder e1(filter);
00105     ExpressionNFCoder e2(filter);
00106
00107     PredicateUtils::PredicateRelationship rel=
00108       EncodedNF::getCodesRelationship(e1.encode(p1),e2.encode(p2));
00109
00110     CountingIterator::delete_list_AND_node(allPreds);
00111     return rel;
00112   }
```

6.117.3.2 predicatesAreEquivalent()

```
bool minerule::PredicateUtils::predicatesAreEquivalent (
            Predicate & p1,
            Predicate & p2,
            const std::string & tab_source )  [static]
```

Definition at line 125 of file PredicateUtils.cpp.

```
00127                                                                    {
00128     // The following two lines assign unique variables id
00129     // to elements in p1,p2. It do so setting the variable id
00130     // for each element of the union (through ci) to the position
00131     // in which the predicate appear in the union itself. Note
00132     // that this work only because two equivalent predicates are
00133     // indeed to references to the same object (look at SimplePredicate
00134     // class to understand why this is true).
00135     size_t counter=0;
00136     list_AND_node* allPreds=NULL;
00137
00138     CountingIterator ci(counter, allPreds);
00139     std::set<SimplePredicate*, PtrSimplePredComp>& sp1 =
00140       p1.getPredicateList();
00141     std::set<SimplePredicate*, PtrSimplePredComp>& sp2 =
00142       p2.getPredicateList();
00143
00144     std::set_union( sp1.begin(), sp1.end(),
00145                     sp2.begin(), sp2.end(),
00146                     ci );
00147
00148
00149     p1.setNumVariables(counter);
00150     p2.setNumVariables(counter);
00151
00152     InvalidConfigurationFilter filter( tab_source, allPreds );
00153
00154     ExpressionNFCoder e1(filter);
00155     ExpressionNFCoder e2(filter);
00156
00157     return e1.encode(p1)==e2.encode(p2);
00158   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/PredicateUtils.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/PredicateUtils.cpp

6.118 minerule::PrepareDataUtils Class Reference

`#include <PrepareDataUtils.hpp>`

Public Member Functions

- PrepareDataUtils (const ParsedMinerule &m, const SourceTableRequirements &requirements)
- std::string buildBodyTableQuery (SourceRowColumnIds &rowDes, const std::string &condition) const
- std::string buildHeadTableQuery (SourceRowColumnIds &rowDes, const std::string &condition) const
- std::string buildExtendedSourceTableQuery (SourceRowColumnIds &rowDes) const
- std::string buildSourceTableQuery (SourceRowColumnIds &rowDes) const
- size_t evaluateTotGroups () const

Static Public Member Functions

- static std::string buildAttrListDescription (const ParsedMinerule::AttrVector &attrs, const std::string &alias="", bool addColAlias=false)
- static void dropTableIfExists (mrdb::Connection *conn, const std::string &tname)
- static size_t evaluateTotGroups (const ParsedMinerule &pmr)

6.118.1 Detailed Description

Definition at line 28 of file PrepareDataUtils.hpp.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 PrepareDataUtils()

```
minerule::PrepareDataUtils::PrepareDataUtils (
            const ParsedMinerule & m,
            const SourceTableRequirements & requirements )  [inline]
```

Definition at line 47 of file PrepareDataUtils.hpp.

```
00049       : mr(m), sourceTableRequirements(requirements) {}
```

6.118.3 Member Function Documentation

6.118.3.1 buildAttrListDescription()

```
std::string minerule::PrepareDataUtils::buildAttrListDescription (
            const ParsedMinerule::AttrVector & attrs,
            const std::string & alias = "",
            bool addColAlias = false )  [static]
```

Definition at line 26 of file PrepareDataUtils.cpp.

```
00026                                                                   {
00027
00028                 #define ADDCOLALIAS (addColAlias?(" AS " + alias+*it):"")
00029
00030                 ParsedMinerule::AttrVector::const_iterator it;
00031                 std::string result;
00032                 it=attrs.begin();
00033
00034     // the first element should not have a leading ","
00035                 assert(it!=attrs.end());
00036                 if(alias.length()!=0)
00037                         result=alias+"."+*it + ADDCOLALIAS;
00038                 else
00039                         result=*it;
00040
00041                 it++;
00042
00043                 for( ;it!=attrs.end(); it++ ) {
00044                         if(alias.length()!=0)
00045                                 result += ","+alias+"."+*it + ADDCOLALIAS;
00046                         else
00047                                 result+=","+*it;
00048                 }
00049
00050                 #undef ADDCOLALIAS
00051
00052                 return result;
00053         }
```

6.118.3.2 buildBodyTableQuery()

```
std::string minerule::PrepareDataUtils::buildBodyTableQuery (
            SourceRowColumnIds & rowDes,
            const std::string & condition ) const
```

Definition at line 141 of file PrepareDataUtils.cpp.

```
00141
                {
00142                std::string queryText;
00143
00144                queryText = "SELECT ";
00145                queryText += buildAttrListDescription(mr.ga);
00146                queryText += "," + buildAttrListDescription(mr.ba);
00147                queryText += " FROM "+mr.tab_source;
00148                if(!body_mining_condition.empty())
00149                        queryText += " WHERE "+body_mining_condition;
00150
00151                if( sourceTableRequirements.sortedGids() )
00152                        queryText += " ORDER BY "+buildAttrListDescription(mr.ga);
00153
00154                unsigned int lastElem;
00155                lastElem= rowDes.setGroupElems(1,mr.ga.size());
00156                rowDes.setBodyElems(lastElem+1,mr.ba.size());
00157
00158                return queryText;
00159        }
```

6.118.3.3 buildExtendedSourceTableQuery()

```
std::string minerule::PrepareDataUtils::buildExtendedSourceTableQuery (
            SourceRowColumnIds & rowDes ) const
```

Definition at line 289 of file PrepareDataUtils.cpp.

```
00289                                                                                                {
00290                std::string tableName = createSourceTable();
00291
00292                std::string queryText = "SELECT * FROM "+tableName;
00293
00294                unsigned int lastElem;
00295                lastElem= rowDes.setGroupElems(1,mr.ga.size());
00296                lastElem= rowDes.setClusterBodyElems(lastElem+1,mr.ca.size());
00297                lastElem= rowDes.setBodyElems(lastElem+1,mr.ba.size());
00298                lastElem= rowDes.setClusterHeadElems(lastElem+1,mr.ca.size());
00299                rowDes.setHeadElems(lastElem+1, mr.ha.size());
00300
00301                return queryText;
00302        }
```

6.118.3.4 buildHeadTableQuery()

```
std::string minerule::PrepareDataUtils::buildHeadTableQuery (
            SourceRowColumnIds & rowDes,
            const std::string & condition ) const
```

Definition at line 162 of file PrepareDataUtils.cpp.

```
00162
                {
00163                std::string queryText;
00164
00165                queryText = "SELECT ";
00166                queryText += buildAttrListDescription(mr.ga);
00167                queryText += "," + buildAttrListDescription(mr.ha);
00168                queryText += " FROM "+mr.tab_source;
00169
```

```
00170                 if(!head_mining_condition.empty())
00171                         queryText += " WHERE "+head_mining_condition;
00172 
00173                 if( sourceTableRequirements.sortedGids() )
00174                         queryText += " ORDER BY "+buildAttrListDescription(mr.ga);
00175 
00176                 unsigned int lastElem;
00177                 lastElem= rowDes.setGroupElems(1,mr.ga.size());
00178                 rowDes.setHeadElems(lastElem+1,mr.ha.size());
00179 
00180                 return queryText;
00181         }
```

6.118.3.5 buildSourceTableQuery()

```
std::string minerule::PrepareDataUtils::buildSourceTableQuery (
             SourceRowColumnIds & rowDes ) const
```

Definition at line 305 of file PrepareDataUtils.cpp.

```
00305                                                                                                 {
00306                 if(sourceTableRequirements.crossProduct() ) {
00307                         return buildExtendedSourceTableQuery(rowDes);
00308                 } else {
00309                         throw MineruleException(MR_ERROR_INTERNAL, "Check for consistency with
      previous implementation");
00310                         return buildBodyTableQuery(rowDes,"");
00311                 }
00312         }
```

6.118.3.6 dropTableIfExists()

```
void minerule::PrepareDataUtils::dropTableIfExists (
             mrdb::Connection * conn,
             const std::string & tname )  [static]
```

Definition at line 221 of file PrepareDataUtils.cpp.

```
00221                                                                                                 {
00222                 std::auto_ptr<mrdb::Statement> state(conn->createStatement());
00223                 state->execute("DROP TABLE IF EXISTS "+tname);
00224         }
```

6.118.3.7 evaluateTotGroups() [1/2]

```
size_t minerule::PrepareDataUtils::evaluateTotGroups ( ) const  [inline]
```

Definition at line 80 of file PrepareDataUtils.hpp.

```
00081       {
00082     return evaluateTotGroups(mr);
00083   }
```

6.118.3.8 evaluateTotGroups() [2/2]

```
size_t minerule::PrepareDataUtils::evaluateTotGroups (
            const ParsedMinerule & pmr )  [static]
```

Query the DBMS in order to get the total number of groups contained in the source table (BEFORE filtering it). It returns that value if everything is ok, it throws a MineruleException(MR_DATABASEERROR) if it cannot access to that number after the query. It may also throw an SQLException in case something goes wrong during the query evaluation.

Definition at line 315 of file PrepareDataUtils.cpp.

```
00315                                                                    {
00316                 MRLogPush("Evaluating Number of groups...");
00317                 std::string qry=
00318                         "SELECT count(distinct " + buildAttrListDescription(pmr.ga) +") "
00319                                 "FROM " + pmr.tab_source;
00320
00321                 mrdb::Connection* conn =
00322                         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00323
00324                 std::auto_ptr<mrdb::Statement> state(conn->createStatement());
00325                 std::auto_ptr<mrdb::ResultSet> rs(state->executeQuery(qry));
00326
00327                 MRLogPop();
00328
00329                 if(!rs->next())
00330                         throw MineruleException(MR_ERROR_DATABASE_ERROR, "Cannot count the number of
      groups in the source table");
00331
00332                 return rs->getInt(1);
00333         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/PrepareDataUtils.hpp
- /Users/esposito/Software/minerule/src/Database/PrepareDataUtils.cpp

6.119 mrdb::PreparedStatement Class Reference

`#include <PreparedStatement.hpp>`

Public Member Functions

- virtual ~PreparedStatement ()
- virtual bool execute ()=0
- virtual ResultSet * executeQuery ()=0
- virtual void setDouble (int idx, double val)=0
- virtual void setInt (int idx, int val)=0
- virtual void setString (int idx, const std::string &val)=0
- virtual void setLong (int idx, long val)=0

6.119.1 Detailed Description

Definition at line 10 of file PreparedStatement.hpp.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 ~PreparedStatement()

```
virtual mrdb::PreparedStatement::~PreparedStatement ( )  [inline], [virtual]
```

Definition at line 12 of file PreparedStatement.hpp.

```
00012 {}
```

6.119.3 Member Function Documentation

6.119.3.1 execute()

```
virtual bool mrdb::PreparedStatement::execute ( )  [pure virtual]
```

Execute this statement (assumes that it does not returns any tuple)

Returns

true if a result set was available, false otherwise

Exceptions

a	MineruleException (with error code MR_ERROR_DATABASE_ERROR) if anything goes wrong.

6.119.3.2 executeQuery()

```
virtual ResultSet * mrdb::PreparedStatement::executeQuery ( )  [pure virtual]
```

Execute this statement. THREAD SAFETY: It returns a ResultSet that should be consumed by the same thread that owns this object.

Returns

a ResultSet

Exceptions

a	MineruleException (with error code MR_ERROR_DATABASE_ERROR) if anything goes wrong. @memory the caller should dealloc the result set once it is no longer needed

6.119.3.3 setDouble()

```
virtual void mrdb::PreparedStatement::setDouble (
            int idx,
            double val )  [pure virtual]
```

Sets a parameter value to a Double

Parameters

idx	the index of the parameter to be set
val	the value to set

6.119.3.4 setInt()

```
virtual void mrdb::PreparedStatement::setInt (
            int idx,
            int val )  [pure virtual]
```

Sets a parameter value to an int

Parameters

idx	the index of the parameter to be set
val	the value to set

6.119.3.5 setLong()

```
virtual void mrdb::PreparedStatement::setLong (
            int idx,
            long val )  [pure virtual]
```

Sets a parameter value to a Long

Parameters

idx	the index of the parameter to be set
val	the value to set

6.119.3.6 setString()

```
virtual void mrdb::PreparedStatement::setString (
            int idx,
            const std::string & val )  [pure virtual]
```

Sets a parameter value to a string

Parameters

idx	the index of the parameter to be set
val	the value to set

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/PreparedStatement.hpp

6.120 mrc::Printer Class Reference

```
#include <Printer.hpp>
```

Public Member Functions

- Printer (std::ostream &out, const Options &options)
- void print (const minerule::CatalogueInfo &info)
- void print (const std::vector< minerule::CatalogueInfo > &list)

6.120.1 Detailed Description

Definition at line 24 of file Printer.hpp.

6.120.2 Constructor & Destructor Documentation

6.120.2.1 Printer()

```
mrc::Printer::Printer (
            std::ostream & out,
            const Options & options )  [inline]
```

Definition at line 30 of file Printer.hpp.

```
00030 : _out(out), _options(options), _result_index(1) {}
```

6.120.3 Member Function Documentation

6.120.3.1 print() [1/2]

```
void mrc::Printer::print (
            const minerule::CatalogueInfo & info )
```

Definition at line 52 of file Printer.cpp.

```
00052                                                           {
00053                 format("name:", info.qryName);
00054
00055                 if(_options.getListFormat().size) {
00056                         format("size:", info.resSize);
00057                 }
00058
00059                 if(_options.getListFormat().result) {
00060                         format("tables:\n", minerule::StringUtils::join(info.resTables, ", "));
00061                 }
00062
00063                 if(_options.getListFormat().text) {
00064                         format("text:\n", info.qryText);
00065                 }
00066         }
```

6.120.3.2 print() [2/2]

```
void mrc::Printer::print (
            const std::vector< minerule::CatalogueInfo > & list )
```

Definition at line 68 of file Printer.cpp.

```
00068                                                                       {
00069                 _out « "Found " « list.size() « " result sets:" « std::endl;
00070                 _result_index = 1;
00071                 std::vector<minerule::CatalogueInfo>::const_iterator it;
00072                 for(it=list.begin(); it!=list.end(); it++) {
00073                         print(*it);
00074                         _result_index++;
00075                 }
00076         }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.hpp
- /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.cpp

6.121 minerule::Progress Class Reference

```
#include <Progress.hpp>
```

Public Types

- typedef void(* UpdateHandler) (int n, int count, int num)

Public Member Functions

- Progress (int num)
- ~Progress ()
- void start ()
- void end ()
- void tick (int n=1)
- void setHandler (UpdateHandler handler)

Static Public Member Functions

- static bool handleStartStop (std::ostream &, int n)
- static UpdateHandler setDefaultHandler (UpdateHandler handler)

6.121.1 Detailed Description

Definition at line 9 of file Progress.hpp.

6.121.2 Member Typedef Documentation

6.121.2.1 UpdateHandler

```
typedef void(* minerule::Progress::UpdateHandler) (int n, int count, int num)
```

Definition at line 11 of file Progress.hpp.

6.121.3 Constructor & Destructor Documentation

6.121.3.1 Progress()

```
minerule::Progress::Progress (
            int num )  [inline]
```

Definition at line 21 of file Progress.hpp.

```
00021 : count_(0), num_(100), handler_(defaultHandler_) {}
```

6.121.3.2 ~Progress()

```
minerule::Progress::~Progress ( )  [inline]
```

Definition at line 22 of file Progress.hpp.

```
00022 {}
```

6.121.4 Member Function Documentation

6.121.4.1 end()

```
void minerule::Progress::end ( )  [inline]
```

Definition at line 29 of file Progress.hpp.

```
00029                {
00030     handler_(-2, 0, 0);
00031   }
```

6.121.4.2 handleStartStop()

```
bool minerule::Progress::handleStartStop (
             std::ostream & logger,
             int n )  [static]
```

Definition at line 33 of file Progress.cpp.

```
00033                                                    {
00034         install_abort_handler();
00035
00036         if(n == -1) {
00037                 MRLog() « "  ";
00038                 logger « "\e[?25l";
00039                 return true;
00040         }
00041
00042         if(n == -2) {
00043                 logger « std::endl;
00044                 logger « "\e[?25h";
00045                 return true;
00046         }
00047
00048         return false;
00049 }
```

6.121.4.3 setDefaultHandler()

```
static UpdateHandler minerule::Progress::setDefaultHandler (
             UpdateHandler handler )  [inline], [static]
```

Definition at line 42 of file Progress.hpp.

```
00042                                                          {
00043     UpdateHandler tmp = defaultHandler_;
00044     defaultHandler_ = handler;
00045     return tmp;
00046   }
```

6.121.4.4 setHandler()

```
void minerule::Progress::setHandler (
            UpdateHandler handler )  [inline]
```

Definition at line 48 of file Progress.hpp.

```
00048                                                {
00049                 handler_ = handler;
00050         }
```

6.121.4.5 start()

```
void minerule::Progress::start ( )  [inline]
```

Definition at line 24 of file Progress.hpp.

```
00024                 {
00025         count_=0;
00026         handler_(-1, 0, 0);
00027   }
```

6.121.4.6 tick()

```
void minerule::Progress::tick (
             int n = 1 )  [inline]
```

Definition at line 34 of file Progress.hpp.

```
00034                       {
00035     handler_(n, count_, num_);
00036     count_+=n;;
00037   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Progress.hpp
- /Users/esposito/Software/minerule/src/Utils/Progress.cpp

6.122 minerule::PtrSimplePredComp Class Reference

`#include <Predicate.hpp>`

Public Member Functions

- bool operator() (const SimplePredicate *lhs, const SimplePredicate *rhs) const

6.122.1 Detailed Description

SimplePredicate ptrs comparison function. It is needed because of the use of std::set<SimplePredicates*> objects. We need to order the objects in the set using the < operator defined on SimplePredicates, not using the < operator defined on pointers (which is the default used by class set.

Definition at line 126 of file Predicate.hpp.

6.122.2 Member Function Documentation

6.122.2.1 operator()()

```
bool minerule::PtrSimplePredComp::operator() (
            const SimplePredicate * lhs,
            const SimplePredicate * rhs ) const  [inline]
```

Definition at line 128 of file Predicate.hpp.

```
00129                                                 {
00130       return *lhs < *rhs;
00131     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp

6.123 minerule::QueryNormalizer Class Reference

`#include <QueryNormalizer.hpp>`

Data Structures

- class SubstEntryBody
- class SubstEntryHead

Public Types

- typedef std::set< SubstEntryHead > HeadInfo
- typedef std::set< SubstEntryBody > BodyInfo

Public Member Functions

- QueryNormalizer (const OptimizerCatalogue &oc, ParsedMinerule &mrule)
- void setMinerule (ParsedMinerule &mrule)
- void normalize ()

6.123.1 Detailed Description

Definition at line 33 of file QueryNormalizer.hpp.

6.123.2 Member Typedef Documentation

6.123.2.1 BodyInfo

```
typedef std::set<SubstEntryBody> minerule::QueryNormalizer::BodyInfo
```

Definition at line 97 of file QueryNormalizer.hpp.

6.123.2.2 HeadInfo

```
typedef std::set<SubstEntryHead> minerule::QueryNormalizer::HeadInfo
```

Definition at line 96 of file QueryNormalizer.hpp.

6.123.3 Constructor & Destructor Documentation

6.123.3.1 QueryNormalizer()

```
minerule::QueryNormalizer::QueryNormalizer (
            const OptimizerCatalogue & oc,
            ParsedMinerule & mrule )  [inline]
```

Definition at line 178 of file QueryNormalizer.hpp.

```
00179                                           : optimizerCatalogue(oc),
00180                                             catalogue( oc.getCatalogue() ),
00181                                             mr(&mrule) {
00182     };
```

6.123.4 Member Function Documentation

6.123.4.1 normalize()

```
void minerule::QueryNormalizer::normalize ( )
```

Definition at line 812 of file QueryNormalizer.cpp.

```
00812                                       {
00813         OptimizerCatalogue::Catalogue::const_iterator it = catalogue.find( mr->tab_source );
00814         if( it==catalogue.end() )
00815             return;  // no mapping info for this table
00816
00817         substituteInAttrList( mr->ga, it->second );
00818         substituteInAttrList( mr->ca, it->second );
00819         substituteInAttrList( mr->ra, it->second );
00820         substituteInAttrList( mr->ba, it->second );
00821         substituteInAttrList( mr->ha, it->second );
00822
00823         substituteInPredicate( mr->mc, it->second );
00824         substituteInPredicate( mr->gc, it->second );
00825         substituteInPredicate( mr->cc, it->second );
00826         relaxOperators(mr->mc);
00827         relaxOperators(mr->gc);
00828         relaxOperators(mr->cc);
00829
00830         // We may have removed a number of l_and lists
00831         // an l_or node having an empty l_and list is an
00832         // invalid node, we need to clear them!
00833         cleanPredicate(mr->mc);
00834         cleanPredicate(mr->gc);
00835         cleanPredicate(mr->cc);
00836     }
```

6.123.4.2 setMinerule()

```
void minerule::QueryNormalizer::setMinerule (
            ParsedMinerule & mrule )  [inline]
```

Definition at line 186 of file QueryNormalizer.hpp.

```
00186                                           {
00187       mr = &mrule;
00188     }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp
- /Users/esposito/Software/minerule/src/Optimizer/QueryNormalizer.cpp

6.124 minerule::QueryResult Class Reference

```
#include <QueryResult-header.hpp>
```

Data Structures

- class FastSorter
- class Iterator
- class ResultSet
- class SortBodyHead
- class SortBodyHeadSuppConf
- class SortConfBodyHeadSupp
- class SortConfBodySuppHead
- class SortConfSuppBodyHead
- class SortHeadBodySuppConf
- class SortSuppConfBodyHead

6.124.1 Detailed Description

Definition at line 27 of file QueryResult-header.hpp.

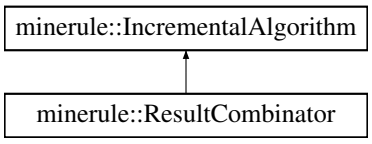
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.125 minerule::ResultCombinator Class Reference

```
#include <ResultCombinator.hpp>
```

Inheritance diagram for minerule::ResultCombinator:



Public Member Functions

- ResultCombinator (const OptimizedMinerule &mr)
- virtual ~ResultCombinator ()
- virtual void execute ()

Static Public Member Functions

- static IncrementalAlgorithm * newIncrementalAlgorithm (const OptimizedMinerule &mr)

Protected Attributes

- const OptimizedMinerule * minerule

6.125.1 Detailed Description

Definition at line 23 of file ResultCombinator.hpp.

6.125.2 Constructor & Destructor Documentation

6.125.2.1 ResultCombinator()

```
minerule::ResultCombinator::ResultCombinator (
            const OptimizedMinerule & mr )  [inline]
```

Definition at line 25 of file ResultCombinator.hpp.

```
00025 : IncrementalAlgorithm(mr) {}
```

6.125.2.2 ~ResultCombinator()

```
virtual minerule::ResultCombinator::~ResultCombinator ( )  [inline], [virtual]
```

Definition at line 26 of file ResultCombinator.hpp.

```
00026 {}
```

6.125.3 Member Function Documentation

6.125.3.1 execute()

```
void minerule::ResultCombinator::execute ( )  [virtual]
```

Implements minerule::IncrementalAlgorithm.

Definition at line 22 of file ResultCombinator.cpp.

```
00022                                   {
00023                 MRLogPusher log1("ResultCombinator incremental algorithm starting...");
00024 
00025                 if( minerule==NULL )
00026                         throw MineruleException(MR_ERROR_INTERNAL,
00027                                 "ResultCombinator::execute called, but the "
00028                                 "optimized minerule is NULL");
00029 
00030                 const std::vector<ParsedMinerule>& minerules=
00031                         minerule->getOptimizationInfo().minerulesToCombine;
00032                 const GAQueryCombinator::QueryOrList& formula =
00033                         minerule->getOptimizationInfo().combinationFormula;
00034 
00035                 QueryResult::ResultSet<QueryResult::SortBodyHead> result;
00036 
00037                 GAQueryCombinator::QueryOrList::const_iterator OrIt;
00038                 for( OrIt=formula.begin();
00039                 OrIt!=formula.end();
00040                 OrIt++) {
00041                         MRLogPusher log2("Analyzing disjunct");
00042                         if(OrIt!=formula.begin()) {
00043                                 MRLog()<<"Analyzing next conjunct"<<std::endl;
00044                         }
00045 
00046                         GAQueryCombinator::QueryAndList::const_iterator AndIt =
00047                                 OrIt->begin();
00048 
00049                         QueryResult::ResultSet<QueryResult::SortBodyHead> curConj;
00050 
00051                         MRLogPush("Loading first query from disk");
00052                         if( AndIt!=OrIt->end() ) {
00053                                 curConj.load( minerules[*AndIt].tab_result );
00054                                 AndIt++;
00055                         }
00056                         MRLogPop();
00057 
00058                         MRLog() << "Current conjunct current result set size:"
00059                                 << curConj.size() << std::endl;
00060 
00061                         MRLog() << "Intersecting with other queries:" << std::endl;
00062                         for( ;  AndIt!=OrIt->end();  AndIt++) {
00063                                 MRLogPusher log3("Analyzing conjunct");
00064 
00065                                 MRLogPush("Loading query from disk.");
00066                                 QueryResult::ResultSet<QueryResult::SortBodyHead>  q;
00067                                 q.load( minerules[*AndIt].tab_result );
00068                                 MRLogPop();
00069 
00070                                 MRLogPush("Intersecting...");
00071                                 curConj.inplace_intersect(q);
00072                                 MRLogPop();
00073 
00074                                 MRLog() << "Current conjunct current result set size:"
00075                                         << curConj.size() << std::endl;
00076                         }
00077 
00078                         MRLogPush("Merging...");
00079                         result.inplace_union(curConj);
00080                         MRLogPop();
00081 
00082                         MRLog() << "Current result set size:"  << result.size() << std::endl;
00083                 }
00084 
00085                 result.save(minerule->getParsedMinerule());
00086         }
```

6.125.3.2 newIncrementalAlgorithm()

```
IncrementalAlgorithm * minerule::IncrementalAlgorithm::newIncrementalAlgorithm (
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.cpp.

```
00025                                                                                {
00026     // if mr has only ItemDependent constraints
00027                 MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028 
00029                 if( mr.hasIDConstraints() ) {
00030                         MRLog() « "The minerule is item dependent!" « std::endl;
00031                         if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                 return new ResultCombinator(mr);
00033                         else
00034                                 return new IDIncrementalAlgorithm(mr);
00035                 }
00036 
00037                 MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038 
00039                 if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040 
00041                         IncrementalAlgorithm* incrAlgo = NULL;
00042 
00043                         switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                 case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                         incrAlgo = new ConstrTree(mr);
00046                                         break;
00047                                 case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                         incrAlgo = new DestrTree(mr);
00049                                         break;
00050                                 case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                         incrAlgo = new ConstrTree(mr);
00052                                         break;
00053                         }
00054 
00055                         return incrAlgo;
00056                 } else {
00057                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                 « " to the system yet." « std::endl;
00059                         return NULL;
00060                 }
00061         }
```

6.125.4 Field Documentation

6.125.4.1 minerule

```
const OptimizedMinerule* minerule::IncrementalAlgorithm::minerule  [protected], [inherited]
```

Definition at line 25 of file IncrementalAlgorithm.hpp.

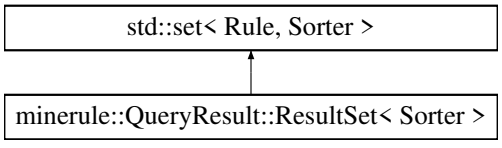
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/ResultCombinator.hpp
- /Users/esposito/Software/minerule/src/Algorithms/ResultCombinator.cpp

6.126 minerule::QueryResult::ResultSet< Sorter > Class Template Reference

```
#include <QueryResult-header.hpp>
```

Inheritance diagram for minerule::QueryResult::ResultSet< Sorter >:



Public Member Functions

- ResultSet ()
- ResultSet (const ResultSet< Sorter > &rhs)
- void load (const std::string &qryName, double sup=-1, double con=-1)
- void save (const ParsedMinerule &mr) const
- ResultSet & inplace_intersect (const ResultSet &)
- ResultSet & inplace_union (const ResultSet &)
- ResultSet & operator&= (const ResultSet &)
- ResultSet & operator|= (const ResultSet &)

Data Fields

- K keys

  STL member.

6.126.1 Detailed Description

template<class Sorter>
class minerule::QueryResult::ResultSet< Sorter >

Definition at line 309 of file QueryResult-header.hpp.

6.126.2 Constructor & Destructor Documentation

6.126.2.1 ResultSet() [1/2]

```
template<class Sorter >
minerule::QueryResult::ResultSet< Sorter >::ResultSet ( )  [inline]
```

Definition at line 312 of file QueryResult-header.hpp.

```
00312 {}
```

6.126.2.2 ResultSet() [2/2]

```
template<class Sorter >
minerule::QueryResult::ResultSet< Sorter >::ResultSet (
            const ResultSet< Sorter > & rhs )  [inline]
```

Definition at line 313 of file QueryResult-header.hpp.

```
00313 {}
```

6.126.3 Member Function Documentation

6.126.3.1 inplace_intersect()

```
template<class Sorter >
QueryResult::ResultSet< Sorter > & minerule::QueryResult::ResultSet< Sorter >::inplace_↩
intersect (
            const ResultSet< Sorter > & rhs )
```

Definition at line 66 of file QueryResult-impl.hpp.

```
00066
               {
00067                  QueryResult::ResultSet<Sorter> tmp;
00068                  set_intersection( this->begin(), this->end(), rhs.begin(),rhs.end(),
00069                         std::insert_iterator<ResultSet<Sorter> >( tmp, tmp.begin() ), Sorter());
00070
00071                  *this = tmp;
00072                  return *this;
00073          }
```

6.126.3.2 inplace_union()

```
template<class Sorter >
QueryResult::ResultSet< Sorter > & minerule::QueryResult::ResultSet< Sorter >::inplace_union
(
            const ResultSet< Sorter > & rhs )
```

Definition at line 76 of file QueryResult-impl.hpp.

```
00076
              {
00077
00078                  copy( rhs.begin(), rhs.end(),
00079                         std::insert_iterator<ResultSet<Sorter> >(*this, this->begin()));
00080                  return *this;
00081          }
```

6.126.3.3 load()

```
template<class Sorter >
void minerule::QueryResult::ResultSet< Sorter >::load (
            const std::string & qryName,
            double sup = -1,
            double con = -1 )
```

Definition at line 28 of file QueryResult-impl.hpp.

```
00028                                                                                                {
00029                  QueryResult::Iterator qit;
00030                  OptimizerCatalogue::getMRQueryResultIterator(qryName,qit,sup,con);
00031
00032                  while( qit.next() ) {
00033                         Rule r;
00034                         qit.getRule(r);
00035
00036                         std::set<Rule,Sorter>::insert(r);
00037                  }
00038          }
```

6.126.3.4 operator&=()

```
template<class Sorter >
QueryResult::ResultSet< Sorter > & minerule::QueryResult::ResultSet< Sorter >::operator&= (
            const ResultSet< Sorter > & rhs )
```

Definition at line 84 of file QueryResult-impl.hpp.

```
00084
        {
00085                 return inplace_intersect(rhs);
00086         }
```

6.126.3.5 operator"|=()

```
template<class Sorter >
QueryResult::ResultSet< Sorter > & minerule::QueryResult::ResultSet< Sorter >::operator|= (
            const ResultSet< Sorter > & rhs )
```

Definition at line 88 of file QueryResult-impl.hpp.

```
00088
        {
00089                 return inplace_union(rhs);
00090         }
```

6.126.3.6 save()

```
template<class Sorter >
void minerule::QueryResult::ResultSet< Sorter >::save (
            const ParsedMinerule & mr ) const
```

Definition at line 94 of file QueryResult-impl.hpp.

```
00094                                                                                {
00095                 Connection connection;
00096                 connection.setOutTableName(mr.tab_result);
00097                 connection.setBodyCardinalities(mr.bodyCardinalities);
00098                 connection.setHeadCardinalities(mr.headCardinalities);
00099                 connection.useMRDBConnection(
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00100
00101                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(), mr));
00102
00103                 typename ResultSet<Sorter>::const_iterator it;
00104                 for(it=this->begin();it!=this->end();it++) {
00105                         connection.insert( it->getBody(), it->getHead(), it->getSupport(),
      it->getConfidence() );
00106                 }
00107         }
```

6.126.4 Field Documentation

6.126.4.1 keys

```
K std::set< K >::keys  [inherited]
```

STL member.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp
- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-impl.hpp

6.127 mrdb::ResultSet Class Reference

```
#include <ResultSet.hpp>
```

Public Member Functions

- virtual ~ResultSet ()
- virtual bool isBeforeFirst () const =0
- virtual bool isAfterLast () const =0
- virtual bool next ()=0
- virtual bool getBoolean (int columnIndex)=0
- virtual float getFloat (int columnIndex)=0
- virtual double getDouble (int columnIndex)=0
- virtual int getInt (int columnIndex)=0
- virtual long getLong (int columnIndex)=0
- virtual std::string getString (int columnIndex)=0
- virtual mrdb::ResultSetMetaData * getMetaData ()=0

6.127.1 Detailed Description

Definition at line 8 of file ResultSet.hpp.

6.127.2 Constructor & Destructor Documentation

6.127.2.1 ~ResultSet()

```
virtual mrdb::ResultSet::~ResultSet ( )  [inline], [virtual]
```

Definition at line 10 of file ResultSet.hpp.

```
00010 {}
```

6.127.3 Member Function Documentation

6.127.3.1 getBoolean()

```
virtual bool mrdb::ResultSet::getBoolean (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type bool) at the current row and given column index.

6.127.3.2 getDouble()

```
virtual double mrdb::ResultSet::getDouble (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type double) at the current row and given column index.

6.127.3.3 getFloat()

```
virtual float mrdb::ResultSet::getFloat (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type float) at the current row and given column index.

6.127.3.4 getInt()

```
virtual int mrdb::ResultSet::getInt (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type int) at the current row and given column index.

6.127.3.5 getLong()

```
virtual long mrdb::ResultSet::getLong (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type long) at the current row and given column index.

6.127.3.6 getMetaData()

```
virtual mrdb::ResultSetMetaData * mrdb::ResultSet::getMetaData ( )  [pure virtual]
```

THREAD SAFETY: It returns a ResultSetMetaData that should be used by the same thread that owns this object.

Returns

the meta data about this result set

6.127.3.7 getString()

```
virtual std::string mrdb::ResultSet::getString (
            int columnIndex )  [pure virtual]
```

Returns

the value (assuming it is of type string) at the current row and given column index.

6.127.3.8 isAfterLast()

```
virtual bool mrdb::ResultSet::isAfterLast ( ) const  [pure virtual]
```

Returns

true if the current result set is after the last record.

6.127.3.9 isBeforeFirst()

```
virtual bool mrdb::ResultSet::isBeforeFirst ( ) const  [pure virtual]
```

Returns

true if the current result set is before the first record.

6.127.3.10 next()

```
virtual bool mrdb::ResultSet::next ( )  [pure virtual]
```

Move the result set to the next available result row. Upon creation the result set is positioned just before the first record, so that a call to this method is necessary *before* accessing to the first row.

Returns

true if the new position is inside the result set

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSet.hpp

6.128 mrdb::ResultSetMetaData Class Reference

```
#include <ResultSetMetaData.hpp>
```

Public Member Functions

- virtual ~ResultSetMetaData ()
- virtual int getColumnCount () const =0
- virtual std::string getColumnName (int column) const =0
- virtual int getColumnType (int column) const =0
- virtual int getPrecision (int column) const =0
- virtual int getScale (int column) const =0
- virtual std::string getColumnTypeName (int column)=0

6.128.1 Detailed Description

Definition at line 7 of file ResultSetMetaData.hpp.

6.128.2 Constructor & Destructor Documentation

6.128.2.1 ~ResultSetMetaData()

```
virtual mrdb::ResultSetMetaData::~ResultSetMetaData ( )  [inline], [virtual]
```

Definition at line 9 of file ResultSetMetaData.hpp.

```
00009 {}
```

6.128.3 Member Function Documentation

6.128.3.1 getColumnCount()

```
virtual int mrdb::ResultSetMetaData::getColumnCount ( ) const  [pure virtual]
```

Returns

the number of columns of a result set

6.128.3.2 getColumnName()

```
virtual std::string mrdb::ResultSetMetaData::getColumnName (
            int column ) const  [pure virtual]
```

Parameters

column	the desired column index (starting from 1)

Returns

the name of the specified column

6.128.3.3 getColumnType()

```
virtual int mrdb::ResultSetMetaData::getColumnType (
            int column ) const  [pure virtual]
```

Parameters

column	the desired column index (starting from 1)

Returns

the column type of the specified column.

6.128.3.4 getColumnTypeName()

```
virtual std::string mrdb::ResultSetMetaData::getColumnTypeName (
            int column )  [pure virtual]
```

Parameters

column	the desired column index (starting from 1)

Returns

the name of the column SQL type

6.128.3.5 getPrecision()

```
virtual int mrdb::ResultSetMetaData::getPrecision (
            int column ) const  [pure virtual]
```

Parameters

column	the desired column index (starting from 1)

Returns

the column precision of the specified column.

6.128.3.6 getScale()

```
virtual int mrdb::ResultSetMetaData::getScale (
            int column ) const  [pure virtual]
```

Parameters

column	the desired column index (starting from 1)

Returns

the column scale of the specified column.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSetMetaData.hpp

6.129 minerule::Rule Class Reference

`#include <Rule.hpp>`

Public Member Functions

- Rule ()
- Rule (const Rule &rule)
- ~Rule ()
- bool operator< (const Rule &rhs) const
- bool operator<= (const Rule &rhs) const
- bool operator== (const Rule &rhs) const
- bool operator> (const Rule &rhs) const
- bool operator>= (const Rule &rhs) const
- bool operator!= (const Rule &rhs) const
- Rule & operator= (const Rule &rule)
- void setBody (ItemSet *b)
- const ItemSet & getBody () const
- ItemSet & getBody ()
- void setHead (ItemSet *h)
- const ItemSet & getHead () const
- ItemSet & getHead ()
- void setSupport (double s)
- double getSupport () const
- void setConfidence (double c)
- double getConfidence () const
- void setBodyId (unsigned int bid)
- void setHeadId (unsigned int hid)
- unsigned int getBodyId () const
- unsigned int getHeadId () const

Protected Member Functions

- void clear ()

6.129.1 Detailed Description

Definition at line 23 of file Rule.hpp.

6.129.2 Constructor & Destructor Documentation

6.129.2.1 Rule() [1/2]

```
minerule::Rule::Rule ( )  [inline]
```

Definition at line 43 of file Rule.hpp.

```
00043 : body(NULL), head(NULL), support(0), confidence(0) {}
```

6.129.2.2 Rule() [2/2]

```
minerule::Rule::Rule (
            const Rule & rule )  [inline]
```

Definition at line 45 of file Rule.hpp.

```
00045                                   : support(rule.support), confidence(rule.confidence),
       bodyId(rule.bodyId), headId(rule.headId) {
00046                         if(rule.body!=NULL) {
00047                                 body= new ItemSet(*rule.body);
00048                         }
00049
00050                         if(rule.head!=NULL) {
00051                                 head= new ItemSet(*rule.head);
00052                         }
00053                 }
```

6.129.2.3 ~Rule()

```
minerule::Rule::~Rule ( )  [inline]
```

Definition at line 55 of file Rule.hpp.

```
00055                         {
00056                         clear();
00057                 }
```

6.129.3 Member Function Documentation

6.129.3.1 clear()

```
void minerule::Rule::clear ( )  [inline], [protected]
```

Definition at line 32 of file Rule.hpp.

```
00032                                 {
00033                         if(body!=NULL)
00034                                 delete body;
00035                         if(head!=NULL)
00036                                 delete head;
00037
00038                         body=NULL;
00039                         head=NULL;
00040                 }
```

6.129.3.2 getBody() [1/2]

```
ItemSet & minerule::Rule::getBody ( )  [inline]
```

Definition at line 108 of file Rule.hpp.

```
00108                                   {
00109                         assert(body!=NULL);
00110                         return *body;
00111                 }
```

6.129.3.3 getBody() [2/2]

```
const ItemSet & minerule::Rule::getBody ( ) const  [inline]
```

Definition at line 103 of file Rule.hpp.

```
00103                                                 {
00104                         assert(body!=NULL);
00105                         return *body;
00106                 }
```

6.129.3.4 getBodyId()

```
unsigned int minerule::Rule::getBodyId ( ) const  [inline]
```

Definition at line 149 of file Rule.hpp.

```
00149 { return bodyId; }
```

6.129.3.5 getConfidence()

```
double minerule::Rule::getConfidence ( ) const  [inline]
```

Definition at line 142 of file Rule.hpp.

```
00142                                       {
00143                         return confidence;
00144                 }
```

6.129.3.6 getHead() [1/2]

```
ItemSet & minerule::Rule::getHead ( )  [inline]
```

Definition at line 124 of file Rule.hpp.

```
00124                                   {
00125                         assert(head!=NULL);
00126                         return *head;
00127                 }
```

6.129.3.7 getHead() [2/2]

```
const ItemSet & minerule::Rule::getHead ( ) const  [inline]
```

Definition at line 119 of file Rule.hpp.

```
00119                                       {
00120                         assert(head!=NULL);
00121                         return *head;
00122                 }
```

6.129.3.8 getHeadId()

```
unsigned int minerule::Rule::getHeadId ( ) const  [inline]
```

Definition at line 150 of file Rule.hpp.

```
00150 { return headId; }
```

6.129.3.9 getSupport()

```
double minerule::Rule::getSupport ( ) const  [inline]
```

Definition at line 134 of file Rule.hpp.

```
00134                                       {
00135                         return support;
00136                 }
```

6.129.3.10 operator"!=()

```
bool minerule::Rule::operator!= (
            const Rule & rhs ) const  [inline]
```

Definition at line 79 of file Rule.hpp.

```
00079                                                       {
00080                         return ! (*this == rhs);
00081                 }
```

6.129.3.11 operator<()

```
bool minerule::Rule::operator< (
            const Rule & rhs ) const  [inline]
```

Definition at line 59 of file Rule.hpp.

```
00059                                                     {
00060                         return *this->body < *rhs.body || ( *this->body == *rhs.body && *this->head <
       *rhs.head );
00061                 }
```

6.129.3.12 operator<=()

```
bool minerule::Rule::operator<= (
            const Rule & rhs ) const  [inline]
```

Definition at line 63 of file Rule.hpp.

```
00063                                                   {
00064                         return *this->body <= *rhs.body || (*this->body == *rhs.body && *this->head <=
      *rhs.head);
00065                 }
```

6.129.3.13 operator=()

```
Rule & minerule::Rule::operator= (
            const Rule & rule )  [inline]
```

Definition at line 83 of file Rule.hpp.

```
00083                                                   {
00084                         clear();
00085                         support=rule.support;
00086                         confidence=rule.confidence;
00087
00088                         if(rule.body!=NULL) {
00089                                 body= new ItemSet(*rule.body);
00090                         }
00091
00092                         if(rule.head!=NULL) {
00093                                 head= new ItemSet(*rule.head);
00094                         }
00095
00096                         return *this;
00097                 }
```

6.129.3.14 operator==()

```
bool minerule::Rule::operator== (
            const Rule & rhs ) const  [inline]
```

Definition at line 67 of file Rule.hpp.

```
00067                                                   {
00068                         return *this->body == *rhs.body && *this->head == *rhs.head;
00069                 }
```

6.129.3.15 operator>()

```
bool minerule::Rule::operator> (
            const Rule & rhs ) const  [inline]
```

Definition at line 71 of file Rule.hpp.

```
00071                                                   {
00072                         return ! (*this <= rhs);
00073                 }
```

6.129.3.16 operator>=()

```
bool minerule::Rule::operator>= (
            const Rule & rhs ) const  [inline]
```

Definition at line 75 of file Rule.hpp.

```
00075                                                   {
00076                       return ! (*this < rhs);
00077                 }
```

6.129.3.17 setBody()

```
void minerule::Rule::setBody (
            ItemSet * b )  [inline]
```

Definition at line 99 of file Rule.hpp.

```
00099                                       {
00100                       body = b;
00101                 }
```

6.129.3.18 setBodyId()

```
void minerule::Rule::setBodyId (
            unsigned int bid )  [inline]
```

Definition at line 147 of file Rule.hpp.

```
00147 { bodyId = bid; }
```

6.129.3.19 setConfidence()

```
void minerule::Rule::setConfidence (
            double c )  [inline]
```

Definition at line 138 of file Rule.hpp.

```
00138                                          {
00139                       confidence = c;
00140                 }
```

6.129.3.20 setHead()

```
void minerule::Rule::setHead (
            ItemSet * h )  [inline]
```

Definition at line 115 of file Rule.hpp.

```
00115                                       {
00116                       head = h;
00117                 }
```

6.129.3.21 setHeadId()

```
void minerule::Rule::setHeadId (
            unsigned int hid )  [inline]
```

Definition at line 148 of file Rule.hpp.

```
00148 { headId = hid; }
```

6.129.3.22 setSupport()

```
void minerule::Rule::setSupport (
            double s )  [inline]
```

Definition at line 130 of file Rule.hpp.

```
00130                                       {
00131                         support = s;
00132                 }
```

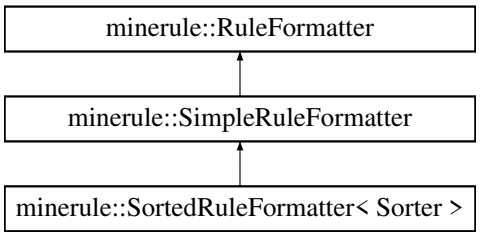
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/Rule.hpp

6.130 minerule::RuleFormatter Class Reference

```
#include <RuleFormatter.hpp>
```

Inheritance diagram for minerule::RuleFormatter:



Data Structures

- class FieldWidths

Public Member Functions

- RuleFormatter ()
- virtual ~RuleFormatter ()
- virtual std::string formatRule (const Rule &, bool includeSuppConf=true)=0
- virtual void printRule (const Rule &)=0
- void setSuppressLog (bool newVal)
- bool suppressLog () const
- void setFieldWidths (const FieldWidths &f)
- FieldWidths & fieldWidths ()
- const FieldWidths & fieldWidths () const
- virtual void postExec ()=0

Static Protected Member Functions

- static std::string quote (const std::string &elem)
- static std::string quoteElems (const ItemSet &elems)

Protected Attributes

- bool _suppressLog
- std::string _bhSep
- FieldWidths _fieldWidths

6.130.1 Detailed Description

This is the base class for all RuleFormatter objects

Definition at line 29 of file RuleFormatter.hpp.

6.130.2 Constructor & Destructor Documentation

6.130.2.1 RuleFormatter()

```
minerule::RuleFormatter::RuleFormatter ( )  [inline]
```

Definition at line 51 of file RuleFormatter.hpp.

```
00051                                : _suppressLog(false), _bhSep("=>"), _fieldWidths(30,30,9,9) {
00052                }
```

6.130.2.2 ~RuleFormatter()

```
virtual minerule::RuleFormatter::~RuleFormatter ( )  [inline], [virtual]
```

Definition at line 54 of file RuleFormatter.hpp.

```
00054 {}
```

6.130.3 Member Function Documentation

6.130.3.1 fieldWidths() [1/2]

```
FieldWidths & minerule::RuleFormatter::fieldWidths ( )  [inline]
```

Definition at line 63 of file RuleFormatter.hpp.

```
00063 { return _fieldWidths; }
```

6.130.3.2 fieldWidths() [2/2]

```
const FieldWidths & minerule::RuleFormatter::fieldWidths ( ) const  [inline]
```

Definition at line 64 of file RuleFormatter.hpp.

```
00064 { return _fieldWidths; }
```

6.130.3.3 formatRule()

```
virtual std::string minerule::RuleFormatter::formatRule (
            const Rule & ,
            bool includeSuppConf = true )  [pure virtual]
```

Implemented in minerule::SimpleRuleFormatter.

6.130.3.4 postExec()

```
virtual void minerule::RuleFormatter::postExec ( )  [pure virtual]
```

Implemented in minerule::SimpleRuleFormatter, and minerule::SortedRuleFormatter< Sorter >.

6.130.3.5 printRule()

```
virtual void minerule::RuleFormatter::printRule (
            const Rule &  )  [pure virtual]
```

Implemented in minerule::SimpleRuleFormatter, and minerule::SortedRuleFormatter< Sorter >.

6.130.3.6 quote()

```
std::string minerule::RuleFormatter::quote (
            const std::string & elem )  [static], [protected]
```

Definition at line 22 of file RuleFormatter.cpp.

```
00022                                                   {
00023                 return "["+elem+"]";
00024         }
```

6.130.3.7 quoteElems()

```
std::string minerule::RuleFormatter::quoteElems (
            const ItemSet & elems )  [static], [protected]
```

Definition at line 26 of file RuleFormatter.cpp.

```
00026                                                                   {
00027                 std::string result;
00028
00029                 ItemSet::const_iterator it=elems.begin();
00030
00031                 if(it!=elems.end()) {
00032                         result = quote(it->asString());
00033                         it++;
00034                 }
00035
00036                 for(; it!=elems.end(); it++) {
00037                         result +=  ", " + quote(it->asString());
00038                 }
00039
00040                 return result;
00041         }
```

6.130.3.8 setFieldWidths()

```
void minerule::RuleFormatter::setFieldWidths (
            const FieldWidths & f )  [inline]
```

Definition at line 62 of file RuleFormatter.hpp.

```
00062 { _fieldWidths = f; }
```

6.130.3.9 setSuppressLog()

```
void minerule::RuleFormatter::setSuppressLog (
            bool newVal )  [inline]
```

Definition at line 59 of file RuleFormatter.hpp.

```
00059 {  _suppressLog = newVal; }
```

6.130.3.10 suppressLog()

```
bool minerule::RuleFormatter::suppressLog ( ) const  [inline]
```

Definition at line 60 of file RuleFormatter.hpp.

```
00060 { return _suppressLog; }
```

6.130.4 Field Documentation

6.130.4.1 _bhSep

```
std::string minerule::RuleFormatter::_bhSep  [protected]
```

Definition at line 44 of file RuleFormatter.hpp.

6.130.4.2 _fieldWidths

```
FieldWidths minerule::RuleFormatter::_fieldWidths  [protected]
```

Definition at line 45 of file RuleFormatter.hpp.

6.130.4.3 _suppressLog

```
bool minerule::RuleFormatter::_suppressLog  [protected]
```

Definition at line 42 of file RuleFormatter.hpp.

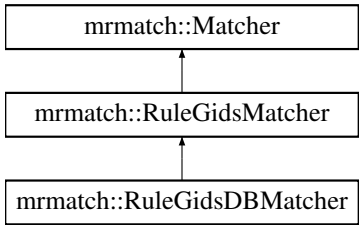
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp
- /Users/esposito/Software/minerule/src/Result/RuleFormatter.cpp

6.131 mrmatch::RuleGidsDBMatcher Class Reference

```
#include <RuleGidsDBMatcher.hpp>
```

Inheritance diagram for mrmatch::RuleGidsDBMatcher:



Public Member Functions

- RuleGidsDBMatcher (const std::string &outTableName, const minerule::ParsedMinerule &minerule)
- virtual ~RuleGidsDBMatcher ()
- virtual void printMatches () const
- virtual minerule::Rule & addRule ()
- virtual void match (minerule::SourceTable &st)
- virtual void matchWithCrossProduct (minerule::SourceTable &st)
- virtual void matchWithoutCrossProduct (minerule::SourceTable &st)

Static Public Member Functions

- static Matcher * newMatcher (const Options &opts, const minerule::ParsedMinerule &p)

Protected Types

- typedef std::vector< minerule::ItemType > Gids
- typedef std::pair< minerule::Rule, Gids > RuleGids
- typedef std::vector< RuleGids > RuleGidsVector

Protected Member Functions

- void createOutputTable (const minerule::ItemType &item) const
- void createOutputTableIndex (const std::string &indexName, const std::string &indexCols) const
- virtual void matchItemTransaction (const minerule::ItemType &gid, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &bodies, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &heads)
- virtual void matchRuleTransaction (const minerule::ItemType &gid, const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > &transaction)
- std::string formatGids (const Gids &gids) const

Protected Attributes

- std::string _outTableName
- minerule::ParsedMinerule _minerule
- RuleGidsVector _ruleGidsVector

6.131.1 Detailed Description

Definition at line 22 of file RuleGidsDBMatcher.hpp.

6.131.2 Member Typedef Documentation

6.131.2.1 Gids

```
typedef std::vector<minerule::ItemType> mrmatch::RuleGidsMatcher::Gids  [protected], [inherited]
```

Definition at line 25 of file RuleGidsMatcher.hpp.

6.131.2.2 RuleGids

```
typedef std::pair< minerule::Rule, Gids > mrmatch::RuleGidsMatcher::RuleGids  [protected],
[inherited]
```

Definition at line 26 of file RuleGidsMatcher.hpp.

6.131.2.3 RuleGidsVector

```
typedef std::vector< RuleGids > mrmatch::RuleGidsMatcher::RuleGidsVector  [protected], [inherited]
```

Definition at line 27 of file RuleGidsMatcher.hpp.

6.131.3 Constructor & Destructor Documentation

6.131.3.1 RuleGidsDBMatcher()

```
mrmatch::RuleGidsDBMatcher::RuleGidsDBMatcher (
            const std::string & outTableName,
            const minerule::ParsedMinerule & minerule )  [inline]
```

Definition at line 30 of file RuleGidsDBMatcher.hpp.

```
00031                       : RuleGidsMatcher(), _outTableName(outTableName), _minerule(minerule) {}
```

6.131.3.2 ~RuleGidsDBMatcher()

```
virtual mrmatch::RuleGidsDBMatcher::~RuleGidsDBMatcher ( )  [inline], [virtual]
```

Definition at line 32 of file RuleGidsDBMatcher.hpp.

```
00032 {}
```

6.131.4 Member Function Documentation

6.131.4.1 addRule()

```
Rule & mrmatch::RuleGidsMatcher::addRule ( )  [virtual], [inherited]
```

Implements mrmatch::Matcher.

Definition at line 26 of file RuleGidsMatcher.cpp.

```
00026                                   {
00027                 _ruleGidsVector.push_back( RuleGids() );
00028                 return _ruleGidsVector.back().first;
00029         }
```

6.131.4.2 createOutputTable()

```
void mrmatch::RuleGidsDBMatcher::createOutputTable (
            const minerule::ItemType & item ) const  [protected]
```

Definition at line 24 of file RuleGidsDBMatcher.cpp.

```
00024                                                                                {
00025                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00026                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00027                 minerule::SourceRowMetaInfo metaInfo(con, _minerule);
00028
00029                 minerule::MRLog() « "Creating dest table: " « _outTableName « std::endl;
00030                 state->execute("\
00031                         CREATE TABLE " + _outTableName + " \
00032                                 (\
00033                                   bodyid integer,\
00034                                   headid integer," +
00035                                   metaInfo.getGroup().getSQLDataDefinition() + ");");
00036
00037                 std::string colNamesWithCommas = metaInfo.getGroup().getSQLColumnNames();
00038                 std::string colNamesWithUnderscores = minerule::StringUtils::join(
      minerule::StringUtils::split(colNamesWithCommas, ","), "_" );
00039
00040                 createOutputTableIndex(_outTableName + "_bodyhead_index", "(bodyid,headid)");
00041                 createOutputTableIndex(_outTableName + "_" +colNamesWithUnderscores + "_index", "("+
      colNamesWithCommas +")");
00042         }
```

6.131.4.3 createOutputTableIndex()

```
void mrmatch::RuleGidsDBMatcher::createOutputTableIndex (
            const std::string & indexName,
            const std::string & indexCols ) const  [protected]
```

Definition at line 44 of file RuleGidsDBMatcher.cpp.

```
00044
                {
00045                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00046                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00047
00048                 minerule::MRLog() « "Building index on body/head..." « std::endl;
00049                 state->execute("\
00050                         CREATE INDEX " + indexName +
00051                         " ON " + _outTableName + indexCols);
00052         }
```

6.131.4.4 formatGids()

```
std::string mrmatch::RuleGidsMatcher::formatGids (
            const Gids & gids ) const  [protected], [inherited]
```

Definition at line 49 of file RuleGidsMatcher.cpp.

```
00049                                                                {
00050                 std::stringstream str;
00051                 for(std::vector<ItemType>::const_iterator it = gids.begin(); it!=gids.end(); ++it) {
00052                         str « *it « " ";
00053                 }
00054
00055                 return StringUtils::toBold(StringUtils::toGreen(str.str()));
00056         }
```

6.131.4.5 match()

```
virtual void mrmatch::Matcher::match (
            minerule::SourceTable & st )  [inline], [virtual], [inherited]
```

Definition at line 39 of file Matcher.hpp.

```
00039                                                       {
00040                       st.usesCrossProduct() ? matchWithCrossProduct(st) :
      matchWithoutCrossProduct(st);
00041               }
```

6.131.4.6 matchItemTransaction()

```
void mrmatch::RuleGidsMatcher::matchItemTransaction (
            const minerule::ItemType & gid,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & bodies,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & heads )
[protected], [virtual], [inherited]
```

Implements mrmatch::Matcher.

Definition at line 31 of file RuleGidsMatcher.cpp.

```
00031                                                                                {
00032                       for(RuleGidsVector::iterator ruleIt = _ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00033                               if( RulesMatcher::match( ruleIt->first, bodies, heads ) ) {
00034                                       ruleIt->second.push_back( gid );
00035                               }
00036                       }
00037       }
```

6.131.4.7 matchRuleTransaction()

```
void mrmatch::RuleGidsMatcher::matchRuleTransaction (
            const minerule::ItemType & gid,
            const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > & transaction
)  [protected], [virtual], [inherited]
```

Implements mrmatch::Matcher.

Definition at line 39 of file RuleGidsMatcher.cpp.

```
00039                                       {
00040                       for(RuleGidsVector::iterator ruleIt=_ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00041                               if( RulesMatcher::match(ruleIt->first, transaction) ) {
00042                                       ruleIt->second.push_back( gid );
00043                               }
00044                       }
00045       }
```

6.131.4.8 matchWithCrossProduct()

```
void mrmatch::Matcher::matchWithCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 38 of file Matcher.cpp.

```
00038                                                       {
00039                 SourceTable::Iterator it = st.newIterator(SourceTable::FullIterator);
00040 
00041                 while(!it.isAfterLast()) {
00042                         ItemType gid = it->getGroup();
00043 
00044                         RuleTransaction<RulesMatcher::RuleSetType> transaction;
00045                         transaction.load(gid, it);
00046 
00047                         matchRuleTransaction(gid, transaction);
00048                 }
00049         }
```

6.131.4.9 matchWithoutCrossProduct()

```
void mrmatch::Matcher::matchWithoutCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 51 of file Matcher.cpp.

```
00051                                                          {
00052                 SourceTable::Iterator bodyIt = st.newIterator(SourceTable::BodyIterator);
00053                 SourceTable::Iterator headIt = st.newIterator(SourceTable::HeadIterator);
00054 
00055                 while(!bodyIt.isAfterLast()) {
00056                         ItemType gid = bodyIt->getGroup();
00057 
00058                         ItemTransaction<RulesMatcher::ItemSetType> bodies;
00059                         ItemTransaction<RulesMatcher::ItemSetType> heads;
00060 
00061                         bodies.loadBody(gid, bodyIt);                   // this advances the body
      iterator
00062 
00063                         if( !TransactionBase<RulesMatcher::ItemSetType>::findGid(gid, headIt) ) {//
      positioning the head iterator
00064                                 break;                                                                  // no
      more heads to load
00065                         }
00066 
00067                         heads.loadHead(gid, headIt);                    // loading the heads
00068 
00069                         matchItemTransaction(gid, bodies, heads);
00070                 } // while
00071 
00072         } // matchWithoutCrossProduct
```

6.131.4.10 newMatcher()

```
Matcher * mrmatch::Matcher::newMatcher (
            const Options & opts,
            const minerule::ParsedMinerule & p )  [static], [inherited]
```

Definition at line 25 of file Matcher.cpp.

```
00025                                                                                   {
00026                 MatcherSpecs specs = opts.matcherSpecs();
00027                 if((specs & MatchOutputMask) == OutOnDB) {
00028                         return new RuleGidsDBMatcher(opts.getMatchOutputTableName(), minerule);
00029                 }
00030 
00031                 switch(specs & MatchKindMask) {
00032                         case RuleGids: return new RuleGidsMatcher();
00033                         case GidRules: return new GidRulesMatcher();
00034                         default: throw MineruleException( MR_ERROR_INTERNAL, "Unknown or unsupported
      matcher kind");
00035                 }
00036         }
```

6.131.4.11 printMatches()

```
void mrmatch::RuleGidsDBMatcher::printMatches ( ) const  [virtual]
```

Reimplemented from mrmatch::RuleGidsMatcher.

Definition at line 55 of file RuleGidsDBMatcher.cpp.

```
00055                                                   {
00056                 bool outTableCreated = false;
00057 
00058                 minerule::MRLogPusher _("Saving matches onto the database...");
00059                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00060 
00061                 minerule::MRLog() « "Inserting rules..." « std::endl;
00062                 std::auto_ptr<mrdb::PreparedStatement> ps(con->prepareStatement("\
00063                         INSERT INTO " + _outTableName + " VALUES (?,?,?);"));
00064 
00065                 for(RuleGidsVector::const_iterator it=_ruleGidsVector.begin();
      it!=_ruleGidsVector.end(); ++it) {
00066                         Gids::const_iterator gids_it;
00067                         for( gids_it = it->second.begin(); gids_it!=it->second.end(); ++gids_it ) {
00068                                 if(!outTableCreated) {
00069                                         createOutputTable(*gids_it);
00070                                         outTableCreated=true;
00071                                 }
00072 
00073                                 ps->setInt(1,it->first.getBodyId());
00074                                 ps->setInt(2,it->first.getHeadId());
00075                                 gids_it->setPreparedStatementParameters(ps.get(),3);
00076                                 ps->execute();
00077                         }
00078                 }
00079 
00080                 minerule::MRLog() « "All done!" « std::endl;
00081         }
```

6.131.5 Field Documentation

6.131.5.1 _minerule

```
minerule::ParsedMinerule mrmatch::RuleGidsDBMatcher::_minerule  [protected]
```

Definition at line 25 of file RuleGidsDBMatcher.hpp.

6.131.5.2 _outTableName

```
std::string mrmatch::RuleGidsDBMatcher::_outTableName  [protected]
```

Definition at line 24 of file RuleGidsDBMatcher.hpp.

6.131.5.3 _ruleGidsVector

```
RuleGidsVector mrmatch::RuleGidsMatcher::_ruleGidsVector  [protected], [inherited]
```

Definition at line 28 of file RuleGidsMatcher.hpp.

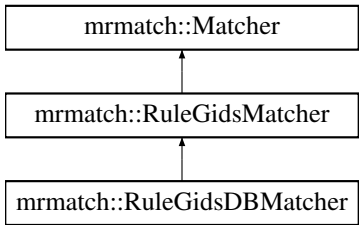
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.hpp
- /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.cpp

6.132 mrmatch::RuleGidsMatcher Class Reference

```
#include <RuleGidsMatcher.hpp>
```

Inheritance diagram for mrmatch::RuleGidsMatcher:



Public Member Functions

- RuleGidsMatcher ()
- virtual ~RuleGidsMatcher ()
- virtual minerule::Rule & addRule ()
- virtual void printMatches () const
- virtual void match (minerule::SourceTable &st)
- virtual void matchWithCrossProduct (minerule::SourceTable &st)
- virtual void matchWithoutCrossProduct (minerule::SourceTable &st)

Static Public Member Functions

- static Matcher * newMatcher (const Options &opts, const minerule::ParsedMinerule &p)

Protected Types

- typedef std::vector< minerule::ItemType > Gids
- typedef std::pair< minerule::Rule, Gids > RuleGids
- typedef std::vector< RuleGids > RuleGidsVector

Protected Member Functions

- virtual void matchItemTransaction (const minerule::ItemType &gid, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &bodies, const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > &heads)
- virtual void matchRuleTransaction (const minerule::ItemType &gid, const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > &transaction)
- std::string formatGids (const Gids &gids) const

Protected Attributes

- RuleGidsVector _ruleGidsVector

6.132.1 Detailed Description

Definition at line 23 of file RuleGidsMatcher.hpp.

6.132.2 Member Typedef Documentation

6.132.2.1 Gids

```
typedef std::vector<minerule::ItemType> mrmatch::RuleGidsMatcher::Gids  [protected]
```

Definition at line 25 of file RuleGidsMatcher.hpp.

6.132.2.2 RuleGids

```
typedef std::pair< minerule::Rule, Gids > mrmatch::RuleGidsMatcher::RuleGids  [protected]
```

Definition at line 26 of file RuleGidsMatcher.hpp.

6.132.2.3 RuleGidsVector

```
typedef std::vector< RuleGids > mrmatch::RuleGidsMatcher::RuleGidsVector  [protected]
```

Definition at line 27 of file RuleGidsMatcher.hpp.

6.132.3 Constructor & Destructor Documentation

6.132.3.1 RuleGidsMatcher()

```
mrmatch::RuleGidsMatcher::RuleGidsMatcher ( )  [inline]
```

Definition at line 36 of file RuleGidsMatcher.hpp.

```
00036 {}
```

6.132.3.2 ~RuleGidsMatcher()

```
virtual mrmatch::RuleGidsMatcher::~RuleGidsMatcher ( )  [inline], [virtual]
```

Definition at line 37 of file RuleGidsMatcher.hpp.

```
00037 {}
```

6.132.4 Member Function Documentation

6.132.4.1 addRule()

```
Rule & mrmatch::RuleGidsMatcher::addRule ( )  [virtual]
```

Implements mrmatch::Matcher.

Definition at line 26 of file RuleGidsMatcher.cpp.

```
00026                                   {
00027                 _ruleGidsVector.push_back( RuleGids() );
00028                 return _ruleGidsVector.back().first;
00029         }
```

6.132.4.2 formatGids()

```
std::string mrmatch::RuleGidsMatcher::formatGids (
            const Gids & gids ) const  [protected]
```

Definition at line 49 of file RuleGidsMatcher.cpp.

```
00049                                                                   {
00050                 std::stringstream str;
00051                 for(std::vector<ItemType>::const_iterator it = gids.begin(); it!=gids.end(); ++it) {
00052                         str « *it « " ";
00053                 }
00054
00055                 return StringUtils::toBold(StringUtils::toGreen(str.str()));
00056         }
```

6.132.4.3 match()

```
virtual void mrmatch::Matcher::match (
            minerule::SourceTable & st )  [inline], [virtual], [inherited]
```

Definition at line 39 of file Matcher.hpp.

```
00039                                                       {
00040                         st.usesCrossProduct() ? matchWithCrossProduct(st) :
      matchWithoutCrossProduct(st);
00041                 }
```

6.132.4.4 matchItemTransaction()

```
void mrmatch::RuleGidsMatcher::matchItemTransaction (
            const minerule::ItemType & gid,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & bodies,
            const minerule::ItemTransaction< minerule::RulesMatcher::ItemSetType > & heads )
[protected], [virtual]
```

Implements mrmatch::Matcher.

Definition at line 31 of file RuleGidsMatcher.cpp.

```
00031                                                                                   {
00032                         for(RuleGidsVector::iterator ruleIt = _ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00033                                 if( RulesMatcher::match( ruleIt->first, bodies, heads ) ) {
00034                                         ruleIt->second.push_back( gid );
00035                                 }
00036                         }
00037         }
```

6.132.4.5 matchRuleTransaction()

```
void mrmatch::RuleGidsMatcher::matchRuleTransaction (
            const minerule::ItemType & gid,
            const minerule::RuleTransaction< minerule::RulesMatcher::RuleSetType > & transaction
)  [protected], [virtual]
```

Implements mrmatch::Matcher.

Definition at line 39 of file RuleGidsMatcher.cpp.

```
00039                                           {
00040                         for(RuleGidsVector::iterator ruleIt=_ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00041                                 if( RulesMatcher::match(ruleIt->first, transaction) ) {
00042                                         ruleIt->second.push_back( gid );
00043                                 }
00044                         }
00045         }
```

6.132.4.6 matchWithCrossProduct()

```
void mrmatch::Matcher::matchWithCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 38 of file Matcher.cpp.

```
00038                                                       {
00039                 SourceTable::Iterator it = st.newIterator(SourceTable::FullIterator);
00040
00041                 while(!it.isAfterLast()) {
00042                         ItemType gid = it->getGroup();
00043
00044                         RuleTransaction<RulesMatcher::RuleSetType> transaction;
00045                         transaction.load(gid, it);
00046
00047                         matchRuleTransaction(gid, transaction);
00048                 }
00049         }
```

6.132.4.7 matchWithoutCrossProduct()

```
void mrmatch::Matcher::matchWithoutCrossProduct (
            minerule::SourceTable & st )  [virtual], [inherited]
```

Definition at line 51 of file Matcher.cpp.

```
00051                                                          {
00052                 SourceTable::Iterator bodyIt = st.newIterator(SourceTable::BodyIterator);
00053                 SourceTable::Iterator headIt = st.newIterator(SourceTable::HeadIterator);
00054
00055                 while(!bodyIt.isAfterLast()) {
00056                         ItemType gid = bodyIt->getGroup();
00057
00058                         ItemTransaction<RulesMatcher::ItemSetType> bodies;
00059                         ItemTransaction<RulesMatcher::ItemSetType> heads;
00060
00061                         bodies.loadBody(gid, bodyIt);                   // this advances the body
      iterator
00062
00063                         if( !TransactionBase<RulesMatcher::ItemSetType>::findGid(gid, headIt) ) {//
      positioning the head iterator
00064                                 break;                                                          // no
      more heads to load
00065                         }
00066
00067                         heads.loadHead(gid, headIt);                    // loading the heads
00068
00069                         matchItemTransaction(gid, bodies, heads);
00070                 } // while
00071
00072         } // matchWithoutCrossProduct
```

6.132.4.8 newMatcher()

```
Matcher * mrmatch::Matcher::newMatcher (
            const Options & opts,
            const minerule::ParsedMinerule & p )  [static], [inherited]
```

Definition at line 25 of file Matcher.cpp.

```
00025                                                                                   {
00026                 MatcherSpecs specs = opts.matcherSpecs();
00027                 if((specs & MatchOutputMask) == OutOnDB) {
00028                         return new RuleGidsDBMatcher(opts.getMatchOutputTableName(), minerule);
00029                 }
00030
00031                 switch(specs & MatchKindMask) {
00032                         case RuleGids: return new RuleGidsMatcher();
00033                         case GidRules: return new GidRulesMatcher();
00034                         default: throw MineruleException( MR_ERROR_INTERNAL, "Unknown or unsupported
      matcher kind");
00035                 }
00036         }
```

6.132.4.9 printMatches()

```
void mrmatch::RuleGidsMatcher::printMatches ( ) const  [virtual]
```

Implements mrmatch::Matcher.

Reimplemented in mrmatch::RuleGidsDBMatcher.

Definition at line 58 of file RuleGidsMatcher.cpp.

```
00058                                                {
00059                 SimpleRuleFormatter sf;
00060                 sf.setFieldWidths( SimpleRuleFormatter::FieldWidths(1,1,1,1) );
00061                 for(RuleGidsVector::const_iterator it=_ruleGidsVector.begin();
      it!=_ruleGidsVector.end(); ++it) {
00062                         MRLog()    « StringUtils::toBold("Rule: ") « sf.formatRule(it->first) « " "
00063                                                « StringUtils::toBold("\tGids: ") «
      formatGids(it->second) « std::endl;
00064                 }
00065         }
```

6.132.5 Field Documentation

6.132.5.1 _ruleGidsVector

```
RuleGidsVector mrmatch::RuleGidsMatcher::_ruleGidsVector  [protected]
```

Definition at line 28 of file RuleGidsMatcher.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.hpp
- /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.cpp

6.133 minerule::RulesMatcher Class Reference

```
#include <RulesMatcher.hpp>
```

Public Types

- typedef std::set< ItemType > ItemSetType
- typedef std::set< std::pair< ItemType, ItemType > > RuleSetType

Static Public Member Functions

- static bool match (const Rule &r, const RuleTransaction< RuleSetType > &t)
- static bool match (const Rule &r, const ItemTransaction< ItemSetType > &bodySet, const ItemTransaction< ItemSetType > &headSet)

6.133.1 Detailed Description

Definition at line 23 of file RulesMatcher.hpp.

6.133.2 Member Typedef Documentation

6.133.2.1 ItemSetType

```
typedef std::set<ItemType> minerule::RulesMatcher::ItemSetType
```

Definition at line 25 of file RulesMatcher.hpp.

6.133.2.2 RuleSetType

```
typedef std::set< std::pair<ItemType, ItemType> > minerule::RulesMatcher::RuleSetType
```

Definition at line 26 of file RulesMatcher.hpp.

6.133.3 Member Function Documentation

6.133.3.1 match() [1/2]

```
bool minerule::RulesMatcher::match (
            const Rule & r,
            const ItemTransaction< ItemSetType > & bodySet,
            const ItemTransaction< ItemSetType > & headSet )  [static]
```

Definition at line 43 of file RulesMatcher.cpp.

```
00043                                   {
00044                 const ItemSet& body = r.getBody();
00045                 for( ItemSet::const_iterator it=body.begin(); it!=body.end();++it) {
00046                         if(bodySet.find(*it) == bodySet.end())
00047                                 return false;
00048                 }
00049
00050                 const ItemSet& head = r.getHead();
00051                 for( ItemSet::const_iterator it=head.begin(); it!=head.end();++it) {
00052                         if(headSet.find(*it) == headSet.end())
00053                                 return false;
00054                 }
00055
00056                 return true;
00057         }
```

6.133.3.2 match() [2/2]

```
bool minerule::RulesMatcher::match (
            const Rule & r,
            const RuleTransaction< RuleSetType > & t )  [static]
```

Definition at line 19 of file RulesMatcher.cpp.

```
00019                                                                                   {
00020                 std::set< ItemType > bodySet;
00021                 std::set< ItemType > headSet;
00022
00023                 for( RuleSetType::const_iterator it = t.begin(); it!=t.end(); ++it ) {
00024                         bodySet.insert(it->first);
00025                         headSet.insert(it->second);
00026                 }
00027
00028                 const ItemSet& body = r.getBody();
00029                 for( ItemSet::const_iterator bodyIt=body.begin(); bodyIt!=body.end();++bodyIt) {
00030                         if(bodySet.find(*bodyIt) == bodySet.end())
00031                                 return false;
00032                 }
00033
00034                 const ItemSet& head = r.getHead();
00035                 for( ItemSet::const_iterator headIt=head.begin(); headIt!=head.end();++headIt) {
00036                         if(headSet.find(*headIt) == headSet.end())
00037                                 return false;
00038                 }
00039
00040                 return true;
00041         }
```

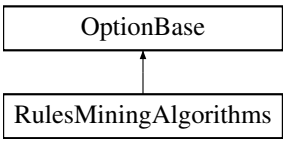
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Result/RulesMatcher.hpp
- /Users/esposito/Software/minerule/src/Result/RulesMatcher.cpp

6.134 RulesMiningAlgorithms Class Reference

```
#include <rulemining.hpp>
```

Inheritance diagram for RulesMiningAlgorithms:



Data Structures

- class FPGrowth
- class PartitionBase
- class PartitionWithClusters

Public Member Functions

- PartitionBase & getPartitionBase ()
- const PartitionBase & getPartitionBase () const
- PartitionWithClusters & getPartitionWithClusters ()
- const PartitionWithClusters & getPartitionWithClusters () const
- FPGrowth & getFPGrowth ()
- const FPGrowth & getFPGrowth () const
- virtual void setOption (const std::string &name, const std::string &value)
- virtual std::string className () const
- virtual OptionBase & subclassForName (const std::string &subclassName)
- RulesMiningAlgorithms ()
- RulesMiningAlgorithms (const RulesMiningAlgorithms &rhs)
- virtual ~RulesMiningAlgorithms ()
- void setPreferredAlgorithm (AlgorithmTypes type)
- AlgorithmTypes getPreferredAlgorithm () const

Data Fields

- PartitionBase partitionbase
- PartitionWithClusters partitiongeneralized
- FPGrowth fpgrowth
- AlgorithmTypes preferredAlgorithm

6.134.1 Detailed Description

Definition at line 16 of file rulemining.hpp.

6.134.2 Constructor & Destructor Documentation

6.134.2.1 RulesMiningAlgorithms() [1/2]

```
RulesMiningAlgorithms::RulesMiningAlgorithms ( )  [inline]
```

Definition at line 143 of file rulemining.hpp.

```
00143 : preferredAlgorithm(ATNone) {  }
```

6.134.2.2 RulesMiningAlgorithms() [2/2]

```
RulesMiningAlgorithms::RulesMiningAlgorithms (
            const RulesMiningAlgorithms & rhs )  [inline]
```

Definition at line 145 of file rulemining.hpp.

```
00145                                                    :
00146           partitionbase(rhs.partitionbase),
00147           partitiongeneralized(rhs.partitiongeneralized),
00148           fpgrowth(rhs.fpgrowth),
00149           preferredAlgorithm(rhs.preferredAlgorithm) {};
```

6.134.2.3 ~RulesMiningAlgorithms()

```
virtual RulesMiningAlgorithms::~RulesMiningAlgorithms ( )  [inline], [virtual]
```

Definition at line 151 of file rulemining.hpp.

```
00151 {}
```

6.134.3 Member Function Documentation

6.134.3.1 className()

```
virtual std::string RulesMiningAlgorithms::className ( ) const  [inline], [virtual]
```

Definition at line 124 of file rulemining.hpp.

```
00124                                                {
00125           return "rulesmining";
00126         }
```

6.134.3.2 getFPGrowth() [1/2]

```
FPGrowth & RulesMiningAlgorithms::getFPGrowth ( )  [inline]
```

Definition at line 112 of file rulemining.hpp.

```
00112                              {
00113           return fpgrowth;
00114         }
```

6.134.3.3 getFPGrowth() [2/2]

```
const FPGrowth & RulesMiningAlgorithms::getFPGrowth ( ) const  [inline]
```

Definition at line 117 of file rulemining.hpp.

```
00117                                    {
00118           return fpgrowth;
00119         }
```

6.134.3.4 getPartitionBase() [1/2]

```
PartitionBase & RulesMiningAlgorithms::getPartitionBase ( )  [inline]
```

Definition at line 92 of file rulemining.hpp.

```
00092                                  {
00093           return partitionbase;
00094         }
```

6.134.3.5 getPartitionBase() [2/2]

```
const PartitionBase & RulesMiningAlgorithms::getPartitionBase ( ) const  [inline]
```

Definition at line 97 of file rulemining.hpp.

```
00097                                       {
00098           return partitionbase;
00099         }
```

6.134.3.6 getPartitionWithClusters() [1/2]

```
PartitionWithClusters & RulesMiningAlgorithms::getPartitionWithClusters ( )  [inline]
```

Definition at line 102 of file rulemining.hpp.

```
00102                                          {
00103           return partitiongeneralized;
00104         }
```

6.134.3.7 getPartitionWithClusters() [2/2]

```
const PartitionWithClusters & RulesMiningAlgorithms::getPartitionWithClusters ( ) const  [inline]
```

Definition at line 107 of file rulemining.hpp.

```
00107                                                {
00108           return partitiongeneralized;
00109         }
```

6.134.3.8 getPreferredAlgorithm()

```
AlgorithmTypes RulesMiningAlgorithms::getPreferredAlgorithm ( ) const  [inline]
```

Definition at line 157 of file rulemining.hpp.

```
00157                                                 {
00158           return preferredAlgorithm;
00159         }
```

6.134.3.9 setOption()

```
virtual void RulesMiningAlgorithms::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.134.3.10 setPreferredAlgorithm()

```
void RulesMiningAlgorithms::setPreferredAlgorithm (
            AlgorithmTypes type )  [inline]
```

Definition at line 153 of file rulemining.hpp.

```
00153                                                   {
00154           preferredAlgorithm = type;
00155         }
```

6.134.3.11 subclassForName()

```
virtual OptionBase & RulesMiningAlgorithms::subclassForName (
            const std::string & subclassName )  [inline], [virtual]
```

Definition at line 128 of file rulemining.hpp.

```
00129           {
00130           if( subclassName=="partitionbase" ) {
00131             return partitionbase;
00132           } else if(subclassName=="partitionwithclusters") {
00133             return partitiongeneralized;
00134           } else if(subclassName=="fpgrowth") {
00135             return fpgrowth;
00136           } else {
00137             return OptionBase::subclassForName(subclassName);
00138           }
00139         }
```

6.134.4 Field Documentation

6.134.4.1 fpgrowth

```
FPGrowth RulesMiningAlgorithms::fpgrowth
```

Definition at line 87 of file rulemining.hpp.

6.134.4.2 partitionbase

```
PartitionBase RulesMiningAlgorithms::partitionbase
```

Definition at line 85 of file rulemining.hpp.

6.134.4.3 partitiongeneralized

```
PartitionWithClusters RulesMiningAlgorithms::partitiongeneralized
```

Definition at line 86 of file rulemining.hpp.

6.134.4.4 preferredAlgorithm

```
AlgorithmTypes RulesMiningAlgorithms::preferredAlgorithm
```

Definition at line 89 of file rulemining.hpp.

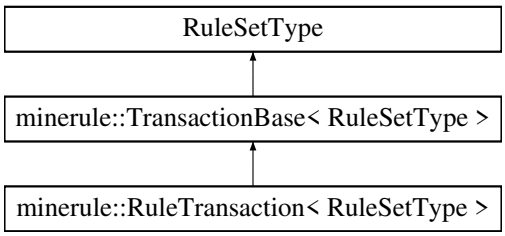
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp

6.135 minerule::RuleTransaction< RuleSetType > Class Template Reference

```
#include <Transaction.hpp>
```

Inheritance diagram for minerule::RuleTransaction< RuleSetType >:



Public Member Functions

- RuleTransaction ()
- void load (ItemType &gid, SourceTable::Iterator &it)

Static Public Member Functions

- static bool findGid (ItemType &gid, SourceTable::Iterator &it)

6.135.1 Detailed Description

template<class RuleSetType>
class minerule::RuleTransaction< RuleSetType >

Definition at line 71 of file Transaction.hpp.

6.135.2 Constructor & Destructor Documentation

6.135.2.1 RuleTransaction()

```
template<class RuleSetType >
minerule::RuleTransaction< RuleSetType >::RuleTransaction ( )  [inline]
```

Definition at line 73 of file Transaction.hpp.

```
00073 : TransactionBase<RuleSetType>() {}
```

6.135.3 Member Function Documentation

6.135.3.1 findGid()

```
static bool minerule::TransactionBase< RuleSetType >::findGid (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [static], [inherited]
```

Definition at line 32 of file Transaction.hpp.

```
00032                                                                    {
00033                         while( !it.isAfterLast() && gid > it->getGroup() ) {
00034                                 ++it;
00035                         }
00036
00037                         return !it.isAfterLast() && gid == it->getGroup();
00038                 }
```

6.135.3.2 load()

```
template<class RuleSetType >
void minerule::RuleTransaction< RuleSetType >::load (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline]
```

Definition at line 75 of file Transaction.hpp.

```
00075                                                                {
00076                                 while (!it.isAfterLast() && gid == it->getGroup()) {
00077
       RuleSetType::insert(RuleSetType::end(),std::pair<ItemType,ItemType>(it->getBody(),it->getHead()));
00078                                         ++it;
00079                                 }
00080             }
```

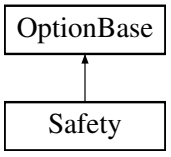
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/Transaction.hpp

6.136 Safety Class Reference

`#include <safety.hpp>`

Inheritance diagram for Safety:



Public Member Functions

- Safety ()
- virtual ~Safety ()
- virtual std::string className () const
- virtual void setOption (const std::string &name, const std::string &value)
- bool getOverwriteHomonymMinerules () const
- void setOverwriteHomonymMinerules (bool rhs)
- bool getAllowCascadeDeletes () const
- void setAllowCascadeDeletes (bool rhs)

6.136.1 Detailed Description

Definition at line 16 of file safety.hpp.

6.136.2 Constructor & Destructor Documentation

6.136.2.1 Safety()

```
Safety::Safety ( )  [inline]
```

Definition at line 20 of file safety.hpp.

```
00020           : overwriteHomonymMinerules(false),
00021                 allowCascadeDeletes(false) {}
```

6.136.2.2 ~Safety()

```
virtual Safety::~Safety ( )  [inline], [virtual]
```

Definition at line 22 of file safety.hpp.

```
00022 {}
```

6.136.3 Member Function Documentation

6.136.3.1 className()

```
virtual std::string Safety::className ( ) const  [inline], [virtual]
```

Definition at line 24 of file safety.hpp.

```
00024                                       {
00025         return "safety";
00026   }
```

6.136.3.2 getAllowCascadeDeletes()

```
bool Safety::getAllowCascadeDeletes ( ) const  [inline]
```

Definition at line 34 of file safety.hpp.

```
00034 { return allowCascadeDeletes; }
```

6.136.3.3 getOverwriteHomonymMinerules()

```
bool Safety::getOverwriteHomonymMinerules ( ) const  [inline]
```

Definition at line 31 of file safety.hpp.

```
00031 { return overwriteHomonymMinerules; };
```

6.136.3.4 setAllowCascadeDeletes()

```
void Safety::setAllowCascadeDeletes (
            bool rhs )  [inline]
```

Definition at line 35 of file safety.hpp.

```
00035 { allowCascadeDeletes = rhs; }
```

6.136.3.5 setOption()

```
virtual void Safety::setOption (
            const std::string & name,
            const std::string & value )  [virtual]
```

6.136.3.6 setOverwriteHomonymMinerules()

```
void Safety::setOverwriteHomonymMinerules (
            bool rhs )  [inline]
```

Definition at line 32 of file safety.hpp.

```
00032 { overwriteHomonymMinerules = rhs; };
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/safety.hpp

6.137 minerule::Scmp Class Reference

`#include <translatedtablestandardsql.hpp>`

Public Member Functions

- int operator() (const char *s1, const char *s2) const

6.137.1 Detailed Description

Definition at line 38 of file translatedtablestandardsql.hpp.

6.137.2 Member Function Documentation

6.137.2.1 operator()()

```
int minerule::Scmp::operator() (
            const char * s1,
            const char * s2 ) const  [inline]
```

Definition at line 40 of file translatedtablestandardsql.hpp.

```
00040                                                 {
00041     return std::strcmp(s1,s2)==0;
00042   }
```

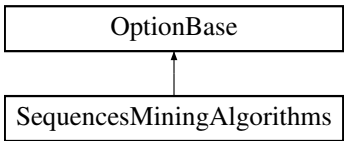
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtablestandardsql.hpp

6.138 SequencesMiningAlgorithms Class Reference

`#include <sequencemining.hpp>`

Inheritance diagram for SequencesMiningAlgorithms:



Public Member Functions

- SequencesMiningAlgorithms ()
- SequencesMiningAlgorithms (const SequencesMiningAlgorithms &rhs)
- virtual ~SequencesMiningAlgorithms ()
- void setPreferredAlgorithm (AlgorithmTypes type)
- AlgorithmTypes getPreferredAlgorithm () const
- virtual std::string className () const
- virtual OptionBase & subclassForName (const std::string &subclassName)
- virtual void setOption (const std::string &name, const std::string &value)

6.138.1 Detailed Description

Definition at line 16 of file sequencemining.hpp.

6.138.2 Constructor & Destructor Documentation

6.138.2.1 SequencesMiningAlgorithms() [1/2]

```
SequencesMiningAlgorithms::SequencesMiningAlgorithms ( )  [inline]
```

Definition at line 19 of file sequencemining.hpp.

```
00019 : preferredAlgorithm(ATNone) {  }
```

6.138.2.2 SequencesMiningAlgorithms() [2/2]

```
SequencesMiningAlgorithms::SequencesMiningAlgorithms (
            const SequencesMiningAlgorithms & rhs )  [inline]
```

Definition at line 21 of file sequencemining.hpp.

```
00021                                                                    :
00022          preferredAlgorithm(rhs.preferredAlgorithm) {}
```

6.138.2.3 ~SequencesMiningAlgorithms()

```
virtual SequencesMiningAlgorithms::~SequencesMiningAlgorithms ( )  [inline], [virtual]
```

Definition at line 24 of file sequencemining.hpp.

```
00024 {}
```

6.138.3 Member Function Documentation

6.138.3.1 className()

```
virtual std::string SequencesMiningAlgorithms::className ( ) const  [inline], [virtual]
```

Definition at line 34 of file sequencemining.hpp.

```
00034                                          {
00035           return "sequencesmining";
00036         }
```

6.138.3.2 getPreferredAlgorithm()

```
AlgorithmTypes SequencesMiningAlgorithms::getPreferredAlgorithm ( ) const  [inline]
```

Definition at line 30 of file sequencemining.hpp.

```
00030                                                    {
00031           return preferredAlgorithm;
00032         }
```

6.138.3.3 setOption()

```
virtual void SequencesMiningAlgorithms::setOption (
             const std::string & name,
             const std::string & value )  [virtual]
```

6.138.3.4 setPreferredAlgorithm()

```
void SequencesMiningAlgorithms::setPreferredAlgorithm (
             AlgorithmTypes type )  [inline]
```

Definition at line 26 of file sequencemining.hpp.

```
00026                                                  {
00027           preferredAlgorithm = type;
00028         }
```

6.138.3.5 subclassForName()

```
virtual OptionBase & SequencesMiningAlgorithms::subclassForName (
             const std::string & subclassName )  [inline], [virtual]
```

Definition at line 39 of file sequencemining.hpp.

```
00040           {
00041             return OptionBase::subclassForName(subclassName);
00042         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/miningalgorithms/sequencemining

6.139 minerule::SimplePredAnalyzer Class Reference

`#include <SimplePredAnalyzer.hpp>`

Static Public Member Functions

- static char getRelation (const char *Xop, const char *Xvalue, const char *Yop, const char *Yvalue, SQLUtils::Type)
- static bool isAttrOpValuePredicate (simple_pred *X, char *&attr, char *&value, bool &reverseOp)

Static Protected Member Functions

- static size_t getOperatorIndex (const char *op)
- static size_t getValuesRelationshipIndex (const char *value1, const char *value2, SQLUtils::Type type)

Static Protected Attributes

- static const char * opRelationTable [6][6]

6.139.1 Detailed Description

Definition at line 28 of file SimplePredAnalyzer.hpp.

6.139.2 Member Function Documentation

6.139.2.1 getOperatorIndex()

```
size_t minerule::SimplePredAnalyzer::getOperatorIndex (
            const char * op )  [static], [protected]
```

Definition at line 49 of file SimplePredAnalyzer.cpp.

```
00050     {
00051 
00052     if(op[0]=='<') {
00053       if(op[1]=='\0')
00054         return 0;
00055       if(op[1]=='=')
00056         return 1;
00057       if(op[1]=='>')
00058         return 5;
00059 
00060       throw MineruleException(MR_ERROR_INTERNAL, (std::string) "Operator "+op+" not recognized");
00061     } else if(op[0]=='>') {
00062       if(op[1]=='\0')
00063         return 4;
00064       if(op[1]=='=')
00065         return 3;
00066 
00067       throw MineruleException(MR_ERROR_INTERNAL,(std::string) "Operator"+op+" not recognized");
00068     } else  if(op[0]=='=') {
00069       if(op[1]=='\0')
00070         return 2;
00071     }
00072 
00073     throw MineruleException(MR_ERROR_INTERNAL,(std::string) "Operator"+op+" not recognized");
00074   }
```

6.139.2.2 getRelation()

```
char minerule::SimplePredAnalyzer::getRelation (
            const char * Xop,
            const char * Xvalue,
            const char * Yop,
            const char * Yvalue,
            SQLUtils::Type type )  [static]
```

Definition at line 125 of file SimplePredAnalyzer.cpp.

```
00130     {
00131
00132     size_t XopIndex = getOperatorIndex(Xop);
00133     size_t YopIndex = getOperatorIndex(Yop);
00134     size_t XYvalsIndex = getValuesRelationshipIndex(Xvalue,Yvalue,type);
00135     return opRelationTable[XopIndex][YopIndex][XYvalsIndex];
00136   }
```

6.139.2.3 getValuesRelationshipIndex()

```
size_t minerule::SimplePredAnalyzer::getValuesRelationshipIndex (
            const char * value1,
            const char * value2,
            SQLUtils::Type type )  [static], [protected]
```

Definition at line 76 of file SimplePredAnalyzer.cpp.

```
00079      {
00080
00081     double order;
00082     if( type==SQLUtils::String) {
00083       order=strcmp(value1,value2);
00084     } else if( type==SQLUtils::Numeric ) {
00085       order =
00086         Converter(value1).toDouble()-Converter(value2).toDouble();
00087     } else {
00088       throw MineruleException(MR_ERROR_INTERNAL,
00089                   "SimplePredAnalyzer still does not support SQL types"
00090                   " which differ fromstd::string and numeric");
00091     }
00092
00093     /*
00094      * It seems that the table is built using the order x>y x=y x<y...
00095      */
00096     if( order<0 )
00097       return 2;
00098     else if(order==0)
00099       return 1;
00100     else return 0;
00101   }
```

6.139.2.4 isAttrOpValuePredicate()

```
bool minerule::SimplePredAnalyzer::isAttrOpValuePredicate (
            simple_pred * X,
            char *& attr,
            char *& value,
            bool & reverseOp )  [static]
```

Definition at line 104 of file SimplePredAnalyzer.cpp.

```
00107                                                   {
00108     if(SQLUtils::isAttribute(X->val1)) {
```

```
00109       attr=X->val1;
00110       value=X->val2;
00111       reverseOp=false;
00112     } else if(SQLUtils::isAttribute(X->val2)) {
00113       attr=X->val2;
00114       value=X->val1;
00115       reverseOp=true;
00116     }
00117
00118     if(attr==NULL || SQLUtils::isAttribute(value))
00119       return false;
00120     else
00121       return true;
00122   }
```

6.139.3 Field Documentation

6.139.3.1 opRelationTable

```
const char * minerule::SimplePredAnalyzer::opRelationTable  [static], [protected]
```

Initial value:
=

```
    {
      {"lbr","lrr","l!!","/!!","/!!","/rr"},
      {"llr","lbr","ll!","/=!","/!!","/<r"},
      {"!!r","!rr","!b!","rr!","r!!","r!r"},
      {"!!/","!=/","!ll","rbl","rll","r>/"},
      {"!!/","!!/","!!l","rrl","rbl","rr/"},
      {"ll/","l</","l!l","/>l","/ll","/b/"}
    }
```

Definition at line 53 of file SimplePredAnalyzer.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/SimplePredAnalyzer.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/SimplePredAnalyzer.cpp

6.140 minerule::SimplePredicate Class Reference

```
#include <Predicate.hpp>
```

Public Member Functions

- const std::string & getVal1 () const
- const std::string & getVal2 () const
- const std::string & getOp () const
- bool operator== (const SimplePredicate &rhs) const
- bool operator< (const SimplePredicate &rhs) const
- void setVarId (size_t id)
- size_t getVarId () const

Static Public Member Functions

- static SimplePredicate & newSimplePredicate (const simple_pred *pred)
- static SimplePredicate & newSimplePredicate (const std::string &val1, const std::string &op, const std::string &val2)
- static void freeSimplePredicatePool ()

6.140.1 Detailed Description

Definition at line 34 of file Predicate.hpp.

6.140.2 Member Function Documentation

6.140.2.1 freeSimplePredicatePool()

```
void minerule::SimplePredicate::freeSimplePredicatePool ( )  [static]
```

Definition at line 62 of file Predicate.cpp.

```
00062                                          {
00063     PredicatePool::iterator it;
00064     for( it=predicatePool.begin(); it!=predicatePool.end(); it++ ) {
00065       delete *it;
00066     }
00067
00068     predicatePool.clear();
00069   }
```

6.140.2.2 getOp()

```
const std::string & minerule::SimplePredicate::getOp ( ) const  [inline]
```

Definition at line 89 of file Predicate.hpp.

```
00089                                     {
00090       return op;
00091     }
```

6.140.2.3 getVal1()

```
const std::string & minerule::SimplePredicate::getVal1 ( ) const  [inline]
```

Definition at line 81 of file Predicate.hpp.

```
00081                                       {
00082       return val1;
00083     }
```

6.140.2.4 getVal2()

```
const std::string & minerule::SimplePredicate::getVal2 ( ) const  [inline]
```

Definition at line 85 of file Predicate.hpp.

```
00085                                    {
00086       return val2;
00087     }
```

6.140.2.5 getVarId()

```
size_t minerule::SimplePredicate::getVarId ( ) const  [inline]
```

Definition at line 108 of file Predicate.hpp.

```
00108                              {
00109       return varId;
00110     }
```

6.140.2.6 newSimplePredicate() [1/2]

```
SimplePredicate & minerule::SimplePredicate::newSimplePredicate (
            const simple_pred * pred )  [static]
```

Definition at line 32 of file Predicate.cpp.

```
00032                                                                {
00033     SimplePredicate* newpred = new SimplePredicate(pred);
00034
00035     PredicatePool::iterator it=predicatePool.find(newpred);
00036     if( it==predicatePool.end() ) {
00037       predicatePool.insert(newpred);
00038       return *newpred;
00039     } else {
00040       delete newpred;
00041       return **it;
00042     }
00043   }
```

6.140.2.7 newSimplePredicate() [2/2]

```
SimplePredicate & minerule::SimplePredicate::newSimplePredicate (
            const std::string & val1,
            const std::string & op,
            const std::string & val2 )  [static]
```

Definition at line 46 of file Predicate.cpp.

```
00048                                                                {
00049     SimplePredicate* newpred = new SimplePredicate(val1,op,val2);
00050     PredicatePool::iterator it=predicatePool.find(newpred);
00051     if( it==predicatePool.end() ) {
00052       predicatePool.insert(newpred);
00053       return *newpred;
00054     } else {
00055       delete newpred;
00056       return **it;
00057     }
00058   }
```

6.140.2.8 operator<()

```
bool minerule::SimplePredicate::operator< (
            const SimplePredicate & rhs ) const  [inline]
```

Definition at line 100 of file Predicate.hpp.

```
00100                                                 {
00101       return val1+op+val2 < rhs.val1+rhs.op+rhs.val2;
00102     }
```

6.140.2.9 operator==()

```
bool minerule::SimplePredicate::operator== (
            const SimplePredicate & rhs ) const  [inline]
```

Definition at line 93 of file Predicate.hpp.

```
00093                                                 {
00094       return
00095         val1==rhs.val1 &&
00096         val2==rhs.val2 &&
00097         op==rhs.op;
00098     }
```

6.140.2.10 setVarId()

```
void minerule::SimplePredicate::setVarId (
            size_t id )  [inline]
```

Definition at line 104 of file Predicate.hpp.

```
00104                                {
00105       varId = id;
00106     }
```

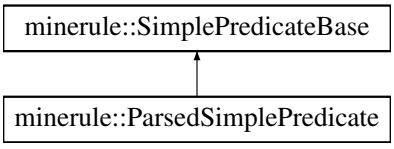
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp
- /Users/esposito/Software/minerule/src/PredicateUtils/Predicate.cpp

6.141 minerule::SimplePredicateBase Class Reference

```
#include <PredicateBase.hpp>
```

Inheritance diagram for minerule::SimplePredicateBase:



Public Member Functions

- SimplePredicateBase (const std::string &v1, const std::string &o, const std::string &v2)
- SimplePredicateBase ()
- SimplePredicateBase (const SimplePredicateBase &s)
- virtual SimplePredicateBase * cloneInstance () const
- virtual SimplePredicateBase * newInstance () const
- virtual ~SimplePredicateBase ()
- virtual PredicateBase * newPredicate () const
- virtual PredConjunctionBase * newPredConjunction () const
- const std::string & getVal1 () const
- const std::string & getVal2 () const
- const std::string & getOp () const
- virtual bool operator== (const SimplePredicateBase &rhs) const
- virtual bool operator!= (const SimplePredicateBase &rhs) const
- virtual bool operator< (const SimplePredicateBase &rhs) const
- virtual PredicateBase & operator! () const

6.141.1 Detailed Description

This class represent a simple Predicate in form of term comparison term

Definition at line 28 of file PredicateBase.hpp.

6.141.2 Constructor & Destructor Documentation

6.141.2.1 SimplePredicateBase() [1/3]

```
minerule::SimplePredicateBase::SimplePredicateBase (
            const std::string & v1,
            const std::string & o,
            const std::string & v2 )  [inline]
```

the Constructor of class SimplePredicate @paramstd::string v1 the first term @paramstd::string op the comparison term @paramstd::string v2 the second term

Definition at line 40 of file PredicateBase.hpp.

```
00042                                                :
00043       val1(v1), val2(v2),  op(o) {  }
```

6.141.2.2 SimplePredicateBase() [2/3]

```
minerule::SimplePredicateBase::SimplePredicateBase ( )  [inline]
```

The empty constructor

Definition at line 48 of file PredicateBase.hpp.

```
00048 {}
```

6.141.2.3 SimplePredicateBase() [3/3]

```
minerule::SimplePredicateBase::SimplePredicateBase (
            const SimplePredicateBase & s )  [inline]
```

The copy constructor

Parameters

SimplePredicateBase	s another object of this class that i want copy

Definition at line 54 of file PredicateBase.hpp.

```
00054                                                   :
00055       val1(s.val1), val2(s.val2), op(s.op) {}
```

6.141.2.4 ~SimplePredicateBase()

```
virtual minerule::SimplePredicateBase::~SimplePredicateBase ( )  [inline], [virtual]
```

This is the destructor of class

Definition at line 72 of file PredicateBase.hpp.

```
00072 {}
```

6.141.3 Member Function Documentation

6.141.3.1 cloneInstance()

```
virtual SimplePredicateBase * minerule::SimplePredicateBase::cloneInstance ( ) const  [inline],
[virtual]
```

- this method is very important because permit to clone the exact objcet in the hierachy of class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

  Returns

  SimplePredicateBase the object of super-class that incapsule a new object of this class

Reimplemented in minerule::ParsedSimplePredicate.

Definition at line 60 of file PredicateBase.hpp.

```
00060                                                   {
00061       return new SimplePredicateBase(*this);
00062     }
```

6.141.3.2 getOp()

```
const std::string & minerule::SimplePredicateBase::getOp ( ) const  [inline]
```

This method return a pointer to comparison operator of Simple Predicate

Returns

std::string& op the comparison operator

Definition at line 103 of file PredicateBase.hpp.

```
00103                                       {
00104       return op;
00105     }
```

6.141.3.3 getVal1()

```
const std::string & minerule::SimplePredicateBase::getVal1 ( ) const  [inline]
```

This method return a pointer to first term of Simple Predicate

Returns

std::string& val1 the firt term

Definition at line 87 of file PredicateBase.hpp.

```
00087                                       {
00088       return val1;
00089     }
```

6.141.3.4 getVal2()

```
const std::string & minerule::SimplePredicateBase::getVal2 ( ) const  [inline]
```

This method return a pointer to second term of Simple Predicate

Returns

std::string& val2 the second term

Definition at line 95 of file PredicateBase.hpp.

```
00095                                       {
00096       return val2;
00097     }
```

6.141.3.5 newInstance()

```
virtual SimplePredicateBase * minerule::SimplePredicateBase::newInstance ( ) const  [inline],
[virtual]
```

- this method is very important because permit create a new instance of the exact objcet in the hierachy of the class * because the RTTI mechanism use the right method with a specific type of object. if you want create a * subclass of this class you must overriding this method for your class *

  Returns

  SimplePredicateBase the object of super-class that incapsule a new object of this class

Reimplemented in minerule::ParsedSimplePredicate.

Definition at line 65 of file PredicateBase.hpp.

```
00065                                                {
00066       return new SimplePredicateBase();
00067     }
```

6.141.3.6 newPredConjunction()

```
PredConjunctionBase * minerule::SimplePredicateBase::newPredConjunction ( ) const  [virtual]
```

Reimplemented in minerule::ParsedSimplePredicate.

Definition at line 14 of file PredicateBase.cpp.

```
00014                                                       {
00015     return new PredConjunctionBase();
00016   }
```

6.141.3.7 newPredicate()

```
PredicateBase * minerule::SimplePredicateBase::newPredicate ( ) const  [virtual]
```

Reimplemented in minerule::ParsedSimplePredicate.

Definition at line 10 of file PredicateBase.cpp.

```
00010                                                 {
00011     return new PredicateBase();
00012   }
```

6.141.3.8 operator"!()

```
PredicateBase & minerule::SimplePredicateBase::operator!  ( ) const  [virtual]
```

Define a method that return a new PredicateBase& that contains the negation of this Simple Predicate

Returns

PredicateBase& that contains the negation of this Simple Predicate

Definition at line 37 of file PredicateBase.cpp.

```
00037                                       {
00038    std::string nop;
00039     if (op==">") { nop="<="; }
00040     else if (op==">=") { nop="<"; }
00041     else if (op=="<") { nop=">="; }
00042     else if (op=="<=") { nop=">"; }
00043     else if (op=="=") { nop="<>"; }
00044     else if (op=="<>") { nop="="; }
00045     SimplePredicateBase* sp= this->cloneInstance();
00046     PredConjunctionBase* pc= this->newPredConjunction();
00047     PredicateBase*       pb= this->newPredicate();
00048
00049     sp->op = nop;
00050     pc->push_back(sp);
00051     pb->push_back(pc);
00052
00053     return *pb;
00054   }
```

6.141.3.9 operator"!=()

```
virtual bool minerule::SimplePredicateBase::operator!= (
            const SimplePredicateBase & rhs ) const  [inline], [virtual]
```

Redefine the operator !=

Returns

true if at least one of the part of Simple Predicate ar different

Definition at line 122 of file PredicateBase.hpp.

```
00122                                                                  {
00123       return
00124         val1!=rhs.val1 ||
00125         val2!=rhs.val2 ||
00126         op!=rhs.op;
00127     }
```

6.141.3.10 operator<()

```
virtual bool minerule::SimplePredicateBase::operator< (
            const SimplePredicateBase & rhs ) const  [inline], [virtual]
```

Redefine the < operator

Returns

true if the first term is lesser than second

Definition at line 133 of file PredicateBase.hpp.

```
00133                                                                  {
00134       return val1+op+val2 < rhs.val1+rhs.op+rhs.val2;
00135     }
```

6.141.3.11 operator==()

```
virtual bool minerule::SimplePredicateBase::operator== (
            const SimplePredicateBase & rhs ) const  [inline], [virtual]
```

Redefine the operator ==

Returns

true if and only if all the part of Simple Predicate are equals

Definition at line 111 of file PredicateBase.hpp.

```
00111                                                                    {
00112       return
00113         val1==rhs.val1 &&
00114         val2==rhs.val2 &&
00115         op==rhs.op;
00116     }
```

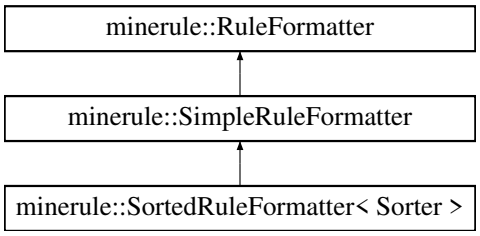
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/PredicateBase.hpp
- /Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp

6.142 minerule::SimpleRuleFormatter Class Reference

```
#include <RuleFormatter.hpp>
```

Inheritance diagram for minerule::SimpleRuleFormatter:



Public Member Functions

- SimpleRuleFormatter ()
- virtual ~SimpleRuleFormatter ()
- virtual std::string formatRule (const Rule &, bool=true)
- virtual void printRule (const Rule &)
- virtual void postExec ()
- void setSuppressLog (bool newVal)
- bool suppressLog () const
- void setFieldWidths (const FieldWidths &f)
- FieldWidths & fieldWidths ()
- const FieldWidths & fieldWidths () const

Static Protected Member Functions

- static std::string quote (const std::string &elem)
- static std::string quoteElems (const ItemSet &elems)

Protected Attributes

- bool _suppressLog
- std::string _bhSep
- FieldWidths _fieldWidths

6.142.1 Detailed Description

Definition at line 71 of file RuleFormatter.hpp.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 SimpleRuleFormatter()

```
minerule::SimpleRuleFormatter::SimpleRuleFormatter ( )  [inline]
```

Definition at line 73 of file RuleFormatter.hpp.

```
00073 : RuleFormatter() {};
```

6.142.2.2 ~SimpleRuleFormatter()

```
virtual minerule::SimpleRuleFormatter::~SimpleRuleFormatter ( )  [inline], [virtual]
```

Definition at line 75 of file RuleFormatter.hpp.

```
00075 {};
```

6.142.3 Member Function Documentation

6.142.3.1 fieldWidths() [1/2]

```
FieldWidths & minerule::RuleFormatter::fieldWidths ( )  [inline], [inherited]
```

Definition at line 63 of file RuleFormatter.hpp.

```
00063 { return _fieldWidths; }
```

6.142.3.2 fieldWidths() [2/2]

```
const FieldWidths & minerule::RuleFormatter::fieldWidths ( ) const  [inline], [inherited]
```

Definition at line 64 of file RuleFormatter.hpp.

```
00064 { return _fieldWidths; }
```

6.142.3.3 formatRule()

```
std::string minerule::SimpleRuleFormatter::formatRule (
            const Rule & rule,
            bool includeSuppConf = true )  [virtual]
```

Implements minerule::RuleFormatter.

Definition at line 45 of file RuleFormatter.cpp.

```
00045                                                                                   {
00046                 std::stringstream out;
00047
00048                 out     << std::setw(_fieldWidths.body) << quoteElems(rule.getBody())
00049                         << _bhSep
00050                         << std::left << std::setw(_fieldWidths.head) << quoteElems(rule.getHead());
00051
00052                 if(includeSuppConf) {
00053                         out << std::right
00054                                 << " "
00055                                 << std::setw(_fieldWidths.supp) << rule.getSupport()
00056                                 << " "
00057                                 << std::setw(_fieldWidths.conf) << rule.getConfidence();
00058                 }
00059
00060                 return out.str();
00061         }
```

6.142.3.4 postExec()

```
virtual void minerule::SimpleRuleFormatter::postExec ( )  [inline], [virtual]
```

Implements minerule::RuleFormatter.

Reimplemented in minerule::SortedRuleFormatter< Sorter >.

Definition at line 81 of file RuleFormatter.hpp.

```
00081 {};
```

6.142.3.5 printRule()

```
void minerule::SimpleRuleFormatter::printRule (
            const Rule & rule )  [virtual]
```

Implements minerule::RuleFormatter.

Reimplemented in minerule::SortedRuleFormatter< Sorter >.

Definition at line 65 of file RuleFormatter.cpp.

```
00065                                                   {
00066                 if( suppressLog() ) {
00067                         std::cout << formatRule(rule)  << std::endl ;
00068                 } else {
00069                         MRLog() << formatRule(rule)  << std::endl;
00070                 }
00071         }
```

6.142.3.6 quote()

```
std::string minerule::RuleFormatter::quote (
            const std::string & elem )  [static], [protected], [inherited]
```

Definition at line 22 of file RuleFormatter.cpp.

```
00022                                                       {
00023                 return "["+elem+"]";
00024         }
```

6.142.3.7 quoteElems()

```
std::string minerule::RuleFormatter::quoteElems (
            const ItemSet & elems )  [static], [protected], [inherited]
```

Definition at line 26 of file RuleFormatter.cpp.

```
00026                                                           {
00027                 std::string result;
00028
00029                 ItemSet::const_iterator it=elems.begin();
00030
00031                 if(it!=elems.end()) {
00032                         result = quote(it->asString());
00033                         it++;
00034                 }
00035
00036                 for(; it!=elems.end(); it++) {
00037                         result +=  ", " + quote(it->asString());
00038                 }
00039
00040                 return result;
00041         }
```

6.142.3.8 setFieldWidths()

```
void minerule::RuleFormatter::setFieldWidths (
            const FieldWidths & f )  [inline], [inherited]
```

Definition at line 62 of file RuleFormatter.hpp.

```
00062 { _fieldWidths = f; }
```

6.142.3.9 setSuppressLog()

```
void minerule::RuleFormatter::setSuppressLog (
            bool newVal )  [inline], [inherited]
```

Definition at line 59 of file RuleFormatter.hpp.

```
00059 {  _suppressLog = newVal; }
```

6.142.3.10 suppressLog()

```
bool minerule::RuleFormatter::suppressLog ( ) const  [inline], [inherited]
```

Definition at line 60 of file RuleFormatter.hpp.

```
00060 { return _suppressLog; }
```

6.142.4 Field Documentation

6.142.4.1 _bhSep

```
std::string minerule::RuleFormatter::_bhSep  [protected], [inherited]
```

Definition at line 44 of file RuleFormatter.hpp.

6.142.4.2 _fieldWidths

```
FieldWidths minerule::RuleFormatter::_fieldWidths  [protected], [inherited]
```

Definition at line 45 of file RuleFormatter.hpp.

6.142.4.3 _suppressLog

```
bool minerule::RuleFormatter::_suppressLog  [protected], [inherited]
```

Definition at line 42 of file RuleFormatter.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp
- /Users/esposito/Software/minerule/src/Result/RuleFormatter.cpp

6.143 minerule::QueryResult::SortBodyHead Class Reference

```
#include <QueryResult-header.hpp>
```

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.143.1 Detailed Description

Definition at line 124 of file QueryResult-header.hpp.

6.143.2 Member Function Documentation

6.143.2.1 operator()()

```
bool minerule::QueryResult::SortBodyHead::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 126 of file QueryResult-header.hpp.

```
00126                                                                   {
00127                                 if( r1.getBody() > r2.getBody() )
00128                                         return true;
00129
00130                                 if( r1.getBody() == r2.getBody() ) {
00131                                         if(r1.getHead() > r2.getHead() )
00132                                                 return true;
00133                                 }
00134
00135                                 return false;
00136                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.144 minerule::QueryResult::SortBodyHeadSuppConf Class Reference

```
#include <QueryResult-header.hpp>
```

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.144.1 Detailed Description

Definition at line 99 of file QueryResult-header.hpp.

6.144.2 Member Function Documentation

6.144.2.1 operator()()

```
bool minerule::QueryResult::SortBodyHeadSuppConf::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 101 of file QueryResult-header.hpp.

```
00101                                                                                {
00102                                 if( r1.getBody() > r2.getBody() )
00103                                         return true;
00104
00105                                 if( r1.getBody() == r2.getBody() ) {
00106                                         if(r1.getHead() > r2.getHead() )
00107                                                 return true;
00108
00109                                         if(r1.getHead() == r2.getHead()) {
00110                                                 return r1.getSupport() > r2.getSupport() ||
00111                                                         (r1.getSupport()==r2.getSupport() &&
      r1.getConfidence() > r2.getConfidence());
00112                                         }
00113                                 }
00114
00115                                 return false;
00116                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.145 minerule::QueryResult::SortConfBodyHeadSupp Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.145.1 Detailed Description

Definition at line 237 of file QueryResult-header.hpp.

6.145.2 Member Function Documentation

6.145.2.1 operator()()

```
bool minerule::QueryResult::SortConfBodyHeadSupp::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 239 of file QueryResult-header.hpp.

```
00239                                                                    {
00240                                 if(r1.getConfidence() > r2.getConfidence())
00241                                         return true;
00242                                 if(r1.getConfidence() < r2.getConfidence())
00243                                         return false;
00244
00245                                 if(r1.getBody() > r2.getBody())
00246                                         return true;
00247                                 if(r1.getBody() < r2.getBody())
00248                                         return false;
00249
00250                                 if(r1.getHead() > r2.getHead())
00251                                         return true;
00252                                 if(r1.getHead() < r2.getHead())
00253                                         return false;
00254
00255                                 if(r1.getSupport() > r2.getSupport())
00256                                         return true;
00257                                 if(r1.getSupport() < r2.getSupport())
00258                                         return false;
00259
00260                                 return false;
00261                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.146 minerule::QueryResult::SortConfBodySuppHead Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.146.1 Detailed Description

Definition at line 269 of file QueryResult-header.hpp.

6.146.2 Member Function Documentation

6.146.2.1 operator()()

```
bool minerule::QueryResult::SortConfBodySuppHead::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 271 of file QueryResult-header.hpp.

```
00271                                                                                {
00272                                 if(r1.getConfidence() > r2.getConfidence())
00273                                         return true;
00274                                 if(r1.getConfidence() < r2.getConfidence())
00275                                         return false;
00276
00277                                 if(r1.getBody() > r2.getBody())
00278                                         return true;
00279                                 if(r1.getBody() < r2.getBody())
00280                                         return false;
00281
00282                                 if(r1.getSupport() > r2.getSupport())
00283                                         return true;
00284                                 if(r1.getSupport() < r2.getSupport())
00285                                         return false;
00286
00287                                 if(r1.getHead() > r2.getHead())
00288                                         return true;
00289                                 if(r1.getHead() < r2.getHead())
00290                                         return false;
00291
00292                         // here all fields are equals
00293                                 return false;
00294                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.147 minerule::QueryResult::SortConfSuppBodyHead Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.147.1 Detailed Description

Definition at line 204 of file QueryResult-header.hpp.

6.147.2 Member Function Documentation

6.147.2.1 operator()()

```
bool minerule::QueryResult::SortConfSuppBodyHead::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 206 of file QueryResult-header.hpp.

```
00206                                                                                {
00207                                 if(r1.getConfidence() > r2.getConfidence())
00208                                         return true;
00209                                 if(r1.getConfidence() < r2.getConfidence())
00210                                         return false;
00211
00212                                 if(r1.getSupport() > r2.getSupport())
00213                                         return true;
00214                                 if(r1.getSupport() < r2.getSupport())
00215                                         return false;
00216
00217                                 if(r1.getBody() > r2.getBody())
00218                                         return true;
00219                                 if(r1.getBody() < r2.getBody())
00220                                         return false;
00221
00222                                 if(r1.getHead() > r2.getHead())
00223                                         return true;
00224                                 if(r1.getHead() < r2.getHead())
00225                                         return false;
00226
00227                         // here all fields are equals
00228                                 return false;
00229                         }
```

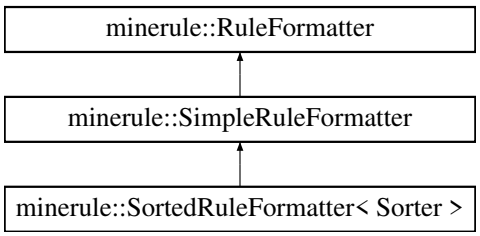
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.148 minerule::SortedRuleFormatter< Sorter > Class Template Reference

```
#include <RuleFormatter.hpp>
```

Inheritance diagram for minerule::SortedRuleFormatter< Sorter >:



Public Member Functions

- SortedRuleFormatter ()
- virtual ~SortedRuleFormatter ()
- virtual void printRule (const Rule &)
- virtual void postExec ()
- virtual std::string formatRule (const Rule &, bool=true)
- void setSuppressLog (bool newVal)
- bool suppressLog () const
- void setFieldWidths (const FieldWidths &f)
- FieldWidths & fieldWidths ()
- const FieldWidths & fieldWidths () const

Static Protected Member Functions

- static std::string quote (const std::string &elem)
- static std::string quoteElems (const ItemSet &elems)

Protected Attributes

- bool _suppressLog
- std::string _bhSep
- FieldWidths _fieldWidths

6.148.1 Detailed Description

template<class Sorter>
class minerule::SortedRuleFormatter< Sorter >

Definition at line 86 of file RuleFormatter.hpp.

6.148.2 Constructor & Destructor Documentation

6.148.2.1 SortedRuleFormatter()

```
template<class Sorter >
minerule::SortedRuleFormatter< Sorter >::SortedRuleFormatter ( )  [inline]
```

Definition at line 90 of file RuleFormatter.hpp.

```
00090 : SimpleRuleFormatter() {};
```

6.148.2.2 ~SortedRuleFormatter()

```
template<class Sorter >
virtual minerule::SortedRuleFormatter< Sorter >::~SortedRuleFormatter ( )  [inline], [virtual]
```

Definition at line 91 of file RuleFormatter.hpp.

```
00091 {};
```

6.148.3 Member Function Documentation

6.148.3.1 fieldWidths() [1/2]

```
FieldWidths & minerule::RuleFormatter::fieldWidths ( )  [inline], [inherited]
```

Definition at line 63 of file RuleFormatter.hpp.

```
00063 { return _fieldWidths; }
```

6.148.3.2 fieldWidths() [2/2]

```
const FieldWidths & minerule::RuleFormatter::fieldWidths ( ) const  [inline], [inherited]
```

Definition at line 64 of file RuleFormatter.hpp.

```
00064 { return _fieldWidths; }
```

6.148.3.3 formatRule()

```
std::string minerule::SimpleRuleFormatter::formatRule (
            const Rule & rule,
            bool includeSuppConf = true )  [virtual], [inherited]
```

Implements minerule::RuleFormatter.

Definition at line 45 of file RuleFormatter.cpp.

```
00045                                                                                   {
00046                 std::stringstream out;
00047
00048                 out     « std::setw(_fieldWidths.body) « quoteElems(rule.getBody())
00049                         « _bhSep
00050                         « std::left « std::setw(_fieldWidths.head) « quoteElems(rule.getHead());
00051
00052                 if(includeSuppConf) {
00053                         out « std::right
00054                                 « " "
00055                                 « std::setw(_fieldWidths.supp) « rule.getSupport()
00056                                 « " "
00057                                 « std::setw(_fieldWidths.conf) « rule.getConfidence();
00058                 }
00059
00060                 return out.str();
00061         }
```

6.148.3.4 postExec()

```
template<class Sorter >
void minerule::SortedRuleFormatter< Sorter >::postExec  [virtual]
```

Reimplemented from minerule::SimpleRuleFormatter.

Definition at line 104 of file RuleFormatter.hpp.

```
00104                                                   {
00105                 typename SortedContainer::const_iterator it;
00106                 for(it=sortedRules.begin(); it!=sortedRules.end(); it++) {
00107                         SimpleRuleFormatter::printRule(*it);
00108                 }
00109         }
```

6.148.3.5 printRule()

```
template<class Sorter >
void minerule::SortedRuleFormatter< Sorter >::printRule (
            const Rule & r )  [virtual]
```

Reimplemented from minerule::SimpleRuleFormatter.

Definition at line 99 of file RuleFormatter.hpp.

```
00099                                                       {
00100                 sortedRules.insert(r);
00101         }
```

6.148.3.6 quote()

```
std::string minerule::RuleFormatter::quote (
            const std::string & elem )  [static], [protected], [inherited]
```

Definition at line 22 of file RuleFormatter.cpp.

```
00022                                                   {
00023                 return "["+elem+"]";
00024         }
```

6.148.3.7 quoteElems()

```
std::string minerule::RuleFormatter::quoteElems (
            const ItemSet & elems )  [static], [protected], [inherited]
```

Definition at line 26 of file RuleFormatter.cpp.

```
00026                                                       {
00027                 std::string result;
00028
00029                 ItemSet::const_iterator it=elems.begin();
00030
00031                 if(it!=elems.end()) {
00032                         result = quote(it->asString());
00033                         it++;
00034                 }
00035
00036                 for(; it!=elems.end(); it++) {
00037                         result +=  ", " + quote(it->asString());
00038                 }
00039
00040                 return result;
00041         }
```

6.148.3.8 setFieldWidths()

```
void minerule::RuleFormatter::setFieldWidths (
            const FieldWidths & f )  [inline], [inherited]
```

Definition at line 62 of file RuleFormatter.hpp.

```
00062 { _fieldWidths = f; }
```

6.148.3.9 setSuppressLog()

```
void minerule::RuleFormatter::setSuppressLog (
            bool newVal )  [inline], [inherited]
```

Definition at line 59 of file RuleFormatter.hpp.
```
00059 {  _suppressLog = newVal; }
```

6.148.3.10 suppressLog()

```
bool minerule::RuleFormatter::suppressLog ( ) const  [inline], [inherited]
```

Definition at line 60 of file RuleFormatter.hpp.
```
00060 { return _suppressLog; }
```

6.148.4 Field Documentation

6.148.4.1 _bhSep

```
std::string minerule::RuleFormatter::_bhSep  [protected], [inherited]
```

Definition at line 44 of file RuleFormatter.hpp.

6.148.4.2 _fieldWidths

```
FieldWidths minerule::RuleFormatter::_fieldWidths  [protected], [inherited]
```

Definition at line 45 of file RuleFormatter.hpp.

6.148.4.3 _suppressLog

```
bool minerule::RuleFormatter::_suppressLog  [protected], [inherited]
```

Definition at line 42 of file RuleFormatter.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp

6.149 minerule::QueryResult::SortHeadBodySuppConf Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.149.1 Detailed Description

Definition at line 144 of file QueryResult-header.hpp.

6.149.2 Member Function Documentation

6.149.2.1 operator()()

```
bool minerule::QueryResult::SortHeadBodySuppConf::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 146 of file QueryResult-header.hpp.

```
00146                                                                   {
00147                                 if( r1.getHead() > r2.getHead() )
00148                                         return true;
00149
00150                                 if( r1.getHead() == r2.getHead() ) {
00151                                         if(r1.getBody() > r2.getBody() )
00152                                                 return true;
00153
00154                                         if(r1.getBody() == r2.getBody()) {
00155                                                 return r1.getSupport() > r2.getSupport() ||
00156                                                         (r1.getSupport()==r2.getSupport() &&
       r1.getConfidence() > r2.getConfidence());
00157                                         }
00158                                 }
00159
00160                                 return false;
00161                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.150 minerule::QueryResult::SortSuppConfBodyHead Class Reference

`#include <QueryResult-header.hpp>`

Public Member Functions

- bool operator() (const Rule &r1, const Rule &r2) const

6.150.1 Detailed Description

Definition at line 169 of file QueryResult-header.hpp.

6.150.2 Member Function Documentation

6.150.2.1 operator()()

```
bool minerule::QueryResult::SortSuppConfBodyHead::operator() (
            const Rule & r1,
            const Rule & r2 ) const  [inline]
```

Definition at line 171 of file QueryResult-header.hpp.

```
00171                                                                                {
00172                                 if(r1.getSupport() > r2.getSupport())
00173                                         return true;
00174                                 if(r1.getSupport() < r2.getSupport())
00175                                         return false;
00176
00177         // here r1.getSupport() == r2.getSupport()
00178                                 if(r1.getConfidence() > r2.getConfidence())
00179                                         return true;
00180                                 if(r1.getConfidence() < r2.getConfidence())
00181                                         return false;
00182
00183         // here r1.sup=r2.sup && r1.conf=r2.conf
00184                                 if(r1.getBody() > r2.getBody())
00185                                         return true;
00186                                 if(r1.getBody() < r2.getBody())
00187                                         return false;
00188
00189                                 if(r1.getHead() > r2.getHead())
00190                                         return true;
00191                                 if(r1.getHead() < r2.getHead())
00192                                         return false;
00193
00194         // here all fields are equals
00195                                 return false;
00196                         }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Result/QueryResult-header.hpp

6.151 minerule::SourceRow Class Reference

```
#include <SourceRow.hpp>
```

Public Member Functions

- SourceRow ()
- SourceRow (mrdb::ResultSet *resultSet, const SourceRowColumnIds &srd)
- SourceRow (const SourceRow &item)
- virtual ~SourceRow ()
- void init (mrdb::ResultSet *resultSet, const SourceRowColumnIds &srd)
- const SourceRowElement & getGroup () const
- const SourceRowElement & getClusterBody () const
- const SourceRowElement & getBody () const
- const SourceRowElement & getClusterHead () const
- const SourceRowElement & getHead () const

Friends

- std::ostream & operator<< (std::ostream &os, const SourceRow &item)

6.151.1 Detailed Description

This class models the concept of SourceRow. It is useful in order to abstract away details about the structure of the source table.

Definition at line 37 of file SourceRow.hpp.

6.151.2 Constructor & Destructor Documentation

6.151.2.1 SourceRow() [1/3]

```
minerule::SourceRow::SourceRow ( )  [inline]
```

Definition at line 50 of file SourceRow.hpp.

```
00050 : group(NULL), clusterBody(NULL), body(NULL), clusterHead(NULL), head(NULL) { }
```

6.151.2.2 SourceRow() [2/3]

```
minerule::SourceRow::SourceRow (
            mrdb::ResultSet * resultSet,
            const SourceRowColumnIds & srd )
```

Definition at line 27 of file SourceRow.cpp.

```
00028                                                                    {
00029     mrdb::ResultSetMetaData* rsmd = resultSet->getMetaData();
00030
00031     group =    SourceRowElement::createElement(rsmd,resultSet,srd.groupElems);
00032     clusterBody =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterBodyElems);
00033     body =         SourceRowElement::createElement(rsmd,resultSet,srd.bodyElems);
00034     clusterHead =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterHeadElems);
00035     head =         SourceRowElement::createElement(rsmd,resultSet,srd.headElems);
00036   }
```

6.151.2.3 SourceRow() [3/3]

```
minerule::SourceRow::SourceRow (
            const SourceRow & item )
```

Definition at line 38 of file SourceRow.cpp.

```
00038                                          {
00039 #define CREATE(a) (((rhs.a)!=NULL)?(a)=rhs.a->copy():(a)=NULL)
00040
00041     CREATE(group);
00042     CREATE(clusterBody);
00043     CREATE(body);
00044     CREATE(clusterHead);
00045     CREATE(head);
00046
00047 #undef CREATE
00048   }
```

6.151.2.4 ~SourceRow()

```
minerule::SourceRow::~SourceRow ( )  [virtual]
```

Definition at line 52 of file SourceRow.cpp.

```
00052                       {
00053 #define DESTROY(a) if((a)!=NULL) delete a;
00054
00055   DESTROY(group);
00056   DESTROY(clusterBody);
00057   DESTROY(body);
00058   DESTROY(clusterHead);
00059   DESTROY(head);
00060
00061 #undef DESTROY
00062 }
```

6.151.3 Member Function Documentation

6.151.3.1 getBody()

```
const SourceRowElement & minerule::SourceRow::getBody ( ) const  [inline]
```

Definition at line 76 of file SourceRow.hpp.

```
00076                                   {
00077                         ELEM_OR_EMPTY(body);
00078                 }
```

6.151.3.2 getClusterBody()

```
const SourceRowElement & minerule::SourceRow::getClusterBody ( ) const  [inline]
```

Definition at line 71 of file SourceRow.hpp.

```
00071                                       {
00072                         ELEM_OR_EMPTY(clusterBody);
00073                 }
```

6.151.3.3 getClusterHead()

```
const SourceRowElement & minerule::SourceRow::getClusterHead ( ) const  [inline]
```

Definition at line 81 of file SourceRow.hpp.

```
00081                                       {
00082                         ELEM_OR_EMPTY(clusterHead);
00083                 }
```

6.151.3.4 getGroup()

```
const SourceRowElement & minerule::SourceRow::getGroup ( ) const  [inline]
```

Definition at line 66 of file SourceRow.hpp.

```
00066                                {
00067                         ELEM_OR_EMPTY(group)
00068                 }
```

6.151.3.5 getHead()

```
const SourceRowElement & minerule::SourceRow::getHead ( ) const  [inline]
```

Definition at line 86 of file SourceRow.hpp.

```
00086                                {
00087                         ELEM_OR_EMPTY(head);
00088                 }
```

6.151.3.6 init()

```
void minerule::SourceRow::init (
            mrdb::ResultSet * resultSet,
            const SourceRowColumnIds & srd )
```

Definition at line 65 of file SourceRow.cpp.

```
00066                                                        {
00067
00068 #define DESTROY(a) if((a)!=NULL) delete a;
00069
00070   DESTROY(group);
00071   DESTROY(clusterBody);
00072   DESTROY(body);
00073   DESTROY(clusterHead);
00074   DESTROY(head);
00075
00076   mrdb::ResultSetMetaData* rsmd = resultSet->getMetaData();
00077
00078   group =    SourceRowElement::createElement(rsmd,resultSet,srd.groupElems);
00079   clusterBody =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterBodyElems);
00080   body =         SourceRowElement::createElement(rsmd,resultSet,srd.bodyElems);
00081   clusterHead =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterHeadElems);
00082   head =         SourceRowElement::createElement(rsmd,resultSet,srd.headElems);
00083
00084 #undef DESTROY
00085 }
```

6.151.4 Friends And Related Function Documentation

6.151.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const SourceRow & item )  [friend]
```

Definition at line 93 of file SourceRow.cpp.

```
00094                                                {
00095   os « "group:" « sr.getGroup() « std::endl;
00096   os « " clusterBody:" « sr.getClusterBody() « std::endl;
00097   os « " body:" « sr.getBody() « std::endl;
00098   os « " clusterHead:" « sr.getClusterHead() « std::endl;
00099   os « " head:" « sr.getHead() « std::endl;
00100
00101   return os;
00102 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRow.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRow.cpp

6.152 minerule::SourceRowAttrCollectionDescriptor Class Reference

```
#include <SourceRowMetaInfo.hpp>
```

Public Member Functions

- SourceRowAttrCollectionDescriptor ()
- SourceRowAttrCollectionDescriptor (mrdb::ResultSet *rs, const std::vector< int > &collectionElems)
- void init (mrdb::ResultSet *rs, const std::vector< int > &collectionElems)
- const std::string & getSQLDataDefinition () const
- const std::string & getSQLColumnNames () const
- unsigned int getColumnsCount () const
- std::string questionMarks () const

6.152.1 Detailed Description

Definition at line 31 of file SourceRowMetaInfo.hpp.

6.152.2 Constructor & Destructor Documentation

6.152.2.1 SourceRowAttrCollectionDescriptor() [1/2]

```
minerule::SourceRowAttrCollectionDescriptor::SourceRowAttrCollectionDescriptor ( )  [inline]
```

Definition at line 43 of file SourceRowMetaInfo.hpp.

```
00043 {}
```

6.152.2.2 SourceRowAttrCollectionDescriptor() [2/2]

```
SourceRowAttrCollectionDescriptor::SourceRowAttrCollectionDescriptor (
            mrdb::ResultSet * rs,
            const std::vector< int > & collectionElems )
```

Definition at line 97 of file SourceRowMetaInfo.cpp.

```
00097                                                                   {
00098         init(rs, collectionElems);
00099 }
```

6.152.3 Member Function Documentation

6.152.3.1 getColumnsCount()

```
unsigned int minerule::SourceRowAttrCollectionDescriptor::getColumnsCount ( ) const  [inline]
```

Definition at line 50 of file SourceRowMetaInfo.hpp.

```
00050 { return columnsCount; }
```

6.152.3.2 getSQLColumnNames()

```
const std::string & SourceRowAttrCollectionDescriptor::getSQLColumnNames ( ) const
```

Definition at line 107 of file SourceRowMetaInfo.cpp.

```
00107                                                                   {
00108   return columnNames;
00109 }
```

6.152.3.3 getSQLDataDefinition()

```
const std::string & SourceRowAttrCollectionDescriptor::getSQLDataDefinition ( ) const
```

Definition at line 102 of file SourceRowMetaInfo.cpp.

```
00102                                                                   {
00103   return dataDefinition;
00104 }
```

6.152.3.4 init()

```
void SourceRowAttrCollectionDescriptor::init (
            mrdb::ResultSet * rs,
            const std::vector< int > & collectionElems )
```

Definition at line 89 of file SourceRowMetaInfo.cpp.

```
00089
      {
00090     setColumnNames(rs,collectionElems);
00091     setDataDefinition(rs,collectionElems);
00092     columnsCount = collectionElems.size();
00093 }
```

6.152.3.5 questionMarks()

```
std::string SourceRowAttrCollectionDescriptor::questionMarks ( ) const
```

Definition at line 62 of file SourceRowMetaInfo.cpp.

```
00062                                                                {
00063         std::string result;
00064         for( size_t i=0; i<columnsCount; ++i) {
00065                 if( i!=0 )
00066                         result += ",?";
00067                 else
00068                         result += "?";
00069         }
00070
00071         return result;
00072 }
```

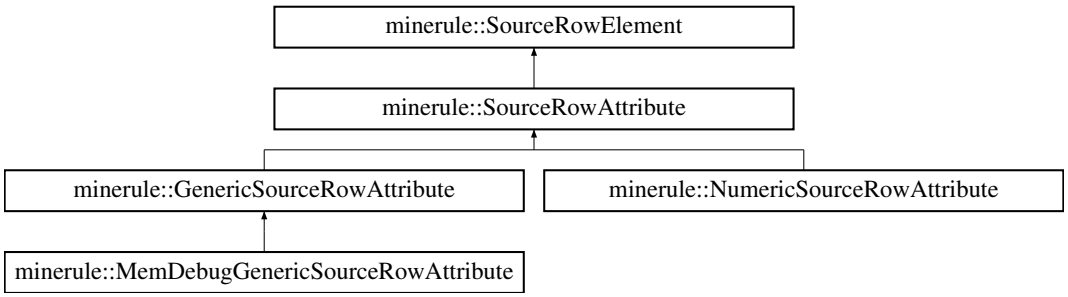
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowMetaInfo.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowMetaInfo.cpp

6.153 minerule::SourceRowAttribute Class Reference

`#include <SourceRowAttribute.hpp>`

Inheritance diagram for minerule::SourceRowAttribute:



Public Types

- typedef char ElementType

Public Member Functions

- virtual ~SourceRowAttribute ()
- virtual SourceRowElement * copy () const =0
- virtual mrdb::Types::SQLType getType () const =0
- virtual void setValue (mrdb::ResultSet *rs, int col)=0
- virtual void setValue (const std::string &value)=0
- virtual int compareTo (const SourceRowAttribute &) const =0
- virtual std::string asString (const std::string &sep=",") const =0
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &s1) const
- virtual bool empty () const

  return true if this attribute is empty
- virtual std::string getSQLData () const =0
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const =0
- virtual ElementType getElementType () const
- virtual std::string getFullElementType () const
- virtual void serialize (std::ostream &os) const =0
- virtual void deserialize (std::istream &is)=0

Static Public Member Functions

- static SourceRowAttribute * createAttribute (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, int elem)
- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

6.153.1 Detailed Description

This is the base class for all attributes types. It should be subclassed in order to provide the actual implementation needed by each type. For instance a NumericalAttribute should exist in order to provide methods specifically tuned to handle numerical data.

Definition at line 36 of file SourceRowAttribute.hpp.

6.153.2 Member Typedef Documentation

6.153.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.153.3 Constructor & Destructor Documentation

6.153.3.1 ~SourceRowAttribute()

```
virtual minerule::SourceRowAttribute::~SourceRowAttribute ( )  [inline], [virtual]
```

Definition at line 47 of file SourceRowAttribute.hpp.

```
00047 {}
```

6.153.4 Member Function Documentation

6.153.4.1 asString()

```
virtual std::string minerule::SourceRowAttribute::asString (
            const std::string & sep = "," ) const  [pure virtual]
```

Convert this attribute to a string.

Parameters

sep	a separator string. Composite attributes should use this value to separate fields.

Returns

the std::string representation of the value of this attribute.

Implements minerule::SourceRowElement.

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.2 compareTo()

```
virtual int minerule::SourceRowAttribute::compareTo (
            const SourceRowAttribute &  ) const  [pure virtual]
```

Compares the present attribute value with the one of the given attribute.

Returns

-1 if the present attribute is less than the argument, 0 if they are equal +1 otherwise

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.3 copy()

```
virtual SourceRowElement * minerule::SourceRowAttribute::copy ( ) const  [pure virtual]
```

Returns

a fresh allocated copy of the current object

Implements minerule::SourceRowElement.

Implemented in minerule::GenericSourceRowAttribute, minerule::MemDebugGenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.4 createAttribute()

```
SourceRowAttribute * minerule::SourceRowAttribute::createAttribute (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            int elem )  [static]
```

Factory method.

Returns

an attribute capable of handling types of the given type

Factory method - returns an attribute capable of handling types of the given type

Definition at line 34 of file SourceRowAttribute.cpp.

```
00037                   {
00038                 mrdb::Types::SQLType colType = (mrdb::Types::SQLType) rsmd->getColumnType(elem);
00039 #ifdef DEBUG
00040 #warning MemDebugGeneric...
00041                 return new MemDebugGenericSourceRowAttribute(rs,elem,colType);
00042 #else
00043                 if(colType==mrdb::Types::INTEGER || colType==mrdb::Types::BIGINT)
00044                         return new NumericSourceRowAttribute(rs,elem);
00045                 else
00046                         return new GenericSourceRowAttribute(rs,elem,colType);
00047 #endif
00048         }
```

6.153.4.5 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.153.4.6 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.153.4.7 deserialize()

```
virtual void minerule::SourceRowElement::deserialize (
            std::istream & is )  [pure virtual], [inherited]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.153.4.8 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                   {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
```

6.153.4.9 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                        {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream>>elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
```

6.153.4.10 empty()

```
virtual bool minerule::SourceRowAttribute::empty ( ) const  [inline], [virtual]
```

return true if this attribute is empty

Implements minerule::SourceRowElement.

Definition at line 113 of file SourceRowAttribute.hpp.

```
00113 { return false; }
```

6.153.4.11 getElementType()

```
virtual ElementType minerule::SourceRowElement::getElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeColle
and minerule::SourceRowEmptyElement.

Definition at line 82 of file SourceRowElement.hpp.

```
00082                                          {
00083     throw MineruleException(MR_ERROR_INTERNAL,
00084                             "getElementType() not implemented!");
00085   }
```

6.153.4.12 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                              {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.153.4.13 getSQLData()

```
virtual std::string minerule::SourceRowAttribute::getSQLData ( ) const  [pure virtual]
```

Returns

the std::string representation for the attribute in a format suitable to be used in SQL queries (tipically it is implemented as return "'"+this->asString()+"'"

Implements minerule::SourceRowElement.

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.14 getType()

```
virtual mrdb::Types::SQLType minerule::SourceRowAttribute::getType ( ) const  [pure virtual]
```

Returns

the type of the attribute

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.15 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                 {
00054     return !this->operator==(el);
00055   }
```

6.153.4.16 operator()()

```
virtual bool minerule::SourceRowAttribute::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [inline], [virtual]
```

Returns

true iff s1<s2

Implements minerule::SourceRowElement.

Definition at line 97 of file SourceRowAttribute.hpp.

```
00098                                                 {
00099     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00100     const SourceRowAttribute &attr2 = dynamic_cast<const SourceRowAttribute &>(s2);
00101
00102     return attr1.compareTo(attr2) < 0;
00103   }
```

6.153.4.17 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                 {
00058     return this->operator()(*this, el);
00059   }
```

6.153.4.18 operator<<()

```
virtual std::ostream & minerule::SourceRowAttribute::operator<< (
            std::ostream & os ) const  [inline], [virtual]
```

Inserts the value of this attribute in the provided ostream and returns the ostream.

Parameters

os	an output stream

Returns

the output stream given in input

Implements minerule::SourceRowElement.

Definition at line 129 of file SourceRowAttribute.hpp.

```
00129                                                 {
00130     os << asString();
00131     return os;
00132   }
```

6.153.4.19 operator==()

```
virtual bool minerule::SourceRowAttribute::operator== (
            const SourceRowElement & s1 ) const  [inline], [virtual]
```

Returns

true iff ∗this == s1

Implements minerule::SourceRowElement.

Definition at line 106 of file SourceRowAttribute.hpp.

```
00106                                                 {
00107     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00108
00109     return this->compareTo(attr1) == 0;
00110   }
```

6.153.4.20 serialize()

```
virtual void minerule::SourceRowElement::serialize (
            std::ostream & os ) const  [pure virtual], [inherited]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.153.4.21 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
        {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.153.4.22 setPreparedStatementParameters()

```
virtual void minerule::SourceRowElement::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [pure virtual], [inherited]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.153.4.23 setValue() [1/2]

```
virtual void minerule::SourceRowAttribute::setValue (
            const std::string & value )  [pure virtual]
```

Sets the value of the attribute according to the given string (converting the value to the proper type when necessary).

Parameters

value	a string representation of the value to be set

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

6.153.4.24 setValue() [2/2]

```
virtual void minerule::SourceRowAttribute::setValue (
            mrdb::ResultSet * rs,
            int col )  [pure virtual]
```

Sets the value of the attribute accordingly to the value of the specified column of the current row of the given result set

Parameters

rs	the result set source of the value
col	the index (starting from 1) of the column in the result set

Implemented in minerule::GenericSourceRowAttribute, and minerule::NumericSourceRowAttribute.

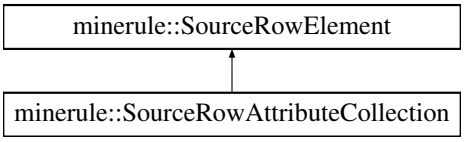
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowAttribute.cpp

6.154 minerule::SourceRowAttributeCollection Class Reference

`#include <SourceRowAttributeCollection.hpp>`

Inheritance diagram for minerule::SourceRowAttributeCollection:



Public Types

- typedef char ElementType

Public Member Functions

- SourceRowAttributeCollection ()
- SourceRowAttributeCollection (mrdb::ResultSetMetaData ∗rsmd, mrdb::ResultSet ∗rs, std::vector< int > elems)
- SourceRowAttributeCollection (const SourceRowAttributeCollection &rhs)
- virtual ∼SourceRowAttributeCollection ()
- virtual SourceRowElement & operator= (const SourceRowElement &rhs)
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &) const
- virtual bool empty () const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement ∗state, size_t start_index) const

- virtual std::string getSQLData () const
- virtual std::string asString (const std::string &sep=",") const
- virtual SourceRowElement * copy () const
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual ElementType getElementType () const
- virtual std::string getFullElementType () const
- virtual void serialize (std::ostream &os) const
- virtual void deserialize (std::istream &is)
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const

Static Public Member Functions

- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

6.154.1 Detailed Description

Implements attribute collections.

Definition at line 33 of file SourceRowAttributeCollection.hpp.

6.154.2 Member Typedef Documentation

6.154.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.154.3 Constructor & Destructor Documentation

6.154.3.1 SourceRowAttributeCollection() [1/3]

```
minerule::SourceRowAttributeCollection::SourceRowAttributeCollection ( )  [inline]
```

Definition at line 45 of file SourceRowAttributeCollection.hpp.

```
00045                                       {
00046     }
```

6.154.3.2 SourceRowAttributeCollection() [2/3]

```
minerule::SourceRowAttributeCollection::SourceRowAttributeCollection (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            std::vector< int > elems )
```

Definition at line 28 of file SourceRowAttributeCollection.cpp.

```
00031                                                                    {
00032     assert(!elems.empty());
00033
00034     std::vector<int>::const_iterator it = elems.begin();
00035     for(; it!=elems.end(); it++) {
00036       SourceRowAttribute* attr = SourceRowAttribute::createAttribute(rsmd,rs,*it);
00037       attributes.push_back(attr);
00038     }
00039   }
```

6.154.3.3 SourceRowAttributeCollection() [3/3]

```
minerule::SourceRowAttributeCollection::SourceRowAttributeCollection (
            const SourceRowAttributeCollection & rhs )
```

Definition at line 43 of file SourceRowAttributeCollection.cpp.

```
00044                                                                                    {
00045     CollectionType::const_iterator it1 = rhs.attributes.begin();
00046
00047     while( it1!=rhs.attributes.end() ) {
00048       attributes.push_back((SourceRowAttribute*)(*it1)->copy());
00049       it1++;
00050     }
00051   }
```

6.154.3.4 ~SourceRowAttributeCollection()

```
virtual minerule::SourceRowAttributeCollection::~SourceRowAttributeCollection ( )  [inline],
[virtual]
```

Definition at line 58 of file SourceRowAttributeCollection.hpp.

```
00058                                                {
00059         freeCollection();
00060     }
```

6.154.4 Member Function Documentation

6.154.4.1 asString()

```
std::string minerule::SourceRowAttributeCollection::asString (
            const std::string & sep = "," ) const  [virtual]
```

Strings related...

Implements minerule::SourceRowElement.

Definition at line 91 of file SourceRowAttributeCollection.cpp.

```
00091                                                                    {
00092     CollectionType::const_iterator it;
00093
00094     std::string attrString;
00095
00096     attrString = "";
00097
00098     //  std::cerr « "in sourcerowattrcollection::c_str collection size:"« attributes->size() «
      std::endl;
00099
00100     it=attributes.begin();
00101     if(it!=attributes.end()) {
00102       attrString+=(*it)->asString(sep);
00103       it++;
00104     }
00105
00106     for(; it!=attributes.end();it++) {
00107       attrString+=sep;
00108       attrString+=(*it)->asString(sep);
00109     }
00110
00111     return attrString;
00112   }
```

6.154.4.2 copy()

```
virtual SourceRowElement * minerule::SourceRowAttributeCollection::copy ( ) const  [inline],
[virtual]
```

Implements minerule::SourceRowElement.

Definition at line 81 of file SourceRowAttributeCollection.hpp.

```
00081                                          {
00082       return new SourceRowAttributeCollection(*this);
00083     }
```

6.154.4.3 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                  {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.154.4.4 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.154.4.5 deserialize()

```
void minerule::SourceRowAttributeCollection::deserialize (
            std::istream & is )  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 252 of file SourceRowAttributeCollection.cpp.

```
00252                                                                   {
00253    std::string buf;
00254     is » buf;
00255     while(buf!="S" && is) {
00256       if( buf.size()!=1 )
00257         throw MineruleException(MR_ERROR_INTERNAL,
00258                                 "Expecting a single letter element class id, but found:"
00259                                 +buf);
00260
00261       SourceRowAttribute* element =
00262         dynamic_cast<SourceRowAttribute*>(SourceRowElement::createElementFromType(buf[0]));
00263
00264       if(element==NULL)
00265         throw MineruleException(MR_ERROR_INTERNAL,
00266                                 "Cannot create a source row element identified by:" +
00267                                 std::string(buf));
00268       element->deserialize(is);
00269
00270       attributes.push_back(element);
00271       is » buf;
00272     }
00273
00274     if(buf!="S")
00275       throw MineruleException(MR_ERROR_INTERNAL,
00276                               "Expecting a source row attribute, but the stream ended!");
00277   }
```

6.154.4.6 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
```

6.154.4.7 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                          {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream>>elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
```

6.154.4.8 empty()

```
virtual bool minerule::SourceRowAttributeCollection::empty ( ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 69 of file SourceRowAttributeCollection.hpp.

```
00069                                {
00070       return attributes.empty();
00071     }
```

6.154.4.9 getElementType()

```
virtual ElementType minerule::SourceRowAttributeCollection::getElementType ( ) const  [inline],
[virtual]
```

Reimplemented from minerule::SourceRowElement.

Definition at line 88 of file SourceRowAttributeCollection.hpp.

```
00088                                          {
00089       return 'A';
00090     }
```

6.154.4.10 getFullElementType()

```
std::string minerule::SourceRowAttributeCollection::getFullElementType ( ) const  [virtual]
```

Reimplemented from minerule::SourceRowElement.

Definition at line 143 of file SourceRowAttributeCollection.cpp.

```
00143                                                                   {
00144           char chstr[2] = { getElementType(), '\0' };
00145           std::string result( chstr );
00146
00147           CollectionType::const_iterator it = attributes.begin();
00148           for(; it!=attributes.end(); ++it ) {
00149                   // nested attributes collections cannot be created, it suffices
00150                   // to iterate over the attributes and collect types with getElementType
00151                   // (instead of getFullElementType).
00152                   result+=(*it)->getElementType();
00153           }
00154
00155           return result;
00156   }
```

6.154.4.11 getSQLData()

```
std::string minerule::SourceRowAttributeCollection::getSQLData ( ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 126 of file SourceRowAttributeCollection.cpp.

```
00126                                                   {
00127     CollectionType::const_iterator it = attributes.begin();
00128    std::string result = "";
00129
00130     if(it!=attributes.end()) {
00131       result+= (*it)->getSQLData();
00132
00133       it++;
00134     }
00135
00136     for(; it!=attributes.end(); it++ ) {
00137       result+="," + (*it)->getSQLData();
00138     }
00139
00140     return result;
00141   }
```

6.154.4.12 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                   {
00054     return !this->operator==(el);
00055   }
```

6.154.4.13 operator()()

```
bool minerule::SourceRowAttributeCollection::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 179 of file SourceRowAttributeCollection.cpp.

```
00179
      {
00180     const SourceRowAttributeCollection& s1 =
00181       dynamic_cast<const SourceRowAttributeCollection&>(e1);
00182     const SourceRowAttributeCollection& s2 =
00183       dynamic_cast<const SourceRowAttributeCollection&>(e2);
00184
00185     CollectionType::const_iterator it1 = s1.attributes.begin();
00186     CollectionType::const_iterator it2 = s2.attributes.begin();
00187
00188
00189     for(; it1!=s1.attributes.end() &&
00190           it2!=s2.attributes.end() &&
00191           (**it1)==(**it2); it1++, it2++ );
00192
00193     // here: it1 and it2 counted the same number of elements from the start of the respective
00194     //   containers, i.e. pos(it1)==pos(it2)==J for a given J.
00195     // moreover: forall i<J s1[i]==s2[i] AND
00196     // (
00197     //  it1==it2 == end() (==> s1 == s2) OR
00198     //  it1==end() && it2!=end() (==> s1<s2)
00199     //  it1!=end() && it2!=end() && it1!=it2 (==> s1<s2 <=> (*it1)->compareTo(**it2)<0) )
00200     // )
00201
00202     if(it1==s1.attributes.end() && it2==s2.attributes.end()) // s1==s2
00203       return false;
00204
00205     if(it1==s1.attributes.end())  // s1<s2
00206       return true;
00207
00208     if(it2==s2.attributes.end()) // s1>s2
00209       return false;
00210
00211
00212     return (**it1)<(**it2);
00213   }
```

6.154.4.14 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                    {
00058     return this->operator()(*this, el);
00059   }
```

6.154.4.15 operator<<()

```
std::ostream & minerule::SourceRowAttributeCollection::operator<< (
            std::ostream & os ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 216 of file SourceRowAttributeCollection.cpp.

```
00216                                                                    {
00217     if(attributes.empty()) {
00218       os « "(attr_collection_empty)";
00219       return os;
00220     }
00221
00222     SourceRowAttributeCollection::CollectionType::const_iterator it;
00223     it = attributes.begin();
00224     os « "attr_collection:(";
00225
00226     os « **it;
00227     it++;
00228
00229     while(it!=attributes.end()) {
00230       os « "," « **it;
00231       it++;
00232     }
00233
00234     os «")";
00235
00236     return os;
00237   }
```

6.154.4.16 operator=()

```
SourceRowElement & minerule::SourceRowAttributeCollection::operator= (
            const SourceRowElement & rhs )  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 54 of file SourceRowAttributeCollection.cpp.

```
00055                                                          {
00056     try {
00057       const SourceRowAttributeCollection& rhs =
00058         dynamic_cast<const SourceRowAttributeCollection&>(_rhs);
00059
00060       freeCollection();
00061       CollectionType::const_iterator it1 = rhs.attributes.begin();
00062
00063       while( it1!=rhs.attributes.end() ) {
00064         attributes.push_back((SourceRowAttribute*)(*it1)->copy());
00065         it1++;
00066       }
00067     } catch (std::bad_cast& bc) {
00068       MRErr() « "Fatal error (SourceRowAttributeCollection::operator=)" « std::endl;
00069       throw;
00070     }
00071
00072       return *this;
00073   }
```

6.154.4.17 operator==()

```
bool minerule::SourceRowAttributeCollection::operator== (
            const SourceRowElement & e2 ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 161 of file SourceRowAttributeCollection.cpp.

```
00161                                                                    {
00162     const SourceRowAttributeCollection& s1 = *this;
00163     const SourceRowAttributeCollection& s2 =
00164       dynamic_cast<const SourceRowAttributeCollection&>(e2);
00165
00166     CollectionType::const_iterator it1 = s1.attributes.begin();
00167     CollectionType::const_iterator it2 = s2.attributes.begin();
00168
00169
00170     for(; it1!=s1.attributes.end() &&
00171           it2!=s2.attributes.end() &&
00172           (**it1)==(**it2); it1++, it2++ );
00173
00174     return (it1==s1.attributes.end() && it2==s2.attributes.end());
00175   }
```

6.154.4.18 serialize()

```
void minerule::SourceRowAttributeCollection::serialize (
            std::ostream & os ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 241 of file SourceRowAttributeCollection.cpp.

```
00241                                                                {
00242     SourceRowAttributeCollection::CollectionType::const_iterator it;
00243     for(it=attributes.begin(); it!=attributes.end(); it++) {
00244       os « " " « (*it)->getElementType() « " ";  // more elements to come, specifying type
00245       (*it)->serialize(os);
00246     }
00247
00248     os « " S "; // stop!
00249   }
```

6.154.4.19 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
      {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.154.4.20 setPreparedStatementParameters()

```
void minerule::SourceRowAttributeCollection::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 116 of file SourceRowAttributeCollection.cpp.

```
00116
                        {
00117           size_t count = start_index;
00118           for( CollectionType::const_iterator it = attributes.begin(); it!=attributes.end(); ++it ) {
00119                   (*it)->setPreparedStatementParameters(state, count++);
00120           }
00121   }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttributeCollection.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowAttributeCollection.cpp

6.155 minerule::SourceRowColumnIds Class Reference

`#include <SourceRowColumnIds.hpp>`

Public Member Functions

- SourceRowColumnIds (const SourceRowColumnIds &rhs)
- SourceRowColumnIds ()
- ~SourceRowColumnIds ()
- unsigned int setGroupElems (unsigned int start, unsigned int numCols)
- unsigned int setClusterBodyElems (unsigned int start, unsigned int numCols)
- unsigned int setBodyElems (unsigned int start, unsigned int numCols)
- unsigned int setClusterHeadElems (unsigned int start, unsigned int numCols)
- unsigned int setHeadElems (unsigned int start, unsigned int numCols)

Data Fields

- std::vector< int > groupElems
- std::vector< int > clusterBodyElems
- std::vector< int > bodyElems
- std::vector< int > clusterHeadElems
- std::vector< int > headElems

6.155.1 Detailed Description

A SourceRowColumnIds object maintains the information about which columns of the source table corresponds to which elements of the mining rule, e.g., which columns are to be used as head elements and which needs to be used as group ones.

Definition at line 34 of file SourceRowColumnIds.hpp.

6.155.2 Constructor & Destructor Documentation

6.155.2.1 SourceRowColumnIds() [1/2]

```
minerule::SourceRowColumnIds::SourceRowColumnIds (
            const SourceRowColumnIds & rhs )  [inline]
```

Definition at line 42 of file SourceRowColumnIds.hpp.

```
00043           : groupElems(rhs.groupElems), clusterBodyElems(rhs.clusterBodyElems),
00044             bodyElems(rhs.bodyElems), clusterHeadElems(rhs.clusterHeadElems),
00045             headElems(rhs.headElems) {}
```

6.155.2.2 SourceRowColumnIds() [2/2]

```
minerule::SourceRowColumnIds::SourceRowColumnIds ( )  [inline]
```

Definition at line 47 of file SourceRowColumnIds.hpp.

```
00047                           {
00048   }
```

6.155.2.3 ~SourceRowColumnIds()

```
minerule::SourceRowColumnIds::~SourceRowColumnIds ( )  [inline]
```

Definition at line 50 of file SourceRowColumnIds.hpp.

```
00050                            {
00051   }
```

6.155.3 Member Function Documentation

6.155.3.1 setBodyElems()

```
unsigned int minerule::SourceRowColumnIds::setBodyElems (
            unsigned int start,
            unsigned int numCols )
```

Parameters

start	the column at which the body elements start
numCols	the number of contiguous columns to be set

Returns

start+numCols+1

Adds the columns start, start+1,...,start+numCols-1 to the body elements.

Definition at line 42 of file SourceRowColumnIds.cpp.

```
00042                                                                    {
00043                 return setElems(bodyElems, start, numCols);
00044         }
```

6.155.3.2 setClusterBodyElems()

```
unsigned int minerule::SourceRowColumnIds::setClusterBodyElems (
            unsigned int start,
            unsigned int numCols )
```

Parameters

start	the column at which the cluster body elements start
numCols	the number of contiguous columns to be set

Returns

start+numCols+1

Adds the columns start, start+1,...,start+numCols-1 to the cluster body elements.

Definition at line 37 of file SourceRowColumnIds.cpp.

```
00037 {
00038         return setElems(clusterBodyElems, start, numCols);
00039 }
```

6.155.3.3 setClusterHeadElems()

```
unsigned int minerule::SourceRowColumnIds::setClusterHeadElems (
            unsigned int start,
            unsigned int numCols )
```

Parameters

start	the column at which the clustr head elements start
numCols	the number of contiguous columns to be set

Returns

start+numCols+1

Adds the columns start, start+1,...,start+numCols-1 to the clustr head elements.

Definition at line 47 of file SourceRowColumnIds.cpp.

```
00047 {
00048         return  setElems(clusterHeadElems, start, numCols);
00049 }
```

6.155.3.4 setGroupElems()

```
unsigned int minerule::SourceRowColumnIds::setGroupElems (
            unsigned int start,
            unsigned int numCols )
```

Parameters

start	the column at which the group elements start
numCols	the number of contiguous columns to be set

Returns

start+numCols+1

Adds the columns start, start+1,...,start+numCols-1 to the group elements.

Definition at line 32 of file SourceRowColumnIds.cpp.

```
00032                                                                    {
00033                 return setElems(groupElems, start, numCols);
00034         }
```

6.155.3.5 setHeadElems()

```
unsigned int minerule::SourceRowColumnIds::setHeadElems (
            unsigned int start,
            unsigned int numCols )
```

Parameters

start	the column at which the head elements start
numCols	the number of contiguous columns to be set

Returns

start+numCols+1

Adds the columns start, start+1,...,start+numCols-1 to the head elements.

Definition at line 52 of file SourceRowColumnIds.cpp.

```
00052                                                                    {
00053                 return setElems(headElems, start, numCols);
00054         }
```

6.155.4 Field Documentation

6.155.4.1 bodyElems

```
std::vector<int> minerule::SourceRowColumnIds::bodyElems
```

Definition at line 38 of file SourceRowColumnIds.hpp.

6.155.4.2 clusterBodyElems

```
std::vector<int> minerule::SourceRowColumnIds::clusterBodyElems
```

Definition at line 37 of file SourceRowColumnIds.hpp.

6.155.4.3 clusterHeadElems

```
std::vector<int> minerule::SourceRowColumnIds::clusterHeadElems
```

Definition at line 39 of file SourceRowColumnIds.hpp.

6.155.4.4 groupElems

```
std::vector<int> minerule::SourceRowColumnIds::groupElems
```

Definition at line 36 of file SourceRowColumnIds.hpp.

6.155.4.5 headElems

```
std::vector<int> minerule::SourceRowColumnIds::headElems
```

Definition at line 40 of file SourceRowColumnIds.hpp.

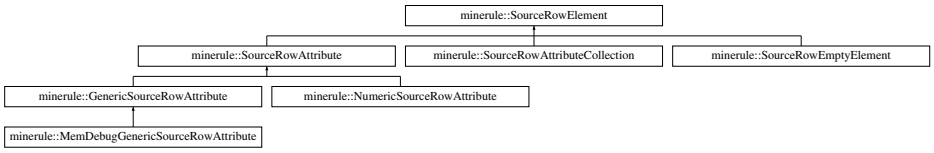
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowColumnIds.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowColumnIds.cpp

6.156 minerule::SourceRowElement Class Reference

```
#include <SourceRowElement.hpp>
```

Inheritance diagram for minerule::SourceRowElement:



Public Types

- typedef char ElementType

Public Member Functions

- virtual ~SourceRowElement ()
- virtual SourceRowElement * copy () const =0
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const =0
- virtual bool operator== (const SourceRowElement &) const =0
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual SourceRowElement & operator= (const SourceRowElement &rhs)=0
- virtual bool empty () const =0
- virtual std::string asString (const std::string &sep=",") const =0
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const =0
- virtual std::string getSQLData () const =0
- virtual ElementType getElementType () const
- virtual std::string getFullElementType () const
- virtual void serialize (std::ostream &os) const =0
- virtual void deserialize (std::istream &is)=0
- virtual std::ostream & operator<< (std::ostream &) const =0

Static Public Member Functions

- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

6.156.1 Detailed Description

Base class for all source row attributes and attributes collection.

Definition at line 34 of file SourceRowElement.hpp.

6.156.2 Member Typedef Documentation

6.156.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType
```

Definition at line 37 of file SourceRowElement.hpp.

6.156.3 Constructor & Destructor Documentation

6.156.3.1 ~SourceRowElement()

```
virtual minerule::SourceRowElement::~SourceRowElement ( )  [inline], [virtual]
```

Definition at line 43 of file SourceRowElement.hpp.

```
00043 {}
```

6.156.4 Member Function Documentation

6.156.4.1 asString()

```
virtual std::string minerule::SourceRowElement::asString (
            const std::string & sep = "," ) const  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
minerule::SourceRowEmptyElement, and minerule::SourceRowAttribute.

6.156.4.2 copy()

```
virtual SourceRowElement * minerule::SourceRowElement::copy ( ) const  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::MemDebugGenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollection, minerule::SourceRowEmptyElement, and minerule::SourceRowAttribute.

6.156.4.3 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.156.4.4 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.156.4.5 deserialize()

```
virtual void minerule::SourceRowElement::deserialize (
            std::istream & is )  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.156.4.6 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                   {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
```

6.156.4.7 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                          {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream>>elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
```

6.156.4.8 empty()

```
virtual bool minerule::SourceRowElement::empty ( ) const  [pure virtual]
```

Implemented in minerule::SourceRowAttribute, minerule::SourceRowAttributeCollection, and minerule::SourceRowEmptyElement.

6.156.4.9 getElementType()

```
virtual ElementType minerule::SourceRowElement::getElementType ( ) const  [inline], [virtual]
```

Reimplemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeColle
and minerule::SourceRowEmptyElement.

Definition at line 82 of file SourceRowElement.hpp.

```
00082                                             {
00083     throw MineruleException(MR_ERROR_INTERNAL,
00084                             "getElementType() not implemented!");
00085   }
```

6.156.4.10 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                                 {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.156.4.11 getSQLData()

```
virtual std::string minerule::SourceRowElement::getSQLData ( ) const  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
minerule::SourceRowEmptyElement, and minerule::SourceRowAttribute.

6.156.4.12 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                    {
00054     return !this->operator==(el);
00055   }
```

6.156.4.13 operator()()

```
virtual bool minerule::SourceRowElement::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [pure virtual]
```

Implemented in minerule::SourceRowAttribute, minerule::SourceRowAttributeCollection, and minerule::SourceRowEmptyElement.

6.156.4.14 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                   {
00058     return this->operator()(*this, el);
00059   }
```

6.156.4.15 operator<<()

```
virtual std::ostream & minerule::SourceRowElement::operator<< (
            std::ostream &  ) const  [pure virtual]
```

Implemented in minerule::SourceRowAttribute, minerule::SourceRowAttributeCollection, and minerule::SourceRowEmptyElement.

6.156.4.16 operator=()

```
virtual SourceRowElement & minerule::SourceRowElement::operator= (
            const SourceRowElement & rhs )  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.156.4.17 operator==()

```
virtual bool minerule::SourceRowElement::operator== (
            const SourceRowElement &  ) const  [pure virtual]
```

Implemented in minerule::SourceRowAttributeCollection, minerule::SourceRowAttribute, and minerule::SourceRowEmptyElement.

6.156.4.18 serialize()

```
virtual void minerule::SourceRowElement::serialize (
            std::ostream & os ) const  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

6.156.4.19 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
        {
00060                   std::ostringstream sstream;
00061                   sstream « elem.getElementType();
00062                   elem.serialize(sstream);
00063                   strRepr = sstream.str();
00064         }
```

6.156.4.20 setPreparedStatementParameters()

```
virtual void minerule::SourceRowElement::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [pure virtual]
```

Implemented in minerule::GenericSourceRowAttribute, minerule::NumericSourceRowAttribute, minerule::SourceRowAttributeCollecti
and minerule::SourceRowEmptyElement.

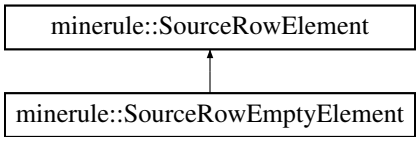
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowElement.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowElement.cpp

6.157 minerule::SourceRowEmptyElement Class Reference

`#include <SourceRowElement.hpp>`

Inheritance diagram for minerule::SourceRowEmptyElement:



Public Types

- typedef char ElementType

Public Member Functions

- SourceRowEmptyElement ()
- virtual ~SourceRowEmptyElement ()
- virtual SourceRowElement * copy () const
- virtual void setPreparedStatementParameters (mrdb::PreparedStatement *state, size_t start_index) const
- virtual bool operator() (const SourceRowElement &s1, const SourceRowElement &s2) const
- virtual bool operator== (const SourceRowElement &s1) const
- virtual SourceRowElement & operator= (const SourceRowElement &rhs)
- virtual bool empty () const
- virtual std::string asString (const std::string &sep=",") const
- virtual std::string getSQLData () const
- virtual std::ostream & operator<< (std::ostream &os) const
- virtual ElementType getElementType () const
- virtual void serialize (std::ostream &os) const
- virtual void deserialize (std::istream &is)
- virtual bool operator!= (const SourceRowElement &el) const
- virtual bool operator< (const SourceRowElement &el) const
- virtual std::string getFullElementType () const

Static Public Member Functions

- static SourceRowElement * createElement (mrdb::ResultSetMetaData *rsmd, mrdb::ResultSet *rs, const std::vector< int > &srd)
- static SourceRowElement * createElementFromType (ElementType el)
- static SourceRowElement * deserializeElementFromString (const std::string &strRepr)
- static SourceRowElement * deserializeElementFromResultSet (mrdb::ResultSet *rs, size_t start_index)
- static void serializeElementToString (const SourceRowElement &elem, std::string &strRepr)

6.157.1 Detailed Description

Definition at line 114 of file SourceRowElement.hpp.

6.157.2 Member Typedef Documentation

6.157.2.1 ElementType

```
typedef char minerule::SourceRowElement::ElementType  [inherited]
```

Definition at line 37 of file SourceRowElement.hpp.

6.157.3 Constructor & Destructor Documentation

6.157.3.1 SourceRowEmptyElement()

```
minerule::SourceRowEmptyElement::SourceRowEmptyElement ( )  [inline]
```

Definition at line 117 of file SourceRowElement.hpp.

```
00117 {}
```

6.157.3.2 ~SourceRowEmptyElement()

```
virtual minerule::SourceRowEmptyElement::~SourceRowEmptyElement ( )  [inline], [virtual]
```

Definition at line 119 of file SourceRowElement.hpp.

```
00119 {}
```

6.157.4 Member Function Documentation

6.157.4.1 asString()

```
virtual std::string minerule::SourceRowEmptyElement::asString (
            const std::string & sep = "," ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 174 of file SourceRowElement.hpp.

```
00174                                                       {
00175     return "";
00176   }
```

6.157.4.2 copy()

```
virtual SourceRowElement * minerule::SourceRowEmptyElement::copy ( ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 121 of file SourceRowElement.hpp.

```
00121 { return new SourceRowEmptyElement(); }
```

6.157.4.3 createElement()

```
SourceRowElement * minerule::SourceRowElement::createElement (
            mrdb::ResultSetMetaData * rsmd,
            mrdb::ResultSet * rs,
            const std::vector< int > & srd )  [static], [inherited]
```

Definition at line 23 of file SourceRowElement.cpp.

```
00026                                   {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
```

6.157.4.4 createElementFromType()

```
SourceRowElement * minerule::SourceRowElement::createElementFromType (
            ElementType el )  [static], [inherited]
```

Definition at line 38 of file SourceRowElement.cpp.

```
00038                                                                   {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

6.157.4.5 deserialize()

```
virtual void minerule::SourceRowEmptyElement::deserialize (
            std::istream & is )  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 189 of file SourceRowElement.hpp.

```
00189                                          {
00190     std::string null;
00191     is » null;
00192     if (null != " NULL")
00193       throw MineruleException(MR_ERROR_INTERNAL,
00194                               "NULL Element expected, but " + null + " found!");
00195   }
```

6.157.4.6 deserializeElementFromResultSet()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromResultSet (
            mrdb::ResultSet * rs,
            size_t start_index )  [static], [inherited]
```

Definition at line 67 of file SourceRowElement.cpp.

```
00067                                                                                                {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
```

6.157.4.7 deserializeElementFromString()

```
SourceRowElement * minerule::SourceRowElement::deserializeElementFromString (
            const std::string & strRepr )  [static], [inherited]
```

Definition at line 81 of file SourceRowElement.cpp.

```
00081                                                                                                {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream»elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
```

6.157.4.8 empty()

```
virtual bool minerule::SourceRowEmptyElement::empty ( ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 172 of file SourceRowElement.hpp.

```
00172 { return false; }
```

6.157.4.9 getElementType()

```
virtual ElementType minerule::SourceRowEmptyElement::getElementType ( ) const  [inline], [virtual]
```

Reimplemented from minerule::SourceRowElement.

Definition at line 185 of file SourceRowElement.hpp.

```
00185 { return '0'; }
```

6.157.4.10 getFullElementType()

```
virtual std::string minerule::SourceRowElement::getFullElementType ( ) const  [inline], [virtual],
[inherited]
```

Reimplemented in minerule::SourceRowAttributeCollection.

Definition at line 92 of file SourceRowElement.hpp.

```
00092                                           {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
```

6.157.4.11 getSQLData()

```
virtual std::string minerule::SourceRowEmptyElement::getSQLData ( ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 178 of file SourceRowElement.hpp.

```
00178 { assert(false); }
```

6.157.4.12 operator"!=()

```
virtual bool minerule::SourceRowElement::operator!= (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 53 of file SourceRowElement.hpp.

```
00053                                                 {
00054     return !this->operator==(el);
00055   }
```

6.157.4.13 operator()()

```
virtual bool minerule::SourceRowEmptyElement::operator() (
            const SourceRowElement & s1,
            const SourceRowElement & s2 ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 131 of file SourceRowElement.hpp.

```
00132                                                                {
00133     const SourceRowEmptyElement *e1 =
00134         dynamic_cast<const SourceRowEmptyElement *>(&s1);
00135
00136     const SourceRowEmptyElement *e2 =
00137         dynamic_cast<const SourceRowEmptyElement *>(&s2);
00138
00139     // if both s1 and s2 are NOT emptyElements, then the
00140     // proper function should be called
00141     if (e1 == NULL && e2 == NULL) {
00142       return s1(s1, s2);
00143       ;
00144     }
00145
00146     // an empty element is always "lesser" than any non
00147     // empty one
00148     if (e1 == NULL)
00149       return true;
00150
00151     // if s1 is empty and s2 is not than s1<s2 is false
00152     // if both are empty then s1==s2 (hence s1<s2 is false)
00153     // in any case we have to return false
00154     return false;
00155   }
```

6.157.4.14 operator<()

```
virtual bool minerule::SourceRowElement::operator< (
            const SourceRowElement & el ) const  [inline], [virtual], [inherited]
```

Definition at line 57 of file SourceRowElement.hpp.

```
00057                                                            {
00058     return this->operator()(*this, el);
00059   }
```

6.157.4.15 operator<<()

```
virtual std::ostream & minerule::SourceRowEmptyElement::operator<< (
            std::ostream & os ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 180 of file SourceRowElement.hpp.

```
00180                                                            {
00181     os « "(emptyElem)";
00182     return os;
00183   }
```

6.157.4.16 operator=()

```
virtual SourceRowElement & minerule::SourceRowEmptyElement::operator= (
            const SourceRowElement & rhs )  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 166 of file SourceRowElement.hpp.

```
00166                                                                   {
00167     assert(dynamic_cast<const SourceRowEmptyElement *>(&rhs) != NULL);
00168
00169     return *this;
00170   }
```

6.157.4.17 operator==()

```
virtual bool minerule::SourceRowEmptyElement::operator== (
            const SourceRowElement & s1 ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 157 of file SourceRowElement.hpp.

```
00157                                                                   {
00158     const SourceRowEmptyElement *e1 =
00159         dynamic_cast<const SourceRowEmptyElement *>(&s1);
00160
00161     return e1 != NULL;
00162   }
```

6.157.4.18 serialize()

```
virtual void minerule::SourceRowEmptyElement::serialize (
            std::ostream & os ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 186 of file SourceRowElement.hpp.

```
00186                                                   {
00187     os « " NULL";
00188   }
```

6.157.4.19 serializeElementToString()

```
void minerule::SourceRowElement::serializeElementToString (
            const SourceRowElement & elem,
            std::string & strRepr )  [static], [inherited]
```

Definition at line 59 of file SourceRowElement.cpp.

```
00059
      {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
```

6.157.4.20 setPreparedStatementParameters()

```
virtual void minerule::SourceRowEmptyElement::setPreparedStatementParameters (
            mrdb::PreparedStatement * state,
            size_t start_index ) const  [inline], [virtual]
```

Implements minerule::SourceRowElement.

Definition at line 123 of file SourceRowElement.hpp.

```
00124                                                                    {
00125     throw MineruleException(MR_ERROR_INTERNAL,
00126                             "setPreparedStatementParameters callled on an "
00127                             "empty element. This is a bug! Please report it!");
00128   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowElement.hpp

6.158 minerule::SourceRowMetaInfo Class Reference

```
#include <SourceRowMetaInfo.hpp>
```

Public Member Functions

- SourceRowMetaInfo (mrdb::ResultSet *rs, const SourceRowColumnIds &rowDes)
- SourceRowMetaInfo (mrdb::Connection *, const ParsedMinerule &minerule)
- const SourceRowAttrCollectionDescriptor & getGroup () const
- const SourceRowAttrCollectionDescriptor & getClusterBody () const
- const SourceRowAttrCollectionDescriptor & getBody () const
- const SourceRowAttrCollectionDescriptor & getClusterHead () const
- const SourceRowAttrCollectionDescriptor & getHead () const

6.158.1 Detailed Description

Definition at line 54 of file SourceRowMetaInfo.hpp.

6.158.2 Constructor & Destructor Documentation

6.158.2.1 SourceRowMetaInfo() [1/2]

```
SourceRowMetaInfo::SourceRowMetaInfo (
            mrdb::ResultSet * rs,
            const SourceRowColumnIds & rowDes )
```

Definition at line 116 of file SourceRowMetaInfo.cpp.

```
00117           : group(rs,rowDes.groupElems),
00118             clusterBody(rs,rowDes.clusterBodyElems),
00119             body(rs,rowDes.bodyElems),
00120             clusterHead(rs,rowDes.clusterHeadElems),
00121             head(rs,rowDes.headElems) { }
```

6.158.2.2 SourceRowMetaInfo() [2/2]

```
SourceRowMetaInfo::SourceRowMetaInfo (
            mrdb::Connection * mrdb_connection,
            const ParsedMinerule & minerule )
```

Definition at line 148 of file SourceRowMetaInfo.cpp.

```
00148                                                                                                                {
00149         ParsedMinerule::AttrVector attr_list;
00150         attr_list.insert(attr_list.end(), minerule.ga.begin(), minerule.ga.end());
00151         attr_list.insert(attr_list.end(), minerule.ca.begin(), minerule.ca.end());
00152     attr_list.insert(attr_list.end(), minerule.ba.begin(), minerule.ba.end());
00153     attr_list.insert(attr_list.end(), minerule.ca.begin(), minerule.ca.end());
00154         attr_list.insert(attr_list.end(), minerule.ha.begin(), minerule.ha.end());
00155
00156         std::string query =
00157                 "SELECT " + AttributesUtil::names_to_string(attr_list) +
00158                 " FROM " + minerule.tab_source + " LIMIT 1";
00159
00160         mrdb::Statement* state = mrdb_connection->createStatement();
00161         mrdb::ResultSet* rs = state->executeQuery(query);
00162
00163         AttributesUtil attrUtils;
00164         group.init(rs, attrUtils.generatePositions(minerule.ga) );
00165         clusterBody.init(rs, attrUtils.generatePositions(minerule.ca) );
00166         body.init(rs, attrUtils.generatePositions(minerule.ba));
00167         clusterHead.init(rs, attrUtils.generatePositions(minerule.ca));
00168         head.init(rs,attrUtils.generatePositions(minerule.ha));
00169
00170         delete rs;
00171         delete state;
00172 }
```

6.158.3 Member Function Documentation

6.158.3.1 getBody()

```
const SourceRowAttrCollectionDescriptor & minerule::SourceRowMetaInfo::getBody ( ) const  [inline]
```

Definition at line 79 of file SourceRowMetaInfo.hpp.

```
00079                                                                                                    {
00080                         return body;
00081                 }
```

6.158.3.2 getClusterBody()

```
const SourceRowAttrCollectionDescriptor & minerule::SourceRowMetaInfo::getClusterBody ( )
const  [inline]
```

Definition at line 75 of file SourceRowMetaInfo.hpp.

```
00075                                                                                                        {
00076                         return clusterBody;
00077                 }
```

6.158.3.3 getClusterHead()

```
const SourceRowAttrCollectionDescriptor & minerule::SourceRowMetaInfo::getClusterHead ( )
const  [inline]
```

Definition at line 83 of file SourceRowMetaInfo.hpp.

```
00083                                                                    {
00084                         return clusterHead;
00085                 }
```

6.158.3.4 getGroup()

```
const SourceRowAttrCollectionDescriptor & minerule::SourceRowMetaInfo::getGroup ( ) const
[inline]
```

Definition at line 71 of file SourceRowMetaInfo.hpp.

```
00071                                                                  {
00072                         return group;
00073                 }
```

6.158.3.5 getHead()

```
const SourceRowAttrCollectionDescriptor & minerule::SourceRowMetaInfo::getHead ( ) const  [inline]
```

Definition at line 87 of file SourceRowMetaInfo.hpp.

```
00087                                                                  {
00088                         return head;
00089                 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceRowMetaInfo.hpp
- /Users/esposito/Software/minerule/src/Database/SourceRowMetaInfo.cpp

6.159 minerule::SourceTable Class Reference

```
#include <SourceTable.hpp>
```

Data Structures

- class Iterator

Public Types

- enum IteratorKind { BodyIterator , HeadIterator , FullIterator }

Public Member Functions

- SourceTable (const MiningAlgorithm &algorithm)
- SourceTable (const ParsedMinerule &minerule, const SourceTableRequirements &requirements)
- virtual ~SourceTable ()
- bool usesCrossProduct () const
- void init ()
- size_t getTotGroups ()
- Iterator newIterator (IteratorKind)

6.159.1 Detailed Description

Definition at line 24 of file SourceTable.hpp.

6.159.2 Member Enumeration Documentation

6.159.2.1 IteratorKind

```
enum minerule::SourceTable::IteratorKind
```

Enumerator

BodyIterator	
HeadIterator	
FullIterator	

Definition at line 26 of file SourceTable.hpp.
```
00026 { BodyIterator, HeadIterator, FullIterator } IteratorKind;
```

6.159.3 Constructor & Destructor Documentation

6.159.3.1 SourceTable() [1/2]

```
minerule::SourceTable::SourceTable (
            const MiningAlgorithm & algorithm )  [inline]
```

Definition at line 71 of file SourceTable.hpp.
```
00072       : _minerule(algorithm.optimizedMinerule().getParsedMinerule()),
00073         _sourceTableRequirements(algorithm.sourceTableRequirements()),
00074         _pdu(_minerule, _sourceTableRequirements), _usesCrossProduct(false),
00075         _bodyStatement(NULL), _headStatement(NULL), _fullStatement(NULL) {
00076     init();
00077   };
```

6.159.3.2 SourceTable() [2/2]

```
minerule::SourceTable::SourceTable (
            const ParsedMinerule & minerule,
            const SourceTableRequirements & requirements )  [inline]
```

Definition at line 79 of file SourceTable.hpp.

```
00081       : _minerule(minerule), _sourceTableRequirements(requirements),
00082         _pdu(_minerule, _sourceTableRequirements), _usesCrossProduct(false),
00083                                 _bodyStatement(NULL), _headStatement(NULL), _fullStatement(NULL)  {
00084     init();
00085   }
```

6.159.3.3 ~SourceTable()

```
minerule::SourceTable::~SourceTable ( )  [virtual]
```

Definition at line 22 of file SourceTable.cpp.

```
00022                                   {
00023                 for(std::vector<mrdb::ResultSet*>::const_iterator it = _managedResults.begin();
       it!=_managedResults.end(); ++it) {
00024                         delete *it;
00025                 }
00026         }
```

6.159.4 Member Function Documentation

6.159.4.1 getTotGroups()

```
size_t minerule::SourceTable::getTotGroups ( )
```

Definition at line 73 of file SourceTable.cpp.

```
00073                                       {
00074                 return _pdu.evaluateTotGroups();
00075         }
```

6.159.4.2 init()

```
void minerule::SourceTable::init ( )
```

Definition at line 106 of file SourceTable.cpp.

```
00106                                   {
00107                 if( _sourceTableRequirements.crossProduct() )
00108                         initFullResultSet();
00109                 else
00110                         initBodyHeadResultSets();
00111         }
```

6.159.4.3 newIterator()

```
SourceTable::Iterator minerule::SourceTable::newIterator (
            SourceTable::IteratorKind kind )
```

Definition at line 45 of file SourceTable.cpp.

```
00045                                                                                {
00046                 switch(kind) {
00047                         case BodyIterator:
00048                                 {
00049                                         _managedResults.push_back(_bodyStatement->executeQuery());
00050                                         SourceRowColumnIds bodyCols = _columnIds;
00051                                         bodyCols.headElems.clear();
00052 
00053                                         return Iterator(_managedResults.back(), bodyCols);
00054                                 }
00055                         case HeadIterator:
00056                                 {
00057                                         _managedResults.push_back(_headStatement->executeQuery());
00058                                         SourceRowColumnIds headCols = _columnIds;
00059                                         headCols.bodyElems.clear();
00060 
00061                                         return Iterator(_managedResults.back(), headCols);
00062                                 }
00063                         case FullIterator:
00064                                 {
00065                                         _managedResults.push_back(_fullStatement->executeQuery());
00066                                         return Iterator(_managedResults.back(), _columnIds);
00067                                 }
00068                 }
00069 
00070                 throw MineruleException(MR_ERROR_INTERNAL, "Uknown IteratorKind. This is a bug, please
      report it!");
00071         }
```

6.159.4.4 usesCrossProduct()

```
bool minerule::SourceTable::usesCrossProduct ( ) const  [inline]
```

Definition at line 89 of file SourceTable.hpp.

```
00089 { return _usesCrossProduct; }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceTable.hpp
- /Users/esposito/Software/minerule/src/Database/SourceTable.cpp

6.160 minerule::SourceTableRequirements Class Reference

```
#include <SourceTableRequirements.hpp>
```

Public Types

- enum Requirements { SortedGids = 1 , CrossProduct = 2 }
- typedef unsigned short RequirementsSet

Public Member Functions

- SourceTableRequirements ()
- SourceTableRequirements (RequirementsSet requirements)
- virtual ~SourceTableRequirements ()
- bool sortedGids () const
- bool crossProduct () const
- void set (RequirementsSet requirements)

6.160.1 Detailed Description

Definition at line 29 of file SourceTableRequirements.hpp.

6.160.2 Member Typedef Documentation

6.160.2.1 RequirementsSet

```
typedef unsigned short minerule::SourceTableRequirements::RequirementsSet
```

Definition at line 32 of file SourceTableRequirements.hpp.

6.160.3 Member Enumeration Documentation

6.160.3.1 Requirements

```
enum minerule::SourceTableRequirements::Requirements
```

Enumerator

SortedGids	
CrossProduct	

Definition at line 31 of file SourceTableRequirements.hpp.

```
00031 { SortedGids = 1, CrossProduct = 2 } Requirements;
```

6.160.4 Constructor & Destructor Documentation

6.160.4.1 SourceTableRequirements() [1/2]

```
minerule::SourceTableRequirements::SourceTableRequirements ( )  [inline]
```

Definition at line 34 of file SourceTableRequirements.hpp.

```
00034 : _requirements(0) {}
```

6.160.4.2 SourceTableRequirements() [2/2]

```
minerule::SourceTableRequirements::SourceTableRequirements (
            RequirementsSet requirements )  [inline]
```

Definition at line 35 of file SourceTableRequirements.hpp.

```
00035 : _requirements(requirements) { }
```

6.160.4.3 ~SourceTableRequirements()

```
virtual minerule::SourceTableRequirements::~SourceTableRequirements ( )  [inline], [virtual]
```

Definition at line 36 of file SourceTableRequirements.hpp.

```
00036 {}
```

6.160.5 Member Function Documentation

6.160.5.1 crossProduct()

```
bool minerule::SourceTableRequirements::crossProduct ( ) const  [inline]
```

Definition at line 39 of file SourceTableRequirements.hpp.

```
00039 { return _requirements & CrossProduct; }
```

6.160.5.2 set()

```
void minerule::SourceTableRequirements::set (
            RequirementsSet requirements )  [inline]
```

Definition at line 41 of file SourceTableRequirements.hpp.

```
00041 { _requirements = requirements; }
```

6.160.5.3 sortedGids()

```
bool minerule::SourceTableRequirements::sortedGids ( ) const  [inline]
```

Definition at line 38 of file SourceTableRequirements.hpp.

```
00038 { return _requirements & SortedGids;   }
```

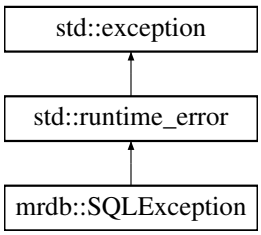
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/SourceTableRequirements.hpp

6.161 mrdb::SQLException Class Reference

```
#include <SQLException.hpp>
```

Inheritance diagram for mrdb::SQLException:



Public Member Functions

- SQLException (const std::string &message)
- virtual ~SQLException () _NOEXCEPT

6.161.1 Detailed Description

Definition at line 12 of file SQLException.hpp.

6.161.2 Constructor & Destructor Documentation

6.161.2.1 SQLException()

```
mrdb::SQLException::SQLException (
            const std::string & message )  [inline]
```

Definition at line 14 of file SQLException.hpp.

```
00014 : std::runtime_error(message){};
```

6.161.2.2 ~SQLException()

```
virtual mrdb::SQLException::~SQLException ( )  [inline], [virtual]
```

Definition at line 15 of file SQLException.hpp.
```
00015 {}
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/SQLException.hpp

6.162 minerule::SQLUtils Class Reference

```
#include <SQLUtils.hpp>
```

Public Types

- enum Type {
  Numeric , String , Binary , DateTime ,
  Bit }

Static Public Member Functions

- static void removeHeadBodyFromAttrName (std::string &str)
- static Type getType (mrdb::Types::SQLType)
- static Type getType (mrdb::ResultSet *rs, int colNum)
- static Type getType (mrdb::Connection *connection, const std::string &tabName, std::string colName)
- static bool isNumericType (mrdb::Types::SQLType type)
- static bool isStringType (mrdb::Types::SQLType type)
- static bool isBinaryType (mrdb::Types::SQLType type)
- static bool isDateTimeType (mrdb::Types::SQLType type)
- static bool isBitType (mrdb::Types::SQLType type)
- static std::string quote (const std::string &str)
- static bool isAttribute (const std::string &str)

6.162.1 Detailed Description

Definition at line 31 of file SQLUtils.hpp.

6.162.2 Member Enumeration Documentation

6.162.2.1 Type

```
enum minerule::SQLUtils::Type
```

Enumerator

Numeric	
String	
Binary	
DateTime	
Bit	

Definition at line 33 of file SQLUtils.hpp.

```
00033                 {
00034       Numeric,
00035       String,
00036       Binary,
00037       DateTime,
00038       Bit
00039     } Type;
```

6.162.3 Member Function Documentation

6.162.3.1 getType() [1/3]

```
SQLUtils::Type minerule::SQLUtils::getType (
            mrdb::Connection * connection,
            const std::string & tabName,
            std::string colName )  [static]
```

Definition at line 95 of file SQLUtils.cpp.

```
00097                                                   {
00098   removeHeadBodyFromAttrName(colName);
00099
00100   try {
00101     mrdb::DatabaseMetaData *dbmd = connection->getMetaData();
00102
00103     return getType(dbmd->getColumnType(tabName, colName));
00104   } catch (mrdb::SQLException &e) {
00105     throw MineruleException(MR_ERROR_DATABASE_ERROR,
00106                             std::string("Cannot access to the database metadata the reason is:") +
00107                                 e.what());
00108   }
00109
00110   throw MineruleException(MR_ERROR_DATABASE_ERROR,
00111                           "Cannot access to the database metadata");
00112 }
```

6.162.3.2 getType() [2/3]

```
static Type minerule::SQLUtils::getType (
            mrdb::ResultSet * rs,
            int colNum )  [inline], [static]
```

Definition at line 48 of file SQLUtils.hpp.

```
00048                                           {
00049       return getType( (mrdb::Types::SQLType)
00050                       rs->getMetaData()->getColumnType(colNum) );
00051     }
```

6.162.3.3 getType() [3/3]

```
SQLUtils::Type minerule::SQLUtils::getType (
            mrdb::Types::SQLType type )  [static]
```

Definition at line 28 of file SQLUtils.cpp.

```
00028                                          {
00029   switch (type) {
00030   case mrdb::Types::BIGINT:
00031   case mrdb::Types::DECIMAL:
00032   case mrdb::Types::DOUBLE:
00033   case mrdb::Types::FLOAT:
00034   case mrdb::Types::INTEGER:
00035   case mrdb::Types::NUMERIC:
00036   case mrdb::Types::REAL:
00037   case mrdb::Types::SMALLINT:
00038   case mrdb::Types::TINYINT:
00039     return Numeric;
00040   case mrdb::Types::CHAR:
00041   case mrdb::Types::VARCHAR:
00042   case mrdb::Types::LONGVARCHAR:
00043     return String;
00044   case mrdb::Types::DATE:
00045   case mrdb::Types::TIME:
00046   case mrdb::Types::TIMESTAMP:
00047     return DateTime;
00048   case mrdb::Types::BINARY:
00049   case mrdb::Types::LONGVARBINARY:
00050   case mrdb::Types::VARBINARY:
00051     return Binary;
00052   case mrdb::Types::BIT:
00053     return Bit;
00054   default:
00055     throw MineruleException(MR_ERROR_INTERNAL,
00056                             "Unexpected kind of SQL data type:" +
00057                                 Converter(type).toString());
00058   }
00059 }
```

6.162.3.4 isAttribute()

```
static bool minerule::SQLUtils::isAttribute (
            const std::string & str )  [inline], [static]
```

Definition at line 85 of file SQLUtils.hpp.

```
00085                                                {
00086     if(str.length()==0)
00087       return false;
00088
00089     if( isdigit(str[0]) )
00090       return false;
00091
00092     if( str.find_first_of("\"'")!=str.npos )
00093       return false;
00094
00095     if( Converter(str).isNumber() )
00096       return false;
00097
00098     return true;
00099     }
```

6.162.3.5 isBinaryType()

```
static bool minerule::SQLUtils::isBinaryType (
            mrdb::Types::SQLType type )  [inline], [static]
```

Definition at line 69 of file SQLUtils.hpp.

```
00069                                              {
00070       return getType(type)==Binary;
00071     }
```

6.162.3.6 isBitType()

```
static bool minerule::SQLUtils::isBitType (
            mrdb::Types::SQLType type )  [inline], [static]
```

Definition at line 79 of file SQLUtils.hpp.

```
00079                                       {
00080       return getType(type)==Bit;
00081     }
```

6.162.3.7 isDateTimeType()

```
static bool minerule::SQLUtils::isDateTimeType (
            mrdb::Types::SQLType type )  [inline], [static]
```

Definition at line 74 of file SQLUtils.hpp.

```
00074                                           {
00075       return getType(type)==DateTime;
00076     }
```

6.162.3.8 isNumericType()

```
static bool minerule::SQLUtils::isNumericType (
            mrdb::Types::SQLType type )  [inline], [static]
```

Definition at line 59 of file SQLUtils.hpp.

```
00059                                           {
00060       return getType(type)==Numeric;
00061     }
```

6.162.3.9 isStringType()

```
static bool minerule::SQLUtils::isStringType (
            mrdb::Types::SQLType type )  [inline], [static]
```

Definition at line 64 of file SQLUtils.hpp.

```
00064                                           {
00065       return getType(type)==String;
00066     }
```

6.162.3.10 quote()

```
std::string minerule::SQLUtils::quote (
            const std::string & str )  [static]
```

Definition at line 61 of file SQLUtils.cpp.

```
00061                                                {
00062   // We have to substitute each "'" with "''" in str (since
00063   // in sql values cannot contain single "'".
00064   std::string rest = str;
00065   std::string result = "";
00066
00067   while (rest != "") {
00068     size_t p = rest.find("'");
00069     if (p == rest.npos) {
00070       result += rest;
00071       rest = "";
00072     } else {
00073       result += rest.substr(0, p);
00074       result += "''";
00075       rest = rest.substr(p + 1, rest.length() - p - 1);
00076     }
00077   }
00078
00079   return "'" + result + "'";
00080 }
```

6.162.3.11 removeHeadBodyFromAttrName()

```
void minerule::SQLUtils::removeHeadBodyFromAttrName (
            std::string & str )  [static]
```

Definition at line 82 of file SQLUtils.cpp.

```
00082                                                          {
00083   size_t pos;
00084   if ((pos = str.find("HEAD.")) != str.npos) {
00085     str.erase(pos, 5);
00086   }
00087   if ((pos = str.find("BODY.")) != str.npos) {
00088     str.erase(pos, 5);
00089   }
00090 }
```

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/SQLUtils.hpp
- /Users/esposito/Software/minerule/src/Utils/SQLUtils.cpp

6.163 mrdb::Statement Class Reference

`#include <Statement.hpp>`

Public Member Functions

- virtual ~Statement ()
- virtual ResultSet * executeQuery (const std::string &sql)=0
- virtual bool execute (const std::string &sql)=0

6.163.1 Detailed Description

Definition at line 11 of file Statement.hpp.

6.163.2 Constructor & Destructor Documentation

6.163.2.1 ~Statement()

```
virtual mrdb::Statement::~Statement ( )  [inline], [virtual]
```

Definition at line 13 of file Statement.hpp.

```
00013 {}
```

6.163.3 Member Function Documentation

6.163.3.1 execute()

```
virtual bool mrdb::Statement::execute (
            const std::string & sql )  [pure virtual]
```

Execute the given query.

Returns

true if a result set was available, false otherwise and false otherwise.

Exceptions

a	MineruleException (with error code MR_ERROR_DATABASE_ERROR) in case something goes wrong.

6.163.3.2 executeQuery()

```
virtual ResultSet * mrdb::Statement::executeQuery (
            const std::string & sql )  [pure virtual]
```

Executes the given query and build the correspondig result set. THREAD SAFETY: It returns a ResultSet that should be consumed by the same thread that owns this object.

Returns

the ResultSet @memory the caller is responsible of deallocing the result set once it is no longer needed.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/Statement.hpp

6.164 minerule::StringCompare Class Reference

`#include <Interval.hpp>`

Public Member Functions

- bool operator() (const char *lhs, const char *rhs) const

6.164.1 Detailed Description

Definition at line 35 of file Interval.hpp.

6.164.2 Member Function Documentation

6.164.2.1 operator()()

```
bool minerule::StringCompare::operator() (
            const char * lhs,
            const char * rhs ) const  [inline]
```

Definition at line 37 of file Interval.hpp.

```
00037                                                    {
00038       return strcmp(lhs,rhs)<0;
00039     }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp

6.165 minerule::StringUtils Class Reference

`#include <StringUtils.hpp>`

Static Public Member Functions

- static std::vector< std::string > * splitToLength (const std::string &str, size_t len)
- static std::vector< std::string > split (const std::string &str, const std::string &sep)
- static std::string join (const std::vector< std::string > &, const std::string &sep)
- static void setColorsEnabled (bool newVal)
- static std::string toBold (const std::string &str)
- static std::string toRed (const std::string &str)
- static std::string toBoldRed (const std::string &str)
- static std::string toGreen (const std::string &str)
- static std::string toWhite (const std::string &str)
- static std::string toBlue (const std::string &str)
- static std::string toYellow (const std::string &str)
- static std::string toOrange (const std::string &str)

Static Public Attributes

- static const std::string RED = "\x1B[31m"
- static const std::string BOLD = "\x1B[1m"
- static const std::string GREEN = "\x1B[32m"
- static const std::string YELLOW = "\x1B[33m"
- static const std::string BLUE = "\x1B[34m"
- static const std::string WHITE = "\x1B[37m"
- static const std::string CLOSE = "\x1B[0m"

6.165.1 Detailed Description

Definition at line 23 of file StringUtils.hpp.

6.165.2 Member Function Documentation

6.165.2.1 join()

```
std::string minerule::StringUtils::join (
            const std::vector< std::string > & vec,
            const std::string & sep )  [static]
```

Definition at line 76 of file StringUtils.cpp.

```
00076                                                                                {
00077                 std::string result;
00078                 for(std::vector<std::string>::const_iterator it=vec.begin(); it!=vec.end(); ++it) {
00079                         if(it!=vec.begin()) { result += sep; }
00080                         result += *it;
00081                 }
00082
00083                 return result;
00084         }
```

6.165.2.2 setColorsEnabled()

```
static void minerule::StringUtils::setColorsEnabled (
            bool newVal )  [inline], [static]
```

Definition at line 41 of file StringUtils.hpp.

```
00041 { enableColors = newVal; }
```

6.165.2.3 split()

```
std::vector< std::string > minerule::StringUtils::split (
            const std::string & str,
            const std::string & sep )  [static]
```

Definition at line 57 of file StringUtils.cpp.

```
00057                                                                                                                {
00058                 std::vector<std::string> result;
00059                 size_t pos = 0;
00060                 size_t size = str.size();
00061                 while( pos < size ) {
00062                         size_t found_pos = str.find(sep,pos);
00063                         std::string cur_piece = str.substr(pos,found_pos);
00064                         pos+=cur_piece.size();
00065
00066                         result.push_back(cur_piece);
00067
00068                         if( pos < size ) { // if something where found, we move beyond the sep string.
00069                                 pos+=sep.size();
00070                         }
00071                 }
00072
00073                 return result;
00074         }
```

6.165.2.4 splitToLength()

```
std::vector< std::string > * minerule::StringUtils::splitToLength (
            const std::string & str,
            size_t len )  [static]
```

Definition at line 29 of file StringUtils.cpp.

```
00029                                                                                                                {
00030                 std::vector<std::string>* result = new std::vector<std::string>;
00031                 if(str.size() <= out_len) {
00032                         result->push_back(str);
00033                         return result;
00034                 }
00035
00036                 size_t len = str.size();
00037                 size_t cur_pos = 0;
00038                 while( cur_pos < len ) {
00039                         std::string chunk;
00040                         size_t new_pos = std::min( cur_pos + out_len, len );
00041
00042
00043                         size_t last_space = str.find_last_of(" ", new_pos);
00044                         if( last_space != std::string::npos &&  // found something AND
00045                                         last_space > cur_pos &&
          // still moving forward AND
00046                                         last_space + 10 >= new_pos &&                    // not too far
     back w.r.t. from new_pos
00047                                         !(cur_pos == len))
                                      // not at the end of the string
00048                                 new_pos = last_space+1;
00049
00050                         result->push_back(str.substr(cur_pos, new_pos-cur_pos));
00051                         cur_pos = new_pos;
00052                 }
00053
00054                 return result;
00055         }
```

6.165.2.5 toBlue()

```
static std::string minerule::StringUtils::toBlue (
            const std::string & str )  [inline], [static]
```

Definition at line 47 of file StringUtils.hpp.
```
00047 { return colorize(str,BLUE); }
```

6.165.2.6 toBold()

```
static std::string minerule::StringUtils::toBold (
            const std::string & str )  [inline], [static]
```

Definition at line 42 of file StringUtils.hpp.
```
00042 { return colorize(str, BOLD); }
```

6.165.2.7 toBoldRed()

```
static std::string minerule::StringUtils::toBoldRed (
            const std::string & str )  [inline], [static]
```

Definition at line 44 of file StringUtils.hpp.
```
00044 { return colorize(colorize(str, RED), BOLD); }
```

6.165.2.8 toGreen()

```
static std::string minerule::StringUtils::toGreen (
            const std::string & str )  [inline], [static]
```

Definition at line 45 of file StringUtils.hpp.
```
00045 { return colorize(str,GREEN); }
```

6.165.2.9 toOrange()

```
static std::string minerule::StringUtils::toOrange (
            const std::string & str )  [inline], [static]
```

Definition at line 49 of file StringUtils.hpp.
```
00049 { return colorize(colorize(str,YELLOW), BOLD); }
```

6.165.2.10 toRed()

```
static std::string minerule::StringUtils::toRed (
            const std::string & str )  [inline], [static]
```

Definition at line 43 of file StringUtils.hpp.

```
00043 { return colorize(str,RED);   }
```

6.165.2.11 toWhite()

```
static std::string minerule::StringUtils::toWhite (
            const std::string & str )  [inline], [static]
```

Definition at line 46 of file StringUtils.hpp.

```
00046 { return colorize(str,WHITE); }
```

6.165.2.12 toYellow()

```
static std::string minerule::StringUtils::toYellow (
            const std::string & str )  [inline], [static]
```

Definition at line 48 of file StringUtils.hpp.

```
00048 { return colorize(str,YELLOW); }
```

6.165.3 Field Documentation

6.165.3.1 BLUE

```
const std::string minerule::StringUtils::BLUE = "\x1B[34m"  [static]
```

Definition at line 37 of file StringUtils.hpp.

6.165.3.2 BOLD

```
const std::string minerule::StringUtils::BOLD = "\x1B[1m"  [static]
```

Definition at line 34 of file StringUtils.hpp.

6.165.3.3 CLOSE

```
const std::string minerule::StringUtils::CLOSE = "\x1B[0m"  [static]
```

Definition at line 39 of file StringUtils.hpp.

6.165.3.4 GREEN

```
const std::string minerule::StringUtils::GREEN = "\x1B[32m"  [static]
```

Definition at line 35 of file StringUtils.hpp.

6.165.3.5 RED

```
const std::string minerule::StringUtils::RED = "\x1B[31m"  [static]
```

Definition at line 33 of file StringUtils.hpp.

6.165.3.6 WHITE

```
const std::string minerule::StringUtils::WHITE = "\x1B[37m"  [static]
```

Definition at line 38 of file StringUtils.hpp.

6.165.3.7 YELLOW

```
const std::string minerule::StringUtils::YELLOW = "\x1B[33m"  [static]
```

Definition at line 36 of file StringUtils.hpp.

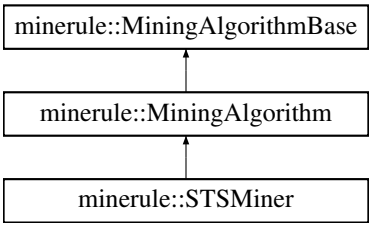
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/StringUtils.hpp
- /Users/esposito/Software/minerule/src/Utils/StringUtils.cpp

6.166 minerule::STSMiner Class Reference

`#include <STSMiner.hpp>`

Inheritance diagram for minerule::STSMiner:



Public Member Functions

- STSMiner (const OptimizedMinerule &mr, const AlgorithmsOptions &opts)
- virtual bool canHandleMinerule () const
- virtual void mineRules ()
- virtual ~STSMiner ()
- virtual SourceTableRequirements sourceTableRequirements () const
- virtual void initialize ()
- virtual void execute ()
- virtual const OptimizedMinerule & optimizedMinerule () const

Static Public Member Functions

- static MiningAlgorithmBase * algorithmForType (AlgorithmTypes t, const OptimizedMinerule &)

Protected Attributes

- Connection connection
- const OptimizedMinerule & minerule

6.166.1 Detailed Description

Definition at line 44 of file STSMiner.hpp.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 STSMiner()

```
minerule::STSMiner::STSMiner (
            const OptimizedMinerule & mr,
            const AlgorithmsOptions & opts )  [inline]
```

Definition at line 152 of file STSMiner.hpp.

```
00152                                                                                :
00153         MiningAlgorithm(mr),pm(ParsedMinerule(mr.getParsedMinerule())) {
00154         MRDebug()«"STSMiner initialization..."«std::endl;
00155        /* double vm, rss;
00156         process_mem_usage(vm, rss);
00157         std::cout « "VM: " « vm « "; RSS: " « rss « std::endl;*/
00158
00159         this->initialize();
00160         connection.createResultTables();
00161
00162         std::string temp = "/tmp/" + pm.tab_result + ".s";
00163         seqicache.open(temp.c_str(), std::ofstream::trunc);
00164
00165         temp = "/tmp/" + pm.tab_result + ".m";
00166         matcher_cache.open(temp.c_str(), std::ofstream::trunc);
00167
00168         options = opts;
00169         log=true;
00170         max_seq_length = pm.length.getMax();
00171         min_seq_length = pm.length.getMin();
00172         active_count=false;
00173
00174         groupAttrList = "";
00175         ordAttrList = "";
00176         bodyAttrList = "";
00177
00178         for(int i=0; i<pm.ga.size()-1; ++i)
00179             groupAttrList+=pm.ga[i]+",";
00180         groupAttrList+=pm.ga[pm.ga.size()-1];
00181
00182         for(int i=0; i<pm.oa.size()-1; ++i)
00183             ordAttrList+=pm.oa[i]+",";
00184         ordAttrList+=pm.oa[pm.oa.size()-1];
00185
00186         for(int i=0; i<pm.ba.size()-1; ++i)
00187             bodyAttrList+=pm.ba[i]+",";
00188         bodyAttrList+=pm.ba[pm.ba.size()-1];
00189
00190         seq_count=1;
00191         DISTINCT=pm.distinct;
00192         context_dep_dist=false;
00193         context_dep_BEM=false;
00194         context_free_BEM=false;
00195         max_MIN_COUNT=1;
00196         bemCD_constraints=0;
00197         bemCF_constraints=0;
00198
00199         initAttrNameList();
00200
00201         for(int i=0;i<pm.seq_bem_vect.size();++i) {
00202             Bem_cond* bc = pm.seq_bem_vect[i];
00203             bool cf=false;
00204             bool cd=false;
00205
00206             do {
00207                 if(!containedIn(bc->attr,pm.ba)) {
00208                     context_dep_BEM=true;
00209                     cd=true;
00210                 }
00211                 else {
00212                     context_free_BEM=true;
00213                     cf=true;
00214                 }
00215                 bc=bc->and_c;
00216             }while(bc);
00217             if(cd)
00218                 bemCD_constraints++;
00219             if(cf)
00220                 bemCF_constraints++;
00221         }
00222
00223         for(int i=0;i<pm.seq_dist_vect.size();++i) {
00224             for(int j=0; j<pm.seq_dist_vect[i]->attr.size();++j) {
00225                 if(!containedIn(pm.seq_dist_vect[i]->attr[j],pm.ba))
00226                     context_dep_dist=true;
00227             }
00228         }
```

```
00229
00230         int x=0;
00231         for(int i=0;i<pm.seq_bem_vect.size();++i)
00232             if(pm.seq_bem_vect[i]->type.compare("MID")==0)
00233                 x++;
00234         bem_vect_length= 3 >= 2+x ? 3 : 2+x;
00235
00236         if(bem_vect_length>0) {
00237
00238             int max_int_val=std::numeric_limits<int>::max();
00239             for(int i=1;i<bem_vect_length-1;++i)
00240                 min_max_mid_count.push_back(MinMaxPair(1,max_int_val));
00241             x=0;
00242             for(int i=0;i<pm.seq_bem_vect.size();++i) {
00243                 if(pm.seq_bem_vect[i]->type.compare("MID")==0) {
00244                     Bem_cond* temp=pm.seq_bem_vect[i];
00245                     do {
00246                         max_MIN_COUNT=std::max(max_MIN_COUNT,temp->count_min);
00247
       min_max_mid_count[x].setMin(std::max(min_max_mid_count[x].getMin(),temp->count_min));
00248
       min_max_mid_count[x].setMax(std::min(min_max_mid_count[x].getMax(),temp->count_max));
00249                         temp = temp->and_c;
00250                     } while(temp);
00251
00252                     active_count= active_count || min_max_mid_count[x].getMin()!=1
00253                          || min_max_mid_count[x].getMax()!=std::numeric_limits<int>::max();
00254                     ++x;
00255                 }
00256             }
00257         }
00258         initAttrTypesList(pm.tab_source);
00259     }
```

6.166.2.2 ~STSMiner()

```
virtual minerule::STSMiner::~STSMiner ( )  [inline], [virtual]
```

Definition at line 267 of file STSMiner.hpp.

```
00267 { }
```

6.166.3 Member Function Documentation

6.166.3.1 algorithmForType()

```
MiningAlgorithmBase * minerule::MiningAlgorithmBase::algorithmForType (
            AlgorithmTypes t,
            const OptimizedMinerule & mr )  [static], [inherited]
```

Definition at line 25 of file MiningAlgorithmBase.cpp.

```
00025                                                                                {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
```

6.166.3.2 canHandleMinerule()

```
virtual bool minerule::STSMiner::canHandleMinerule ( ) const  [inline], [virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 261 of file STSMiner.hpp.

```
00261                                       {
00262       return pm.miningTask == MTMineSequences;
00263     }
```

6.166.3.3 execute()

```
virtual void minerule::MiningAlgorithm::execute ( )  [inline], [virtual], [inherited]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 93 of file MiningAlgorithmBase.hpp.

```
00093                                   {
00094                         initialize();
00095                         mineRules();
00096                 }
```

6.166.3.4 initialize()

```
virtual void minerule::MiningAlgorithm::initialize ( )  [inline], [virtual], [inherited]
```

Definition at line 66 of file MiningAlgorithmBase.hpp.

```
00066                                   {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
```

6.166.3.5 mineRules()

```
void minerule::STSMiner::mineRules ( )  [virtual]
```

Reimplemented from minerule::MiningAlgorithm.

Definition at line 1572 of file STSMiner.cpp.

```
01572                           {
01573 
01574     int k = 3;
01575 
01576     try {
01577         MRLogPop();
01578         std::cout«std::endl;
01579         MRLogPush("Starting STSMiner mining algorithm...");
01580         dropTable(pm.tab_result+"_matcher");
01581 
01582         std::string strK;
01583         bool EmptyResult = false;
01584 
01585         if(log)
01586             MRLog() « "Extracting 1-Sequences set..." « std::endl;
01587 
01588         if(!extract1Sequences()) {
01589 
01590             int singleton_num = pruning1Seq();
01591             if(singleton_num>0) {
01592                 createBitVectors();
01593                 if(singleton_num>1 && max_seq_length>1) {
01594                     if(log)
01595                         MRLog() « "Extracting 2-Sequences set..."«std::endl;
01596                     EmptyResult = extract2Sequences();
01597                     if(!EmptyResult && max_seq_length==2) {
01598                         saveSequences(2);
01599                         delete_all(2);
01600                     }
01601 
01602                     while(!EmptyResult && k <= max_seq_length) {
01603                         if(k%10 == 0)
01604                             MRLog()«"Please wait..."«std::endl;
01605                         strK = Converter(k).toString();
01606                         if(log)
01607                             MRLog() « "Extracting " + strK + "-Sequences set..."«std::endl;
01608                         EmptyResult = extractKSequences(k);
01609 
01610                         if(EmptyResult)
01611                             MRLog()« "Cannot generate sequences of length " + strK « std::endl;
01612                         else if(k==max_seq_length) {
01613                             saveSequences(k);
01614                             delete_all(k);
01615                         }
01616                         ++k;
01617                     }
01618                 }
01619                 else {
01620                     saveSequences(1);
01621                     delete_all(1);
01622                 }
01623             }//nothing generated...
01624             else
01625                 delete_all(1);
01626         }//Source table is empty
01627         else
01628             k=2;
01629         std::string empty_t_name=pm.tab_result +"_Seq"+ Converter(k-1).toString();
01630         dropTable(empty_t_name);
01631         dropTable(pm.tab_result+"_Source");
01632         MRLogPop();
01633         MRLogPush("Saving frequent patterns...");
01634         connection.finalize(true);
01635         finalize_matcher(seq_count>1);
01636         MRLogPop();
01637         show_Results(log);
01638         std::cout«std::endl;
01639         MRLog()«"Extracted "+Converter(seq_count-1).toString()+" sequences."«std::endl;
01640     } catch (const mrdb::SQLException& e) {
01641         std::cout«StringUtils::toBoldRed("MRDB Exception:");
01642         std::cout«e.what() «std::endl;
01643         dropResultsTables(k);
01644         delete_all(-1);
01645         throw e;
01646     } catch (const minerule::MineruleException& e) {
01647         std::cout«StringUtils::toBoldRed("Minerule Exception:");
```

```
01648         std::cout<<e.what()<<std::endl;
01649         dropResultsTables(k);
01650         delete_all(-1);
01651         throw e;
01652     } catch (const std::exception& e) {
01653         std::cout<<StringUtils::toBoldRed("Exception:");
01654         std::cout<<e.what()<<std::endl;
01655         dropResultsTables(k);
01656         delete_all(-1);
01657         throw e;
01658     }   catch(...) {
01659         std::cout<<StringUtils::toBoldRed("An unknown exception has been thrown... \n ");
01660         dropResultsTables(k);
01661         delete_all(-1);
01662         throw std::exception();
01663     }
01664 }
```

6.166.3.6 optimizedMinerule()

```
virtual const OptimizedMinerule & minerule::MiningAlgorithmBase::optimizedMinerule ( ) const
[inline], [virtual], [inherited]
```

Definition at line 52 of file MiningAlgorithmBase.hpp.

```
00052 { return minerule; }
```

6.166.3.7 sourceTableRequirements()

```
virtual SourceTableRequirements minerule::STSMiner::sourceTableRequirements ( ) const  [inline],
[virtual]
```

Reimplemented from minerule::MiningAlgorithmBase.

Definition at line 269 of file STSMiner.hpp.

```
00269                                                                   {
00270             return SourceTableRequirements(SourceTableRequirements::SortedGids);
00271     }
```

6.166.4 Field Documentation

6.166.4.1 connection

```
Connection minerule::MiningAlgorithm::connection  [protected], [inherited]
```

Definition at line 62 of file MiningAlgorithmBase.hpp.

6.166.4.2 minerule

```
const OptimizedMinerule& minerule::MiningAlgorithmBase::minerule  [protected], [inherited]
```

Definition at line 34 of file MiningAlgorithmBase.hpp.

The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp
- /Users/esposito/Software/minerule/src/Algorithms/STSMiner.cpp

6.167 minerule::QueryNormalizer::SubstEntryBody Class Reference

```
#include <QueryNormalizer.hpp>
```

Public Member Functions

- SubstEntryBody ()
- SubstEntryBody (const SubstEntryBody &seb)
- bool operator< (const SubstEntryBody &s2) const

Data Fields

- list_AND_node ** pred
- size_t pos

6.167.1 Detailed Description

Definition at line 37 of file QueryNormalizer.hpp.

6.167.2 Constructor & Destructor Documentation

6.167.2.1 SubstEntryBody() [1/2]

```
minerule::QueryNormalizer::SubstEntryBody::SubstEntryBody ( )  [inline]
```

Definition at line 46 of file QueryNormalizer.hpp.

```
00046                                {
00047         pred=NULL;
00048         pos=0;
00049       }
```

6.167.2.2 SubstEntryBody() [2/2]

```
minerule::QueryNormalizer::SubstEntryBody::SubstEntryBody (
            const SubstEntryBody & seb )  [inline]
```

Definition at line 51 of file QueryNormalizer.hpp.

```
00051                                                :
00052         pred( seb.pred ),
00053         pos( seb.pos ) { }
```

6.167.3 Member Function Documentation

6.167.3.1 operator<()

```
bool minerule::QueryNormalizer::SubstEntryBody::operator< (
            const SubstEntryBody & s2 ) const  [inline]
```

Definition at line 55 of file QueryNormalizer.hpp.

```
00055                                                {
00056         return this->pos < s2.pos;
00057       }
```

6.167.4 Field Documentation

6.167.4.1 pos

```
size_t minerule::QueryNormalizer::SubstEntryBody::pos
```

Definition at line 44 of file QueryNormalizer.hpp.

6.167.4.2 pred

```
list_AND_node** minerule::QueryNormalizer::SubstEntryBody::pred
```

Definition at line 43 of file QueryNormalizer.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp

6.168 minerule::QueryNormalizer::SubstEntryHead Class Reference

```
#include <QueryNormalizer.hpp>
```

Data Structures

- class Elem

Public Member Functions

- SubstEntryHead ()
- SubstEntryHead (const SubstEntryHead &seh)
- bool operator< (const SubstEntryHead &s2) const

Data Fields

- OptimizerCatalogue::KeyCols refKey
- std::vector< Elem > elems

6.168.1 Detailed Description

Definition at line 60 of file QueryNormalizer.hpp.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 SubstEntryHead() [1/2]

```
minerule::QueryNormalizer::SubstEntryHead::SubstEntryHead ( )  [inline]
```

Definition at line 84 of file QueryNormalizer.hpp.

```
00084                         {
00085       }
```

6.168.2.2 SubstEntryHead() [2/2]

```
minerule::QueryNormalizer::SubstEntryHead::SubstEntryHead (
            const SubstEntryHead & seh )  [inline]
```

Definition at line 87 of file QueryNormalizer.hpp.

```
00087                                                 :
00088         refKey( seh.refKey ),
00089         elems( seh.elems ) {}
```

6.168.3 Member Function Documentation

6.168.3.1 operator<()

```
bool minerule::QueryNormalizer::SubstEntryHead::operator< (
            const SubstEntryHead & s2 ) const  [inline]
```

Definition at line 91 of file QueryNormalizer.hpp.

```
00091                                                   {
00092         return this->refKey<s2.refKey;
00093       }
```

6.168.4 Field Documentation

6.168.4.1 elems

```
std::vector<Elem> minerule::QueryNormalizer::SubstEntryHead::elems
```

Definition at line 82 of file QueryNormalizer.hpp.

6.168.4.2 refKey

```
OptimizerCatalogue::KeyCols minerule::QueryNormalizer::SubstEntryHead::refKey
```

Definition at line 81 of file QueryNormalizer.hpp.

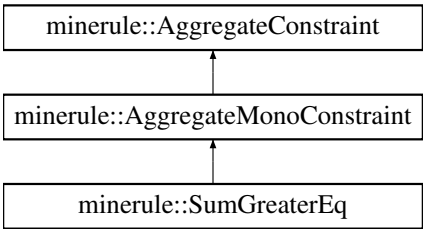
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp

6.169 minerule::SumGreaterEq Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::SumGreaterEq:



Public Member Functions

- SumGreaterEq (int ai, int v)
- SumGreaterEq (int v)
- SumGreaterEq ()
- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)
- virtual bool check (Transaction &items)

Data Fields

- int attributeIndex

6.169.1 Detailed Description

Definition at line 90 of file Constraints.hpp.

6.169.2 Constructor & Destructor Documentation

6.169.2.1 SumGreaterEq() [1/3]

```
minerule::SumGreaterEq::SumGreaterEq (
            int ai,
            int v )  [inline]
```

Definition at line 93 of file Constraints.hpp.

```
00093 : value(v) { attributeIndex = ai;}
```

6.169.2.2 SumGreaterEq() [2/3]

```
minerule::SumGreaterEq::SumGreaterEq (
            int v )  [inline]
```

Definition at line 94 of file Constraints.hpp.

```
00094 : value(v)  {}
```

6.169.2.3 SumGreaterEq() [3/3]

```
minerule::SumGreaterEq::SumGreaterEq ( )  [inline]
```

Definition at line 95 of file Constraints.hpp.

```
00095 : value(0) {}
```

6.169.3 Member Function Documentation

6.169.3.1 check() [1/2]

```
bool minerule::SumGreaterEq::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [virtual]
```

Implements minerule::AggregateMonoConstraint.

Definition at line 68 of file Constraints.cpp.

```
00068                                                                       {
00069         int sum = 0;
00070         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00071                 sum += itemMap[*i].attribute[attributeIndex].maxValue();
00072         return sum >= value;
00073 }
```

6.169.3.2 check() [2/2]

```
bool minerule::SumGreaterEq::check (
            Transaction & items )  [virtual]
```

Implements minerule::AggregateMonoConstraint.

Definition at line 75 of file Constraints.cpp.

```
00075                                                {
00076         int sum = 0; int j = 0;
00077         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00078                 sum += items.values[j++];
00079                 //sum += items.values[attributeIndex][j++];
00080         return sum >= value;
00081 }
```

6.169.4 Field Documentation

6.169.4.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

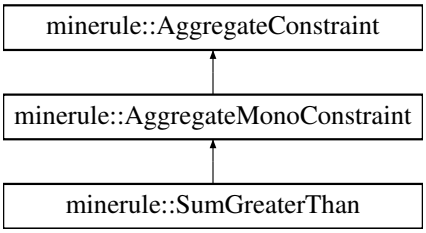
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp
- /Users/esposito/Software/minerule/src/Utils/Constraints.cpp

6.170 minerule::SumGreaterThan Class Reference

`#include <Constraints.hpp>`

Inheritance diagram for minerule::SumGreaterThan:



Public Member Functions

- SumGreaterThan (int ai, int v)
- SumGreaterThan (int v)
- SumGreaterThan ()
- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)
- virtual bool check (Transaction &items)

Data Fields

- int attributeIndex

6.170.1 Detailed Description

Definition at line 80 of file Constraints.hpp.

6.170.2 Constructor & Destructor Documentation

6.170.2.1 SumGreaterThan() [1/3]

```
minerule::SumGreaterThan::SumGreaterThan (
            int ai,
            int v )  [inline]
```

Definition at line 83 of file Constraints.hpp.

```
00083 : value(v) { attributeIndex = ai;}
```

6.170.2.2 SumGreaterThan() [2/3]

```
minerule::SumGreaterThan::SumGreaterThan (
            int v )  [inline]
```

Definition at line 84 of file Constraints.hpp.

```
00084 : value(v)  {}
```

6.170.2.3 SumGreaterThan() [3/3]

```
minerule::SumGreaterThan::SumGreaterThan ( )  [inline]
```

Definition at line 85 of file Constraints.hpp.

```
00085 : value(0) {}
```

6.170.3 Member Function Documentation

6.170.3.1 check() [1/2]

```
bool minerule::SumGreaterThan::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [virtual]
```

Implements minerule::AggregateMonoConstraint.

Definition at line 53 of file Constraints.cpp.

```
00053                                                                       {
00054         int sum = 0;
00055         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00056                 sum += itemMap[*i].attribute[attributeIndex].maxValue();
00057         return sum > value;
00058 }
```

6.170.3.2 check() [2/2]

```
bool minerule::SumGreaterThan::check (
            Transaction & items )  [virtual]
```

Implements minerule::AggregateMonoConstraint.

Definition at line 60 of file Constraints.cpp.

```
00060                                           {
00061         int sum = 0; int j = 0;
00062         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00063                 sum += items.values[j++];
00064                 //sum += items.values[attributeIndex][j++];
00065         return sum > value;
00066 }
```

6.170.4 Field Documentation

6.170.4.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

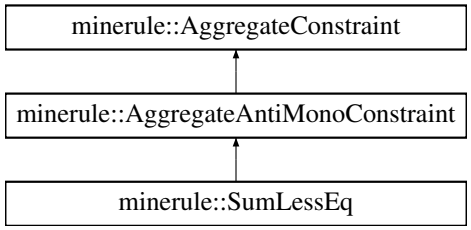
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp
- /Users/esposito/Software/minerule/src/Utils/Constraints.cpp

6.171 minerule::SumLessEq Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::SumLessEq:



Public Member Functions

- SumLessEq (int ai, int v)
- SumLessEq (int v)
- SumLessEq ()
- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)
- virtual bool check (Transaction &items)

Data Fields

- int attributeIndex

6.171.1 Detailed Description

Definition at line 70 of file Constraints.hpp.

6.171.2 Constructor & Destructor Documentation

6.171.2.1 SumLessEq() [1/3]

```
minerule::SumLessEq::SumLessEq (
            int ai,
            int v )  [inline]
```

Definition at line 73 of file Constraints.hpp.

```
00073 : value(v) { attributeIndex = ai;}
```

6.171.2.2 SumLessEq() [2/3]

```
minerule::SumLessEq::SumLessEq (
            int v )  [inline]
```

Definition at line 74 of file Constraints.hpp.

```
00074 : value(v)  {}
```

6.171.2.3 SumLessEq() [3/3]

```
minerule::SumLessEq::SumLessEq ( )  [inline]
```

Definition at line 75 of file Constraints.hpp.

```
00075 : value(0) {}
```

6.171.3 Member Function Documentation

6.171.3.1 check() [1/2]

```
bool minerule::SumLessEq::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [virtual]
```

Implements minerule::AggregateAntiMonoConstraint.

Definition at line 38 of file Constraints.cpp.

```
00038                                                                   {
00039         int sum = 0;
00040         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00041                 sum += itemMap[*i].attribute[attributeIndex].minValue();
00042         return sum <= value;
00043 }
```

6.171.3.2 check() [2/2]

```
bool minerule::SumLessEq::check (
            Transaction & items )  [virtual]
```

Implements minerule::AggregateAntiMonoConstraint.

Definition at line 45 of file Constraints.cpp.

```
00045                                       {
00046         int sum = 0; int j = 0;
00047         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00048                 sum += items.values[j++];
00049                 //sum += items.values[attributeIndex][j++];
00050         return sum <= value;
00051 }
```

6.171.4 Field Documentation

6.171.4.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

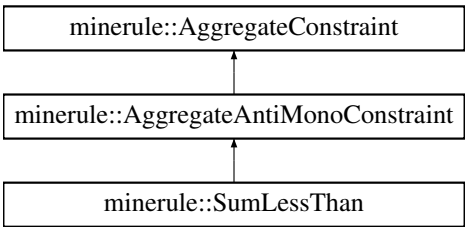
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp
- /Users/esposito/Software/minerule/src/Utils/Constraints.cpp

6.172 minerule::SumLessThan Class Reference

```
#include <Constraints.hpp>
```

Inheritance diagram for minerule::SumLessThan:



Public Member Functions

- SumLessThan (int ai, int v)
- SumLessThan (int v)
- SumLessThan ()
- virtual bool check (BodyMap &itemMap, std::vector< ItemType > &items)
- virtual bool check (Transaction &items)

Data Fields

- int attributeIndex

6.172.1 Detailed Description

Definition at line 60 of file Constraints.hpp.

6.172.2 Constructor & Destructor Documentation

6.172.2.1 SumLessThan() [1/3]

```
minerule::SumLessThan::SumLessThan (
            int ai,
            int v )  [inline]
```

Definition at line 63 of file Constraints.hpp.

```
00063 : value(v) { attributeIndex = ai;}
```

6.172.2.2 SumLessThan() [2/3]

```
minerule::SumLessThan::SumLessThan (
            int v )  [inline]
```

Definition at line 64 of file Constraints.hpp.

```
00064 : value(v)  {}
```

6.172.2.3 SumLessThan() [3/3]

```
minerule::SumLessThan::SumLessThan ( )  [inline]
```

Definition at line 65 of file Constraints.hpp.

```
00065 : value(0) {}
```

6.172.3 Member Function Documentation

6.172.3.1 check() [1/2]

```
bool minerule::SumLessThan::check (
            BodyMap & itemMap,
            std::vector< ItemType > & items )  [virtual]
```

Implements minerule::AggregateAntiMonoConstraint.

Definition at line 23 of file Constraints.cpp.

```
00023                                                                   {
00024         int sum = 0;
00025         for(std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum < value; i++)
00026                 sum += itemMap[*i].attribute[attributeIndex].minValue();
00027         return sum < value;
00028 }
```

6.172.3.2 check() [2/2]

```
bool minerule::SumLessThan::check (
            Transaction & items )  [virtual]
```

Implements minerule::AggregateAntiMonoConstraint.

Definition at line 30 of file Constraints.cpp.

```
00030                                           {
00031         int sum = 0; int j = 0;
00032         for (Transaction::iterator i = items.begin(); i != items.end() && sum < value; i++)
00033                 sum += items.values[j++];
00034                 //sum += items.values[attributeIndex][j++];
00035         return sum < value;
00036 }
```

6.172.4 Field Documentation

6.172.4.1 attributeIndex

```
int minerule::AggregateConstraint::attributeIndex  [inherited]
```

Definition at line 28 of file Constraints.hpp.

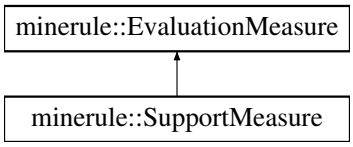
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp
- /Users/esposito/Software/minerule/src/Utils/Constraints.cpp

6.173 minerule::SupportMeasure Class Reference

`#include <SupportMeasure.hpp>`

Inheritance diagram for minerule::SupportMeasure:



Public Types

- enum MeasureType { MONOTONE , ANTIMONOTONE , OTHER }
- enum RelOperator { LessEqual , Less , Greater , GreaterEqual }

Public Member Functions

- SupportMeasure (EvaluationMeasure::RelOperator rel, double th)
- bool evalSupport (double freq)
- EvaluationMeasure::MeasureType getMeasureType ()
- double getThreshold ()

6.173.1 Detailed Description

Definition at line 11 of file SupportMeasure.hpp.

6.173.2 Member Enumeration Documentation

6.173.2.1 MeasureType

`enum minerule::EvaluationMeasure::MeasureType [inherited]`

Enumerator

MONOTONE	
ANTIMONOTONE	
OTHER	

Definition at line 13 of file EvaluationMeasure.hpp.

```
00013             {
00014     MONOTONE,
00015     ANTIMONOTONE,
00016     OTHER
```

```
00017   } MeasureType;
```

6.173.2.2 RelOperator

```
enum minerule::EvaluationMeasure::RelOperator  [inherited]
```

Enumerator

LessEqual	
Less	
Greater	
GreaterEqual	

Definition at line 19 of file EvaluationMeasure.hpp.

```
00019                {
00020     LessEqual,
00021     Less,
00022     Greater,
00023     GreaterEqual
00024   } RelOperator;
```

6.173.3 Constructor & Destructor Documentation

6.173.3.1 SupportMeasure()

```
minerule::SupportMeasure::SupportMeasure (
             EvaluationMeasure::RelOperator rel,
             double th )  [inline]
```

Definition at line 19 of file SupportMeasure.hpp.

```
00019 : relOp(rel), threshold(th) {};
```

6.173.4 Member Function Documentation

6.173.4.1 evalSupport()

```
bool minerule::SupportMeasure::evalSupport (
             double freq )
```

Definition at line 11 of file SupportMeasure.cpp.

```
00011                                          {
00012     switch (relOp){
00013     case EvaluationMeasure::LessEqual:
00014       return freq <= threshold;
00015         break;
00016     case EvaluationMeasure::Less:
```

```
00017       return freq < threshold;
00018         break;
00019     case EvaluationMeasure::Greater:
00020       return freq > threshold;
00021         break;
00022     case EvaluationMeasure::GreaterEqual:
00023       return freq >= threshold;
00024         break;
00025     default: assert(false);
00026     }
00027   }
```

6.173.4.2 getMeasureType()

```
EvaluationMeasure::MeasureType minerule::SupportMeasure::getMeasureType ( )  [virtual]
```

Implements minerule::EvaluationMeasure.

Definition at line 29 of file SupportMeasure.cpp.

```
00029                                                                 {
00030     if(relOp == EvaluationMeasure::LessEqual || relOp == EvaluationMeasure::Less)
00031       return EvaluationMeasure::MONOTONE;
00032     else
00033       if(relOp == EvaluationMeasure::GreaterEqual || relOp == EvaluationMeasure::Greater)
00034         return EvaluationMeasure::ANTIMONOTONE;
00035       else assert(false);
00036   }
```

6.173.4.3 getThreshold()

```
double minerule::SupportMeasure::getThreshold ( )  [inline]
```

Definition at line 39 of file SupportMeasure.hpp.

```
00039                         {
00040     return threshold;
00041   }
```

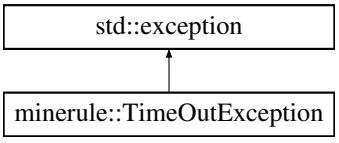
The documentation for this class was generated from the following files:

- /Users/esposito/Software/minerule/include/minerule/Parsers/SupportMeasure.hpp
- /Users/esposito/Software/minerule/src/Parsers/SupportMeasure.cpp

6.174 minerule::TimeOutException Class Reference

```
#include <GAQueryCombinator.hpp>
```

Inheritance diagram for minerule::TimeOutException:



6.174.1 Detailed Description

Definition at line 47 of file GAQueryCombinator.hpp.

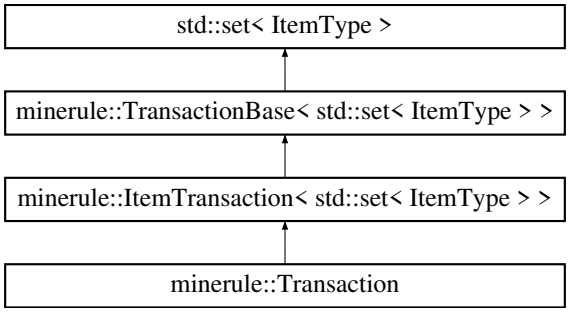
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Optimizer/GAQueryCombinator.hpp

6.175 minerule::Transaction Class Reference

`#include <Bodymap.hpp>`

Inheritance diagram for minerule::Transaction:



Public Member Functions

- void loadBody (ItemType &gid, SourceTable::Iterator &it)
- void loadHead (ItemType &gid, SourceTable::Iterator &it)

Static Public Member Functions

- static bool findGid (ItemType &gid, SourceTable::Iterator &it)

Data Fields

- std::vector< int > values
- K keys

  STL member.

6.175.1 Detailed Description

Definition at line 43 of file Bodymap.hpp.

6.175.2 Member Function Documentation

6.175.2.1 findGid()

```
static bool minerule::TransactionBase< std::set< ItemType > >::findGid (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [static], [inherited]
```

Definition at line 32 of file Transaction.hpp.

```
00032                                                                   {
00033                         while( !it.isAfterLast() && gid > it->getGroup() ) {
00034                                 ++it;
00035                         }
00036
00037                         return !it.isAfterLast() && gid == it->getGroup();
00038                 }
```

6.175.2.2 loadBody()

```
void minerule::ItemTransaction< std::set< ItemType > >::loadBody (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [inherited]
```

Definition at line 50 of file Transaction.hpp.

```
00050                                                                   {
00051                         while (!it.isAfterLast() && gid == it->getGroup()) {
00052
00053                                 ItemSetType::insert(it->getBody());
00054                                 ++it;
00055                         }
00056                 }
```

6.175.2.3 loadHead()

```
void minerule::ItemTransaction< std::set< ItemType > >::loadHead (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [inherited]
```

Definition at line 58 of file Transaction.hpp.

```
00058                                                                   {
00059                         while (!it.isAfterLast() && gid == it->getGroup()) {
00060
00061                                 ItemSetType::insert(it->getHead());
00062                                 ++it;
00063                         }
00064                 }
```

6.175.3 Field Documentation

6.175.3.1 keys

```
K std::set< K >::keys  [inherited]
```

STL member.

6.175.3.2 values

```
std::vector<int> minerule::Transaction::values
```

Definition at line 45 of file Bodymap.hpp.

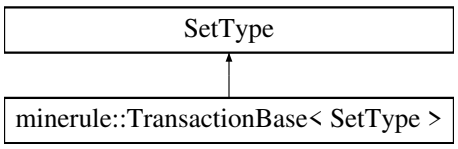
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp

6.176 minerule::TransactionBase< SetType > Class Template Reference

```
#include <Transaction.hpp>
```

Inheritance diagram for minerule::TransactionBase< SetType >:



Public Member Functions

- TransactionBase ()

Static Public Member Functions

- static bool findGid (ItemType &gid, SourceTable::Iterator &it)

6.176.1 Detailed Description

template<class SetType>
class minerule::TransactionBase< SetType >

Definition at line 28 of file Transaction.hpp.

6.176.2 Constructor & Destructor Documentation

6.176.2.1 TransactionBase()

```
template<class SetType >
minerule::TransactionBase< SetType >::TransactionBase ( )  [inline]
```

Definition at line 30 of file Transaction.hpp.

```
00030 : SetType() {}
```

6.176.3 Member Function Documentation

6.176.3.1 findGid()

```
template<class SetType >
static bool minerule::TransactionBase< SetType >::findGid (
            ItemType & gid,
            SourceTable::Iterator & it )  [inline], [static]
```

Definition at line 32 of file Transaction.hpp.

```
00032                                                                    {
00033                         while( !it.isAfterLast() && gid > it->getGroup() ) {
00034                                 ++it;
00035                         }
00036
00037                         return !it.isAfterLast() && gid == it->getGroup();
00038                 }
```

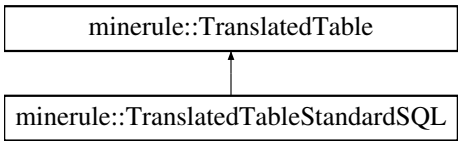
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/Database/Transaction.hpp

6.177 minerule::TranslatedTable Class Reference

```
#include <translatedtable.hpp>
```

Inheritance diagram for minerule::TranslatedTable:



Public Member Functions

- virtualstd::string getTranslatedName () const =0
- virtual std::string getOriginalName () const =0
- virtual std::string getTranslatedColumnName (const std::string &columnName) const =0
- virtual std::string getTranslatedValue (const std::string &columnName, const std::string &value) const =0
- virtual std::string getOriginalValue (const std::string &columnName, const std::string &translatedValue) const =0

6.177.1 Detailed Description

Provides methods to access to translated tables.

Definition at line 31 of file translatedtable.hpp.

6.177.2 Member Function Documentation

6.177.2.1 getOriginalName()

```
virtual std::string minerule::TranslatedTable::getOriginalName ( ) const  [pure virtual]
```

Returns

the name of the original table.

Implemented in minerule::TranslatedTableStandardSQL.

6.177.2.2 getOriginalValue()

```
virtual std::string minerule::TranslatedTable::getOriginalValue (
            const std::string & columnName,
            const std::string & translatedValue ) const  [pure virtual]
```

Parameters

columnName	a column name of the original table
translatedValue	a translated value

Returns

the value of translatedValue in the original table

Implemented in minerule::TranslatedTableStandardSQL.

6.177.2.3 getTranslatedColumnName()

```
virtual std::string minerule::TranslatedTable::getTranslatedColumnName (
            const std::string & columnName ) const  [pure virtual]
```

Takes a column name of the current table and returns its translated name. The returned value can be used to build queries on the translated tables.

Parameters

columnName	the name of a table column

Returns

the name of the translated column

Implemented in minerule::TranslatedTableStandardSQL.

6.177.2.4 getTranslatedName()

```
virtualstd::string minerule::TranslatedTable::getTranslatedName ( ) const  [pure virtual]
```

Returns

the name of the translated table. The returned value can be used to form queries involving the translated values.

Implemented in minerule::TranslatedTableStandardSQL.

6.177.2.5 getTranslatedValue()

```
virtual std::string minerule::TranslatedTable::getTranslatedValue (
            const std::string & columnName,
            const std::string & value ) const  [pure virtual]
```

Parameters

columnName	a column name of the original table
value	the value to be translated

Returns

the translatedValue

Implemented in minerule::TranslatedTableStandardSQL.

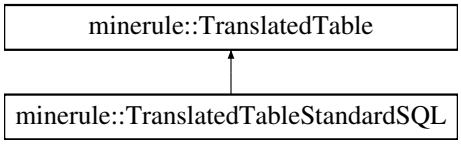
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtable.hpp

6.178 minerule::TranslatedTableStandardSQL Class Reference

`#include <translatedtablestandardsql.hpp>`

Inheritance diagram for minerule::TranslatedTableStandardSQL:



Public Member Functions

- TranslatedTableStandardSQL (const TranslationManagerStandardSQL *manager, const std::string &par↩ OriginalTable, const std::string &parTranslatedTable, bool parTranslateEverything)
- virtualstd::string getTranslatedName () const
- virtualstd::string getOriginalName () const
- virtualstd::string getTranslatedColumnName (const std::string &columnName) const
- virtualstd::string getTranslatedValue (const std::string &columnName, const std::string &value) const
- virtual std::string getOriginalValue (const std::string &columnName, const std::string &translatedValue) const
- virtual ~TranslatedTableStandardSQL ()

Protected Member Functions

- void setOriginalTableName (const std::string &table)
- void setTranslatedTableName (const std::string &table)
- virtual void updateColumnsToTranslate ()

Friends

- class TranslationManagerStandardSQL

6.178.1 Detailed Description

Definition at line 45 of file translatedtablestandardsql.hpp.

6.178.2 Constructor & Destructor Documentation

6.178.2.1 TranslatedTableStandardSQL()

```
minerule::TranslatedTableStandardSQL::TranslatedTableStandardSQL (
            const TranslationManagerStandardSQL * manager,
            const std::string & parOriginalTable,
            const std::string & parTranslatedTable,
            bool parTranslateEverything )  [inline]
```

Definition at line 82 of file translatedtablestandardsql.hpp.

```
00085                                                    :
00086     tmanager(manager), originalTable(parOriginalTable), translatedTable(parTranslatedTable) {
00087     translateEverything = parTranslateEverything;
00088
00089     updateColumnsToTranslate();
00090   }
```

6.178.2.2 ~TranslatedTableStandardSQL()

```
virtual minerule::TranslatedTableStandardSQL::~TranslatedTableStandardSQL ( )  [virtual]
```

6.178.3 Member Function Documentation

6.178.3.1 getOriginalName()

```
virtualstd::string minerule::TranslatedTableStandardSQL::getOriginalName ( ) const  [virtual]
```

Returns

the name of the original table.

Implements minerule::TranslatedTable.

6.178.3.2 getOriginalValue()

```
virtual std::string minerule::TranslatedTableStandardSQL::getOriginalValue (
            const std::string & columnName,
            const std::string & translatedValue ) const  [virtual]
```

Parameters

columnName	a column name of the original table
translatedValue	a translated value

Returns

the value of translatedValue in the original table

Implements minerule::TranslatedTable.

6.178.3.3 getTranslatedColumnName()

```
virtualstd::string minerule::TranslatedTableStandardSQL::getTranslatedColumnName (
            const std::string & columnName ) const  [virtual]
```

Takes a column name of the current table and returns its translated name. The returned value can be used to build queries on the translated tables.

Parameters

columnName	the name of a table column

Returns

the name of the translated column

Implements minerule::TranslatedTable.

6.178.3.4 getTranslatedName()

```
virtualstd::string minerule::TranslatedTableStandardSQL::getTranslatedName ( ) const  [virtual]
```

Returns

the name of the translated table. The returned value can be used to form queries involving the translated values.

Implements minerule::TranslatedTable.

6.178.3.5 getTranslatedValue()

```
virtualstd::string minerule::TranslatedTableStandardSQL::getTranslatedValue (
            const std::string & columnName,
            const std::string & value ) const  [virtual]
```

Parameters

columnName	a column name of the original table
value	the value to be translated

Returns

the translatedValue

Implements minerule::TranslatedTable.

6.178.3.6 setOriginalTableName()

```
void minerule::TranslatedTableStandardSQL::setOriginalTableName (
            const std::string & table )  [inline], [protected]
```

Definition at line 63 of file translatedtablestandardsql.hpp.

```
00063                                                {
00064     originalTable = table;
00065   }
```

6.178.3.7 setTranslatedTableName()

```
void minerule::TranslatedTableStandardSQL::setTranslatedTableName (
            const std::string & table )  [inline], [protected]
```

Definition at line 68 of file translatedtablestandardsql.hpp.

```
00068                                                {
00069     translatedTable = table;
00070   }
```

6.178.3.8 updateColumnsToTranslate()

```
virtual void minerule::TranslatedTableStandardSQL::updateColumnsToTranslate ( )  [protected],
[virtual]
```

This method should be called only in the table constructor, it update the translateEverything table in order to reflect the actual configuration of manager->translateEverything and the table structure.

6.178.4 Friends And Related Function Documentation

6.178.4.1 TranslationManagerStandardSQL

```
friend class TranslationManagerStandardSQL  [friend]
```

Definition at line 46 of file translatedtablestandardsql.hpp.

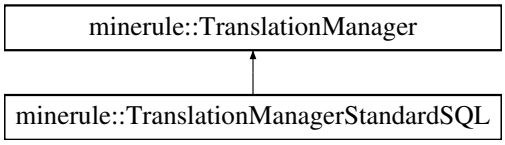
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtablestandardsql.hpp

6.179 minerule::TranslationManager Class Reference

```
#include <translationmanager.hpp>
```

Inheritance diagram for minerule::TranslationManager:



Public Member Functions

- virtual TranslatedTable * translateTable (const std::string &tableName) const =0
- virtual pair< bool, string > alreadyTranslated (const std::string &tableName) const =0

Protected Member Functions

- std::string getTranslationTableNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getTranslatedColumnNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getOriginalColumnNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getTranslationTableNameForTable (const std::string &tableName) const

6.179.1 Detailed Description

Definition at line 54 of file translationmanager.hpp.

6.179.2 Member Function Documentation

6.179.2.1 alreadyTranslated()

```
virtual pair< bool, string > minerule::TranslationManager::alreadyTranslated (
            const std::string & tableName ) const  [pure virtual]
```

Look up if, in the current DB, there is an original table whose name is tableName and for which a translatedTable already exists. If so, it returns <true, nameOfTranslatedTable>, otherwise it returns <false, astring> where std::string contains an unpredictable value.

Parameters

tableName	the name of the original table

Returns

<bool,string> see above description

Exceptions

SQLException	whenever something weird occurs with the mrdb connection

Implemented in minerule::TranslationManagerStandardSQL.

6.179.2.2 getOriginalColumnNameForColumn()

```
string minerule::TranslationManager::getOriginalColumnNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected]
```

It returns a column name of `getTranslationTableNameForColumn(tableName, columnName)`. In particular it returns the column that holds the original values.

Parameters

tableName	
columnName	

6.179.2.3 getTranslatedColumnNameForColumn()

```
string minerule::TranslationManager::getTranslatedColumnNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected]
```

It returns a column name of `getTranslationTableNameForColumn(tableName, columnName)`. In particular it returns the column that holds the translated values.

Parameters

tableName	
columnName	

6.179.2.4 getTranslationTableNameForColumn()

```
std::string minerule::TranslationManager::getTranslationTableNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected]
```

Once a column in a table get translated, a new table is created to hold the associations between its original values and their translations. The method returns the name of the translation table for column columnName of table table↩ Name.

Parameters

tableName	the original table name
columnName	a column of table "tableName"

6.179.2.5 getTranslationTableNameForTable()

```
string minerule::TranslationManager::getTranslationTableNameForTable (
            const std::string & tableName ) const  [protected]
```

It returns the name of the translated table associated to table tableName. This function does not access to the dictionary table to get the name and hence should be used only to know how the table name should look like (probably you want to use it when you create the translation table).

Parameters

tableName	

6.179.2.6 translateTable()

```
virtual TranslatedTable * minerule::TranslationManager::translateTable (
            const std::string & tableName ) const  [pure virtual]
```

If "tableName" has never been translated the function translates the values in "tableName", sets up any structure necessary to keep track of the mapping, and returns the object needed to access to the translated values, otherwise it simply returns the access object for the translated values. The function * throws a SQLException exception if something goes wrong (in particular if tableName does not exist in the current DB).

Note: is left to the user the duty of freeing the returned TranslatedTable object when it is no longer useful.

Parameters

tableName	the name of a table in the current DB

Returns

a TranslatedTable, that is an object allowing to access to the translated values.

Implemented in minerule::TranslationManagerStandardSQL.

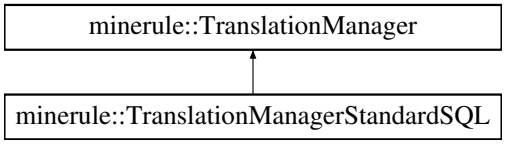
The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanager.hpp

6.180 minerule::TranslationManagerStandardSQL Class Reference

```
#include <translationmanagerstandardsql.hpp>
```

Inheritance diagram for minerule::TranslationManagerStandardSQL:



Public Member Functions

- TranslationManagerStandardSQL (mrdb::Connection *aConnection, bool translateEverythingOption=true)
- TranslatedTable * translateTable (const std::string &tableName) const
- pair< bool, string > alreadyTranslated (const std::string &tableName) const

Protected Member Functions

- std::string getTranslationTableNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getTranslatedColumnNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getOriginalColumnNameForColumn (const std::string &tableName, const std::string &columnName) const
- string getTranslationTableNameForTable (const std::string &tableName) const

Friends

- class TranslatedTableStandardSQL

6.180.1 Detailed Description

Definition at line 36 of file translationmanagerstandardsql.hpp.

6.180.2 Constructor & Destructor Documentation

6.180.2.1 TranslationManagerStandardSQL()

```
minerule::TranslationManagerStandardSQL::TranslationManagerStandardSQL (
            mrdb::Connection * aConnection,
            bool translateEverythingOption = true )
```

Constructor. The object will use the given connection for each and every access to the DB. Moreover the parameter translateEverythingOption will be used to determine when care has to be taken in choosing which attribute has to be translated. When this option is false the method shouldTranslateColumnsOfType should decide which columns types has to be translated.

6.180.3 Member Function Documentation

6.180.3.1 alreadyTranslated()

```
pair< bool, string > minerule::TranslationManagerStandardSQL::alreadyTranslated (
            const std::string & tableName ) const  [virtual]
```

Implements TranslationTable::alreadyTranslated using only standard SQL primitives.

Implements minerule::TranslationManager.

6.180.3.2 getOriginalColumnNameForColumn()

```
string minerule::TranslationManager::getOriginalColumnNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected], [inherited]
```

It returns a column name of `getTranslationTableNameForColumn(tableName, columnName)`. In particular it returns the column that holds the original values.

Parameters

tableName	
columnName	

6.180.3.3 getTranslatedColumnNameForColumn()

```
string minerule::TranslationManager::getTranslatedColumnNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected], [inherited]
```

It returns a column name of `getTranslationTableNameForColumn(tableName, columnName)`. In particular it returns the column that holds the translated values.

Parameters

tableName	
columnName	

6.180.3.4 getTranslationTableNameForColumn()

```
std::string minerule::TranslationManager::getTranslationTableNameForColumn (
            const std::string & tableName,
            const std::string & columnName ) const  [protected], [inherited]
```

Once a column in a table get translated, a new table is created to hold the associations between its original values and their translations. The method returns the name of the translation table for column columnName of table table↩ Name.

Parameters

tableName	the original table name
columnName	a column of table "tableName"

6.180.3.5 getTranslationTableNameForTable()

```
string minerule::TranslationManager::getTranslationTableNameForTable (
            const std::string & tableName ) const  [protected], [inherited]
```

It returns the name of the translated table associated to table tableName. This function does not access to the dictionary table to get the name and hence should be used only to know how the table name should look like (probably you want to use it when you create the translation table).

Parameters

tableName	

6.180.3.6 translateTable()

```
TranslatedTable * minerule::TranslationManagerStandardSQL::translateTable (
            const std::string & tableName ) const  [virtual]
```

Implements TranslationTable::translateTable using only standard SQL primitives.

Implements minerule::TranslationManager.

6.180.4 Friends And Related Function Documentation

6.180.4.1 TranslatedTableStandardSQL

```
friend class TranslatedTableStandardSQL  [friend]
```

Definition at line 37 of file translationmanagerstandardsql.hpp.

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanagerstandardsql.hpp

6.181 mrdb::Types Struct Reference

`#include <Types.hpp>`

Public Types

- enum SQLType {
  BIGINT = -5 , BINARY = -2 , BIT = -7 , CHAR = 1 ,
  DATE = 9 , DECIMAL = 3 , DOUBLE = 8 , FLOAT = 6 ,
  INTEGER = 4 , LONGVARBINARY = -4 , LONGVARCHAR = -1 , NUMERIC = 2 ,
  REAL = 7 , SMALLINT = 5 , TIME = 10 , TIMESTAMP = 11 ,
  TINYINT = -6 , VARBINARY = -3 , VARCHAR = 12 }

6.181.1 Detailed Description

Definition at line 8 of file Types.hpp.

6.181.2 Member Enumeration Documentation

6.181.2.1 SQLType

`enum mrdb::Types::SQLType`

Enumerator

BIGINT	
BINARY	
BIT	
CHAR	
DATE	
DECIMAL	
DOUBLE	
FLOAT	
INTEGER	
LONGVARBINARY	
LONGVARCHAR	
NUMERIC	
REAL	
SMALLINT	
TIME	
TIMESTAMP	
TINYINT	
VARBINARY	
VARCHAR	

Definition at line 9 of file Types.hpp.

```
00009                {
00010     BIGINT = -5,
00011     BINARY = -2,
00012     BIT = -7,
00013     CHAR = 1,
00014     DATE = 9,
00015     DECIMAL = 3,
00016     DOUBLE = 8,
00017     FLOAT = 6,
00018     INTEGER = 4,
00019     LONGVARBINARY = -4,
00020     LONGVARCHAR = -1,
00021     NUMERIC = 2,
00022     REAL = 7,
00023     SMALLINT = 5,
00024     TIME = 10,
00025     TIMESTAMP = 11,
00026     TINYINT = -6,
00027     VARBINARY = -3,
00028     VARCHAR = 12
00029   };
```

The documentation for this struct was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/mrdb/Types.hpp

6.182 minerule::VarSet Class Reference

`#include <VarSet.hpp>`

Public Member Functions

- VarSet (size_t numVars)
- VarSet ()
- void setSize (size_t numVars)
- void setVar (size_t varNum, bool value)
- bool getVar (size_t varNum) const
- bool operator++ (int)
- VarSet & operator= (const VarSet &v)
- bool operator< (const VarSet &v) const
- bool operator== (const VarSet &v) const
- bool operator> (const VarSet &v) const
- size_t size () const

6.182.1 Detailed Description

Definition at line 38 of file VarSet.hpp.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 VarSet() [1/2]

```
minerule::VarSet::VarSet (
            size_t numVars )  [inline]
```

Definition at line 45 of file VarSet.hpp.

```
00045                         :vset(0), vsetSize(numVars) {
00046     assert( vsetSize < sizeof(SetType)*8 );
00047   }
```

6.182.2.2 VarSet() [2/2]

```
minerule::VarSet::VarSet ( )  [inline]
```

Definition at line 49 of file VarSet.hpp.

```
00049           : vset(0), vsetSize(0) {
00050
00051   }
```

6.182.3 Member Function Documentation

6.182.3.1 getVar()

```
bool minerule::VarSet::getVar (
            size_t varNum ) const  [inline]
```

Definition at line 69 of file VarSet.hpp.

```
00069                                    {
00070     assert(varNum < vsetSize);
00071
00072     return (1<<varNum) & vset;
00073   }
```

6.182.3.2 operator++()

```
bool minerule::VarSet::operator++ (
            int  )  [inline]
```

Definition at line 83 of file VarSet.hpp.

```
00083                         {
00084     if(vset+1 < (unsigned int)(1<<vsetSize)) {
00085       vset++;
00086       return true;
00087     } else
00088       return false;
00089   }
```

6.182.3.3 operator<()

```
bool minerule::VarSet::operator< (
            const VarSet & v ) const  [inline]
```

Definition at line 97 of file VarSet.hpp.

```
00097                                       {
00098     assert(v.vsetSize==vsetSize);
00099     return vset<v.vset;
00100   }
```

6.182.3.4 operator=()

```
VarSet & minerule::VarSet::operator= (
            const VarSet & v )  [inline]
```

Definition at line 91 of file VarSet.hpp.

```
00091                                     {
00092     vset=v.vset;
00093     vsetSize=v.vsetSize;
00094     return *this;
00095   }
```

6.182.3.5 operator==()

```
bool minerule::VarSet::operator== (
            const VarSet & v ) const  [inline]
```

Definition at line 102 of file VarSet.hpp.

```
00102                                       {
00103     assert(v.vsetSize==vsetSize);
00104     return vset==v.vset;
00105   }
```

6.182.3.6 operator>()

```
bool minerule::VarSet::operator> (
            const VarSet & v ) const  [inline]
```

Definition at line 107 of file VarSet.hpp.

```
00107                                       {
00108     assert(v.vsetSize==vsetSize);
00109     return vset>v.vset;
00110   }
```

6.182.3.7 setSize()

```
void minerule::VarSet::setSize (
            size_t numVars )  [inline]
```

Definition at line 53 of file VarSet.hpp.

```
00053                                {
00054     assert( numVars < sizeof(SetType)*8 );
00055     vsetSize = numVars;
00056   }
```

6.182.3.8 setVar()

```
void minerule::VarSet::setVar (
            size_t varNum,
            bool value )  [inline]
```

Definition at line 58 of file VarSet.hpp.

```
00058                                          {
00059     assert(varNum < vsetSize);
00060     size_t mask = 1<<varNum;
00061
00062     if(value) {
00063       vset |= mask;
00064     } else {
00065       vset &= ~mask;
00066     }
00067   }
```

6.182.3.9 size()

```
size_t minerule::VarSet::size ( ) const  [inline]
```

Definition at line 112 of file VarSet.hpp.

```
00112                        {
00113     return vsetSize;
00114   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/VarSet.hpp

6.183 minerule::VarSetEnumerator Class Reference

`#include <VarSet.hpp>`

Public Member Functions

- VarSetEnumerator (size_t numVars)
- const VarSet & operator* () const
- bool operator++ (int)
- size_t enumerationSize (size_t val=0) const

6.183.1 Detailed Description

Definition at line 118 of file VarSet.hpp.

6.183.2 Constructor & Destructor Documentation

6.183.2.1 VarSetEnumerator()

```
minerule::VarSetEnumerator::VarSetEnumerator (
            size_t numVars )  [inline]
```

Definition at line 124 of file VarSet.hpp.

```
00124                                   : varset(numVars), firstCall(true) {
00125
00126   }
```

6.183.3 Member Function Documentation

6.183.3.1 enumerationSize()

```
size_t minerule::VarSetEnumerator::enumerationSize (
            size_t val = 0 ) const  [inline]
```

Definition at line 144 of file VarSet.hpp.

```
00144                                                {
00145     if( val==0 )
00146       val = varset.size();
00147
00148     return (size_t)(1«val);
00149   }
```

6.183.3.2 operator∗()

```
const VarSet & minerule::VarSetEnumerator::operator* ( ) const  [inline]
```

Definition at line 128 of file VarSet.hpp.

```
00128                                    {
00129     return varset;
00130   }
```

6.183.3.3 operator++()

```
bool minerule::VarSetEnumerator::operator++ (
            int  )  [inline]
```

Definition at line 132 of file VarSet.hpp.

```
00132                            {
00133     if(firstCall && varset.size()!=0 ) {
00134       firstCall=false;
00135       return true;
00136     }
00137     else
00138       return varset++;
00139   }
```

The documentation for this class was generated from the following file:

- /Users/esposito/Software/minerule/include/minerule/PredicateUtils/VarSet.hpp

Chapter 7

File Documentation

7.1 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ Algorithms.hpp File Reference

```
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Algorithms/AlgorithmsOptions.hpp"
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Optimizer/OptimizedMinerule.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
#include <exception>
```

Data Structures

- class minerule::Algorithms

Namespaces

- namespace minerule

7.2 Algorithms.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MINERULE_ALGORITHMS_H__
```

```
00017 #define __MINERULE_ALGORITHMS_H__
00018
00019 #include "minerule/Database/SourceRow.hpp"
00020 #include "minerule/Database/ItemType.hpp"
00021 #include "minerule/Algorithms/AlgorithmsOptions.hpp"
00022 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00023 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00024 #include "minerule/Utils/AlgorithmTypes.hpp"
00025 #include <exception>
00026
00027 namespace minerule {
00028
00029 class Algorithms {
00030  public:
00035   static MiningAlgorithmBase* newAlgorithm(const OptimizedMinerule& mr);
00036
00039   static void executeMinerule(OptimizedMinerule& mr) ;
00040
00042   static void executeExtractionAlgorithm(OptimizedMinerule& mr) ;
00043
00046   static bool executeIncrementalAlgorithm(OptimizedMinerule& mr) ;
00047
00049   static MiningAlgorithmBase* getBestRulesMiningAlgorithm(const OptimizedMinerule& mr);
00050
00052   static MiningAlgorithmBase* getBestItemsetsMiningAlgorithm(const OptimizedMinerule& mr);
00053
00055   static MiningAlgorithmBase* getBestSequencesMiningAlgorithm( const OptimizedMinerule& mr );
00056
00057 private:
00058         static void checkAndHandleHomonymMinerules(OptimizedMinerule& mr) ;
00059         static void showDebugInfo(const std::string& msg, OptimizedMinerule& mr);
00060 };
00061
00062 }
00063
00064
00065 #endif
```

7.3 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ AlgorithmsOptions.hpp File Reference

```
#include <string.h>
#include "minerule/mrdb/PreparedStatement.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
```

Data Structures

- class minerule::AlgorithmsOptions

Namespaces

- namespace minerule

7.4 AlgorithmsOptions.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __ALGORITHMS_OPTIONS_H__
00017 #define __ALGORITHMS_OPTIONS_H__
00018
00019 #include <string.h>
00020 #include "minerule/mrdb/PreparedStatement.hpp"
00021 #include "minerule/mrdb/Connection.hpp"
00022
00023 #include "minerule/Database/SourceRow.hpp"
00024 #include "minerule/Parsers/ParsedMinerule.hpp"
00025 #include "minerule/Database/MRResultSet.hpp"
00026 #include "minerule/Utils/MineruleOptions.hpp"
00027
00028 namespace minerule {
00029
00030 class AlgorithmsOptions {
00031 private:
00032   size_t totGroups; // total number of groups in the unfiltered table
00033   double support;
00034   double confidence;
00035   unsigned int rowsPerPartition;
00036   MinMaxPair headCardinalities;
00037   MinMaxPair bodyCardinalities;
00038
00039   // the following will substitute the prepared statement
00040   // when the MRResultSetIterator will be integrated in the proj.
00041   MRResultSetIterator *mriterator;
00042
00043   SourceRowColumnIds sourceRowDescription;
00044   std::string outTableName;
00045
00046   MineruleOptions::MiningAlgorithms miningAlgorithmsOptions;
00047
00048 public:
00049   // Note that headCardinalities and maxBodyCardinalities=1000
00050   // is far beyond the manageable sizes (i.e. it is the same
00051   // as setting them to infinity).
00052   AlgorithmsOptions()
00053       : headCardinalities(MinMaxPair(1, 1000)),
00054         bodyCardinalities(MinMaxPair(1, 1000)){};
00055
00058   void setSupport(double sup);
00059
00062   void setConfidence(double conf);
00063
00067   void setHeadCardinalities(const MinMaxPair &n) { headCardinalities = n; }
00068
00072   void setBodyCardinalities(const MinMaxPair &n) { bodyCardinalities = n; }
00073
00078   void setConnection(mrdb::Connection *connection);
00079
00085   void setStatement(mrdb::PreparedStatement *statement);
00086
00090   void setSourceRowDescription(const SourceRowColumnIds &srdes);
00091
00096   void setOutTableName(const std::string &fname) { outTableName = fname; }
00097
00101   void
00102   setMiningAlgorithmsOptions(const MineruleOptions::MiningAlgorithms &opt) {
00103     miningAlgorithmsOptions = opt;
00104   }
00105
00107   double getSupport() const;
00108
00110   double getConfidence() const;
00111
00113   const std::string &getOutTableName() const { return outTableName; }
00114
```

```
00116   const MineruleOptions::MiningAlgorithms &getMiningAlgorithmsOptions() const {
00117     return miningAlgorithmsOptions;
00118   }
00119
00122   const MinMaxPair &getHeadCardinalities() const { return headCardinalities; }
00123
00126   const MinMaxPair &getBodyCardinalities() const { return bodyCardinalities; }
00127
00131   const SourceRowColumnIds &getSourceRowDescription() const;
00132 };
00133
00134 } // end minerule namespace
00135
00136 #endif
```

7.5 /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsAndCross.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Database/Transaction.hpp"
```

Data Structures

- class minerule::BFSWithGidsAndCross

Namespaces

- namespace minerule

7.6 BFSWithGidsAndCross.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __BFSWITHGIDSANDCROSS_H__
00017 #define __BFSWITHGIDSANDCROSS_H__
00018
00019 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00020 #include "minerule/Database/Connection.hpp"
00021 #include "minerule/Database/Transaction.hpp"
00022
00023 namespace minerule {
00024
00025   class BFSWithGidsAndCross : public MiningAlgorithm {
00026
00027           typedef RuleTransaction< std::vector<std::pair<ItemType,ItemType> > > Transaction;
00028
00029
00030   //class MapElement : public std::set<ItemType> {
```

```
00031   class MapElement : public std::set<int> {
00032   public:
00033     int count;
00034     MapElement() : count(0) {}
00035   };
00036   typedef MapElement GidList;
00037
00038
00039   class BodyMapElement : public MapElement {
00040   public:
00041         std::pair<iterator, bool> insert(const value_type& x) { return std::set<int>::insert(x); }
00042     std::map<ItemType, MapElement > heads;
00043     void insert(const ItemType& item, MapElement& gidList, bool secondPass = false);
00044     bool pruneMap (float threshold);
00045     bool updateCount ();
00046   };
00047
00048   class NewRule {
00049   public:
00050           std::map<ItemType, BodyMapElement>::iterator lastBody;
00051           std::vector<ItemType> body, head;
00052           std::map<ItemType, MapElement> headTable;
00053           std::map<ItemType, MapElement>::iterator lastHead;
00054           double conf;
00055
00056           GidList gids;
00057           int bodySupp;
00058
00059           NewRule( const NewRule& r ) : lastBody(r.lastBody), body(r.body), head(r.head),
      headTable(r.headTable), conf(r.conf), gids(r.gids), bodySupp(r.bodySupp) {
00060                   if( r.lastHead == r.headTable.end() )
00061                           lastHead = headTable.end();
00062                   else
00063                           lastHead = headTable.find( r.lastHead->first );
00064           }
00065
00066           NewRule (std::map<ItemType, BodyMapElement>::iterator b, GidList& g) : lastBody(b), gids(g)
      {
00067                   body.push_back(b->first);
00068                   headTable = b->second.heads;
00069                   lastHead = headTable.begin();
00070           }
00071
00072           NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList& g) :
      body(r.body), lastBody(b), gids(g) {
00073                   body.push_back(b->first);
00074                   headTable = hash_intersection( r.headTable, b->second.heads );
00075                   lastHead = headTable.begin();
00076           }
00077
00078           NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList& g, int bs) :
      body(r.body), head(r.head), headTable(r.headTable), lastBody(b), lastHead(r.lastHead), gids(g),
      bodySupp(bs) {
00079                   body.push_back(b->first);
00080           }
00081
00082           NewRule (NewRule& r, std::map<ItemType, MapElement>::iterator h, GidList& g) : body(r.body),
      head(r.head), headTable(r.headTable), lastBody(r.lastBody), gids(g), bodySupp(r.bodySupp) {
00083                   head.push_back(h->first);
00084                   lastHead = headTable.find( h->first );
00085           }
00086
00087           friend std::ostream& operator«(std::ostream& out, NewRule r) {
00088                   std::vector<ItemType>::iterator i;
00089                   for (i=r.body.begin(); i!=r.body.end(); i++) {
00090                           if (i != r.body.begin()) out « ", ";
00091                           out « *i;
00092                   }
00093                   out « " -> ";
00094                   for (i=r.head.begin(); i!=r.head.end(); i++) {
00095                           if (i != r.head.begin()) out « ", ";
00096                           out « *i;
00097                   }
00098                   out « std::endl;
00099                   return out;
00100           }
00101
00102    private:
00103           static std::map<ItemType, MapElement> hash_intersection( std::map<ItemType, MapElement>&
      lhs, const std::map<ItemType, MapElement>& rhs );
00104    };
00105
00106   class NewRuleSet : public std::vector<NewRule> {  };
00107
00108   class BodyMap : public std::map<ItemType, BodyMapElement> {
00109     Connection* connection;
00110     void insertRules( const NewRuleSet& rs, double totGroups );
```

```
00111     Progress* progress;
00112   public:
00113     BodyMap(Connection& cc, Progress& progr) : connection(&cc), progress(&progr) {};
00114
00115     int add(int gid, Transaction& t1, bool secondPass = false);
00116     int add(int gid, mrdb::ResultSet* rs, SourceRow& hbsr, bool secondPass = false);
00117
00118     void pruneMap(float threshold);
00119     void updateCount ();
00120     // void createBodies (NewRuleSet& rs, float threshold, size_t maxBody);
00121     // void createHeads (NewRuleSet& rs, NewRuleSet& rs1, float threshold, size_t maxHead);
00122
00123     bool checkHeads (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList& g, float
      threshold);
00124     int generateRules (float support, int totGroups, int maxBody, int maxHead);
00125     void addHead (NewRuleSet& rs, float threshold, int maxHead, int suppBody, NewRule& rc);
00126
00127   };
00128   private:
00129     SourceRowColumnIds rowDes;
00130         static bool mineruleHasSameBodyHead;
00131
00132         SourceTable* sourceTable;
00133         SourceTable::Iterator ruleIterator;
00134
00135
00136     void insertRules( const NewRuleSet& rs, double totGroups );
00137
00138     void prepareData();
00139
00140   public:
00141     BFSWithGidsAndCross(const OptimizedMinerule& mr) :
00142       MiningAlgorithm(mr), sourceTable(NULL) {}
00143
00144     virtual ~BFSWithGidsAndCross() {
00145         if(sourceTable!=NULL) delete sourceTable;
00146     }
00147
00148                 virtual SourceTableRequirements sourceTableRequirements() const {
00149                         return SourceTableRequirements(SourceTableRequirements::CrossProduct |
      SourceTableRequirements::SortedGids);
00150                 };
00151
00152         static bool getMineruleHasSameBodyHead() {
00153                 return mineruleHasSameBodyHead;
00154         }
00155
00156     virtual void mineRules();
00157
00158     virtual bool canHandleMinerule() const {
00159       return !minerule.getParsedMinerule().requiresClusters() &&
      !minerule.getParsedMinerule().hasDisjuctionsInMC();
00160     }
00161
00162   };
00163 }
00164
00165
00166
00167 #endif
```

7.7 /Users/esposito/Software/minerule/include/minerule/Algorithms/BFSWithGidsNoCross.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Database/SourceTable.hpp"
#include "minerule/Database/Transaction.hpp"
```

Data Structures

- class minerule::BFSWithGidsNoCross

Namespaces

- namespace minerule

7.8 BFSWithGidsNoCross.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __BFSWITHGIDSNOCROSS_H__
00017 #define __BFSWITHGIDSNOCROSS_H__
00018
00019 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00020 #include "minerule/Database/Connection.hpp"
00021 #include "minerule/Database/SourceTable.hpp"
00022 #include "minerule/Database/Transaction.hpp"
00023
00024 namespace minerule {
00025
00026         class BFSWithGidsNoCross : public MiningAlgorithm {
00027
00028                 typedef ItemTransaction< std::set<ItemType> > Transaction;
00029
00030                 class MapElement : public std::set<int> {
00031                 public:
00032                         int count;
00033                         MapElement() : count(0) {}
00034                 };
00035
00036                 typedef MapElement GidList;
00037
00038                 class BodyMapElement : public MapElement {
00039                 public:
00040     //pair<iterator, bool> insert(const value_type& x) { return std::set<ItemType>::insert(x); }
00041                         std::pair<iterator, bool> insert(const value_type& x) { return
       std::set<int>::insert(x); }
00042                         std::map<ItemType, MapElement > heads;
00043                         void insert(const ItemType& item, MapElement& gidList, bool secondPass =
       false);
00044                         bool pruneMap (float threshold);
00045                         bool updateCount ();
00046                 };
00047
00048                 class NewRule {
00049                 public:
00050                         std::vector<ItemType> body, head;
00051                         std::map<ItemType, BodyMapElement>::iterator lastBody;
00052                         std::map<ItemType, MapElement>::iterator lastHead;
00053                         double conf;
00054
00055                         GidList gids;
00056                         int bodySupp;
00057
00058                         NewRule (std::map<ItemType, BodyMapElement>::iterator b, GidList& g) :
       lastBody(b), gids(g) {
00059                                 body.push_back(b->first);
00060                                 lastHead = b->second.heads.begin();
00061                         }
00062                         NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList&
       g) : body(r.body), lastBody(b), gids(g) {
00063                                 body.push_back(b->first);
00064                                 lastHead = b->second.heads.begin();
00065                         }
00066                         NewRule (std::map<ItemType, BodyMapElement>::iterator b, std::map<ItemType,
       MapElement>::iterator h, GidList& g, int bs) : lastBody(b), lastHead(h), gids(g), bodySupp(bs) {
00067                                 body.push_back(b->first);
00068                                 head.push_back(h->first);
```

```
00069                         }
00070                         NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList&
    g, int bs) : body(r.body), head(r.head), lastBody(b), lastHead(r.lastHead), gids(g), bodySupp(bs) {
00071                                 body.push_back(b->first);
00072                         }
00073                         NewRule (NewRule& r, std::map<ItemType, MapElement>::iterator h, GidList& g) :
    body(r.body), head(r.head), lastBody(r.lastBody), lastHead(h), gids(g), bodySupp(r.bodySupp) {
00074                                 head.push_back(h->first);
00075                         }
00076                         friend std::ostream& operator<<(std::ostream& out, NewRule r) {
00077                                 std::vector<ItemType>::iterator i;
00078                                 for (i=r.body.begin(); i!=r.body.end(); i++) {
00079                                         if (i != r.body.begin()) out << ", ";
00080                                         out << *i;
00081                                 }
00082                                 out << " -> ";
00083                                 for (i=r.head.begin(); i!=r.head.end(); i++) {
00084                                         if (i != r.head.begin()) out << ", ";
00085                                         out << *i;
00086                                 }
00087                                 out << std::endl;
00088                                 return out;
00089                         }
00090                 };
00091
00092                 class NewRuleSet : public std::vector<NewRule> {};
00093
00094                 class BodyMap : public std::map<ItemType, BodyMapElement> {
00095                         Connection* connection;
00096                         Progress* progress;
00097                         void insertRules( const NewRuleSet& rs, double totGroups );
00098                 public:
00099                         BodyMap(Connection& cc, Progress& prog) : connection(&cc), progress(&prog) {};
00100
00101    //int add(ItemType& gid, Transaction& t1, Transaction& t2, bool secondPass = false);
00102                         int add(int gid, Transaction& t1, Transaction& t2, bool secondPass = false);
00103    //  void saveMap(std::ostream& out, bool withGids = true);
00104    //  void loadMap(istream& in, bool withGids = true);
00105                         void pruneMap(float threshold);
00106                         void updateCount ();
00107                         void createBodies (NewRuleSet& rs, float threshold, size_t maxBody);
00108                         void createHeads (NewRuleSet& rs, NewRuleSet& rs1, float threshold, size_t
    maxHead);
00109    //void generateRules (NewRuleSet& rs, float threshold, int maxBody, int maxHead);
00110                         int generateRules (float support, int totGroups, int maxBody, int maxHead);
00111                         void addHead (NewRuleSet& rs, float threshold, int maxHead, int suppBody,
    NewRule& rc);
00112
00113                 };
00114         private:
00115                 static bool mineruleHasSameBodyHead;
00116                 SourceTable* sourceTable;
00117
00118                 SourceTable::Iterator bodyIterator;
00119                 SourceTable::Iterator headIterator;
00120
00121                 void insertRules( const NewRuleSet& rs, double totGroups );
00122
00123                 void prepareData();
00124         public:
00125                 BFSWithGidsNoCross(const OptimizedMinerule& mr) : MiningAlgorithm(mr),
    sourceTable(NULL) {
00126                         mineruleHasSameBodyHead = mr.getParsedMinerule().hasSameBodyHead();
00127                 }
00128
00129                 virtual ~BFSWithGidsNoCross() {
00130                         if(sourceTable!=NULL) delete sourceTable;
00131                 }
00132
00133                 virtual SourceTableRequirements sourceTableRequirements() const {
00134                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00135                 };
00136
00137
00138                 virtual void mineRules();
00139
00140                 static bool getMineruleHasSameBodyHead() {
00141                         return mineruleHasSameBodyHead;
00142                 }
00143
00144                 virtual bool canHandleMinerule() const {
00145                         return
00146                                 !minerule.getParsedMinerule().hasCrossConditions() &&
00147                                 !minerule.getParsedMinerule().requiresClusters() &&
00148                                 !minerule.getParsedMinerule().hasDisjuctionsInMC();
00149                 }
00150
```

```
00151         };
00152 }
00153
00154
00155
00156 #endif
```

7.9 /Users/esposito/Software/minerule/include/minerule/Algorithms/BitVector.hpp File Reference

```
#include <vector>
#include <cstring>
```

Data Structures

- class minerule::BitVector

Namespaces

- namespace minerule

7.10 BitVector.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef BIT_VECTOR__H
00017 #define BIT_VECTOR__H
00018 #include <vector>
00019 #include <cstring>
00020
00021 namespace minerule {
00022         class BitVector: public std::vector<bool>{
00023
00024         public:
00025
00026
00027                 BitVector operator& (BitVector input){
00028                         BitVector ris;
00029                         for (size_t i=0;i<this->size();++i)
00030                                 ris.push_back((*this)[i]&input[i]);
00031                         return ris;
00032                 }
00033
00034                 BitVector operator| (BitVector input){
00035                         BitVector ris;
00036                         for (size_t i=0;i<this->size();++i)
00037                                 ris.push_back((*this)[i]|input[i]);
00038                         return ris;
00039                 }
00040
```

```
00041
00042         };
00043
00044 }
00045
00046
00047 #endif
00048
```

7.11 /Users/esposito/Software/minerule/include/minerule/Algorithms/Bodymap.hpp File Reference

```
#include <limits.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <iterator>
#include <algorithm>
#include "minerule/Algorithms/AlgorithmsOptions.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Database/Transaction.hpp"
#include "minerule/Utils/Bitstring.hpp"
```

Data Structures

- class minerule::Transaction
- class minerule::MapElement
- class minerule::MinMax
- class minerule::BodyMapElement
- class minerule::NewRule
- class minerule::NewRuleSet
- class minerule::BodyMap

Namespaces

- namespace minerule

Typedefs

- typedef MapElement minerule::GidList

7.12 Bodymap.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __BODYMAP_H
00017 #define __BODYMAP_H
00018
00019 #include<limits.h>
00020 #include<stdio.h>
00021 #include<string.h>
00022 #include<stdlib.h>
00023 #include <iostream>
00024 #include <fstream>
00025 #include <iterator>
00026 #include <algorithm>
00027
00028 #include "minerule/Algorithms/AlgorithmsOptions.hpp"
00029 #include "minerule/Utils/MineruleOptions.hpp"
00030 #include "minerule/Database/MRResultSet.hpp"
00031 #include "minerule/Database/ItemType.hpp"
00032 #include "minerule/Utils/Converter.hpp"
00033 #include "minerule/Database/Connection.hpp"
00034 #include "minerule/Database/Transaction.hpp"
00035 #include "minerule/Utils/Bitstring.hpp"
00036
00037 namespace minerule {
00038
00039         class AggregateMonoConstraint;
00040         class AggregateAntiMonoConstraint;
00041         class NewRule;
00042
00043         class Transaction : public ItemTransaction< std::set<ItemType> >  {
00044         public:
00045                 std::vector<int> values;
00046         };
00047
00048 //class MapElement : public std::set<ItemType> {
00049         class MapElement : public BitString {
00050         public:
00051                 int counter;
00052                 void insert(const int x) { set(x,true); }
00053                 MapElement() : counter(0) {}
00054                 void operator=(const BitString& bs) { BitString::operator=(bs); }
00055         };
00056         typedef MapElement GidList;
00057
00058         class MinMax {
00059         public:
00060                 int min;
00061                 int max;
00062                 MinMax () : min(INT_MAX), max(0) {}
00063                 MinMax (int m1, int m2) : min(m1), max(m2) {}
00064                 int minValue() { return min; }
00065                 int maxValue() { return max; }
00066         };
00067
00068 //class BodyMapElement : public std::set<ItemType> {
00069         class BodyMapElement : public MapElement {
00070         public:
00071                 std::vector<MinMax> attribute;
00072                 bool done;
00073                 BodyMapElement() { done = false; }
00074                 BodyMapElement(const BodyMapElement& bm) {
00075                         done = bm.done;
00076                         for (size_t i=0; i<bm.attribute.size(); i++)
       attribute.insert(attribute.end(),bm.attribute[i]);
00077                         BitString::operator=((BitString&)bm);
00078                 }
00079         //pair<iterator, bool> insert(const value_type& x) { return std::set<ItemType>::insert(x); }
00080                 void insert(const int x) { set(x,true); }
00081                 std::map<ItemType, MapElement > heads;
```

```
00082                 void insert(const ItemType& item, const int gid, bool secondPass = false);
00083                 bool pruneMap(double threshold, bool onlyBody = false);
00084                 bool updateCount ();
00085 //      void saveMap (std::ostream& out, bool withGids = true);
00086 //      void loadMap (istream& in, bool withGids = true);
00087                 void setMinMax (int which, int v);
00088         };
00089
00090         class NewRule {
00091         public:
00092                 std::vector<ItemType> body, head;
00093                 std::map<ItemType, BodyMapElement>::iterator lastBody;
00094                 std::map<ItemType, MapElement>::iterator lastHead;
00095                 bool satisfy;
00096                 GidList gids;
00097                 int bodySupp;
00098                 NewRule () : satisfy(false) {}
00099                 NewRule (std::map<ItemType, BodyMapElement>::iterator b, GidList& g) : lastBody(b),
        satisfy(false), gids(g) {
00100                         body.push_back(b->first);
00101                         lastHead = b->second.heads.begin();
00102                 }
00103                 NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList& g) :
        body(r.body), lastBody(b), satisfy(r.satisfy), gids(g) {
00104                         body.push_back(b->first);
00105                         lastHead = b->second.heads.begin();
00106                 }
00107                 NewRule (std::map<ItemType, BodyMapElement>::iterator b, std::map<ItemType,
        MapElement>::iterator h, GidList& g, int bs) : lastBody(b), lastHead(h), satisfy(false), gids(g),
        bodySupp(bs) {
00108                         body.push_back(b->first);
00109                         head.push_back(h->first);
00110                 }
00111                 NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, GidList& g, int
        bs) : body(r.body), head(r.head), lastBody(b), lastHead(r.lastHead), satisfy(r.satisfy), gids(g),
        bodySupp(bs) {
00112                         body.push_back(b->first);
00113                 }
00114                 NewRule (NewRule& r, std::map<ItemType, BodyMapElement>::iterator b, int bs, GidList&
        g) : body(r.body), head(r.head), lastBody(b), lastHead(r.lastHead), satisfy(r.satisfy), gids(g),
        bodySupp(bs) {
00115                         head.push_back(b->first);
00116                 }
00117                 NewRule (NewRule& r, std::map<ItemType, MapElement>::iterator h, GidList& g) :
        body(r.body), head(r.head), lastBody(r.lastBody), lastHead(h), satisfy(r.satisfy), gids(g),
        bodySupp(r.bodySupp) {
00118                         head.push_back(h->first);
00119                 }
00120                 friend std::ostream& operator<<(std::ostream& out, NewRule r) {
00121                         std::vector<ItemType>::iterator i;
00122                         for (i=r.body.begin(); i!=r.body.end(); i++) {
00123                                 if (i != r.body.begin()) out << ", ";
00124                                 out << *i;
00125                         }
00126                         out << " -> ";
00127                         for (i=r.head.begin(); i!=r.head.end(); i++) {
00128                                 if (i != r.head.begin()) out << ", ";
00129                                 out << *i;
00130                         }
00131                 //out << std::endl;
00132                         return out;
00133                 }
00134         };
00135
00136         class NewRuleSet : public std::vector<NewRule> {};
00137
00138         class BodyMap : public std::map<ItemType, BodyMapElement> {
00139                 static int nextid;
00140                 std::string outfile;
00141                 Connection * connection;
00142                 double totGroups;
00143                 std::ofstream outR;
00144                 std::ofstream outHB;
00145 // std::vector<GidList> bodySupports;
00146                 std::vector<int> itemsets;
00147 //      int curBodySupp;
00148         public:
00149                 int nextID() { return nextid++; }
00150                 BodyMap(std::string f, double totG) : outfile(f), totGroups(totG) {};
00151                 BodyMap(Connection& cc, double totG) : connection(&cc), totGroups(totG) {};
00152                 void openOutputFiles() {
00153                         std::string filename = outfile + ".r";
00154                         outR.open(filename.c_str());
00155                         std::string filename1 = outfile + ".hb";
00156                         outHB.open(filename1.c_str());
00157                 }
00158                 void closeOutputFiles() { outR.close(); outHB.close(); }
```

```
00159                 void setTotalGroups(double totG) { totGroups = totG; }
00160                 std::vector<AggregateMonoConstraint *> monoConstr;
00161                 std::vector<AggregateAntiMonoConstraint *> antiMonoConstr;
00162                 bool checkAntiMono (NewRule& rc);
00163                 bool checkMono (NewRule& rc);
00164                 int add(const int gid, Transaction& t1, Transaction& t2, bool secondPass = false);
00165                 int add(Transaction& t1, const int gid);
00166                 void pruneMap(double threshold, bool onlyBody = false);
00167                 void updateCount ();
00168                 int buildItemset (NewRuleSet& rs, NewRule& rc, std::vector<BodyMap::iterator>& v, int
      maxCard, float threshold, int j);
00169                 int generateStartItemSets (NewRuleSet& rs, NewRule& rc,
      std::vector<BodyMap::iterator>& v, int maxCard, float threshold, int j);
00170                 int buildRules (NewRuleSet& rs2, NewRule rc, BodyMap::iterator p, float threshold, int
      maxBody, int maxHead);
00171                 int buildHead (NewRuleSet& rs, float threshold, int maxHead, int suppBody, NewRule&
      rc, std::map<ItemType, MapElement>::iterator j);
00172                 int buildRules (NewRuleSet& rs, NewRule& rc, std::vector<BodyMap::iterator>& vb,
      std::vector<BodyMap::iterator>& vh, int maxBodyCard, int maxHeadCard, float threshold, int j);
00173                 std::vector<BodyMap::iterator> buildHead (NewRuleSet& rs1, NewRule rc,
      std::vector<BodyMap::iterator>& v, int maxCard, float threshold, int j);
00174                 void howManyItemsets();
00175                 void saveRules( const NewRuleSet& rs, double bodySupp);
00176                 void saveItemsets( const NewRuleSet& rs, double bodySupp);
00177                 void saveItemset( const NewRule& r, double bodySupp);
00178         };
00179
00180 } // end namespace
00181
00182 #endif
```

7.13 /Users/esposito/Software/minerule/include/minerule/Algorithms/Care.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Algorithms/Bodymap.hpp"
```

Data Structures

- class minerule::CARE

Namespaces

- namespace minerule

7.14 Care.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __CARE_H
00017 #define __CARE_H
00018
00019 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00020 #include "minerule/Database/Connection.hpp"
00021 #include "minerule/Algorithms/Bodymap.hpp"
00022
00023 namespace minerule {
00024
00025   class CARE : public MiningAlgorithm {
00026   private:
00027         SourceTable* sourceTable;
00028         SourceTable::Iterator bodyIterator;
00029         SourceTable::Iterator headIterator;
00030
00031
00032     size_t buildAttrStr(const ParsedMinerule::AttrVector& attr,
00033                         size_t startIndex,
00034                         std::string& attrStr,
00035                         std::vector<int>& des) const;
00036
00037     std::string buildQry( const std::string& groupAttrStr,
00038                           const std::string& attrStr,
00039                           const std::string& constraints) const;
00040
00041     void prepareData();
00042
00043   public:
00044     CARE(const OptimizedMinerule& mr) : MiningAlgorithm(mr) {}
00045
00046     virtual ~CARE() {}
00047
00048     virtual void mineRules();
00049
00050     virtual bool canHandleMinerule() const {
00051       return
00052         !minerule.getParsedMinerule().hasCrossConditions() &&
00053         !minerule.getParsedMinerule().requiresClusters() &&
00054         !minerule.getParsedMinerule().hasDisjuctionsInMC();
00055     }
00056
00057   };
00058 }
00059
00060
00061
00062 #endif
00063
```

7.15 /Users/esposito/Software/minerule/include/minerule/Algorithms/CCSMiner.hpp File Reference

```
#include "minerule/Algorithms/CCSMSequence.hpp"
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
```

Data Structures

- class minerule::CCSMiner

Namespaces

- namespace minerule

7.16 CCSMiner.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __CCSMINER_H__
00017 #define __CCSMINER_H__
00018
00019 #include "minerule/Algorithms/CCSMSequence.hpp"
00020 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00021 #include "minerule/Database/Connection.hpp"
00022 #include "minerule/Utils/AlgorithmTypes.hpp"
00023
00024 namespace minerule {
00025
00026   class CCSMiner : public MiningAlgorithm {
00027   private:
00028     AlgorithmsOptions options;
00029     Connection connection;
00030     SourceRowColumnIds rowDes;
00031     mrdb::PreparedStatement* statement;
00032
00033     void prepareData();
00034
00035   public:
00036     CCSMiner(const OptimizedMinerule& mr, const AlgorithmsOptions& opts) :
00037       MiningAlgorithm(mr),  statement(NULL) {
00038         options = opts;
00039         connection.useMRDBConnection(connection.getMRDBConnection());
00040 }
00041
00042     virtual ~CCSMiner() {}
00043
00044     virtual void mineRules();
00045
00046     virtual bool canHandleMinerule() const {
00047       return minerule.getParsedMinerule().miningTask == MTMineSequences;
00048     }
00049
00050                 virtual SourceTableRequirements sourceTableRequirements() const {
00051                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00052                 };
00053
00054     bool find(std::vector<CCSMSequence*>*vec, CCSMSequence* elem);
00055
00056     void combina(std::vector<CCSMSequence::ResultItems>&, std::vector<CCSMSequence*>* k2, size_t k,
      int min_g, int max_g, double threshold, int check);
00057
00058   };
00059 }
00060
00061
00062
00063 #endif
```

7.17 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ CCSMSequence.hpp File Reference

```
#include "BitVector.hpp"
#include <list>
#include <string>
#include <iostream>
#include "minerule/Database/ItemType.hpp"
```

```
#include "minerule/Database/SourceRowColumnIds.hpp"
```

Data Structures

- class minerule::CCSMSequence

Namespaces

- namespace minerule

Macros

- #define PRESENT 1
- #define ABSENT 0

7.17.1 Macro Definition Documentation

7.17.1.1 ABSENT

```
#define ABSENT 0
```

Definition at line 27 of file CCSMSequence.hpp.

7.17.1.2 PRESENT

```
#define PRESENT 1
```

Definition at line 26 of file CCSMSequence.hpp.

7.18 CCSMSequence.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef CCSMSequence__H
00017 #define CCSMSequence__H
00018 #include "BitVector.hpp"
00019
00020 #include <list>
00021 #include <string>
00022 #include <iostream>
00023 #include "minerule/Database/ItemType.hpp"
00024 #include "minerule/Database/SourceRowColumnIds.hpp"
00025
00026 #define PRESENT 1
00027 #define ABSENT 0
00028
00029
00030 namespace minerule{
00031
00032 class CCSMSequence {
00033
00034
00035 public:
00036         typedef std::vector<std::pair<int,int> > Eid_List;
00037
00038 private:
00039         bool singleton;
00040         //typedef std::vector<std::pair<int,int> > Eid_List;
00041         BitVector listSid;
00042         typedef std::vector<ItemType> List_Type;
00043         List_Type* seq;
00044         int count;
00045         std::vector<Eid_List*> eid_vector;
00046
00047         //puntatore al singleton presente in unltima posizione di questa sequenza
00048         //utile per velocizzare la generazione di una sequenza di lunghezza k da due sequenze di
       lunghezza k-1 che condividono k-2 elementi
00049         // a-b-c e b-c-d condividono b-c e da queste due genererò a-b-c-d come intersezione di a-b-c e
       d, L(a-b-c) intersect L(d)
00050         //appunto d è l'ultimo singleton di b-c-d
00051         CCSMSequence* last;
00052         std::list<CCSMSequence*> prefix;
00053         std::list<CCSMSequence*> suffix;
00054
00055         static std::vector<std::pair<int,int> >* mergeEidRtoLeft(const Eid_List& first, const
       Eid_List& second, int min_gap ,int max_gap);
00056
00057 public:
00058
00059         typedef std::pair<std::vector<ItemType>,int> ResultItems;
00060
00061         CCSMSequence(CCSMSequence* l=NULL) {
00062                 singleton=false;
00063                 count=0;
00064                 seq = new List_Type();
00065                 last=l;
00066                 listSid.reserve(1000000);
00067                 eid_vector.reserve(1000000);
00068                 /*cout<<"bit_vector.capacity "<<listSid.capacity()<<std::endl;
00069                 std::cout<<"bit_vector.max_size "<<listSid.max_size()<<std::endl;
00070                                 std::cout<<"eid_vector.capacity "<<eid_vector.capacity()<<std::endl;
00071                 std::cout<<"eid_vector.max_size "<<eid_vector.max_size()<<std::endl;
00072                 getchar();*/
00073         };
00074
00075         CCSMSequence(const CCSMSequence &in, size_t start, size_t n_item, CCSMSequence* l=NULL){
00076                 seq = new List_Type();
00077                 if (n_item==1){
00078                         seq->push_back((*in.seq)[start]);
00079                 }
```

```
00080                 else {
00081                         List_Type::iterator it = in.seq->begin();
00082                         for (size_t i=0;i<start;++i){
00083                                 ++it;
00084                         }
00085                         for (size_t i=start;(i<in.seq->size())&&(i<start+n_item);++i){
00086                                 seq->push_back(*it);
00087                                 ++it;
00088                         }
00089                 }
00090                 count=0;
00091                 if (seq->size()==1)
00092                         singleton=true;
00093                 else singleton=false;
00094                 last=1;
00095         }
00096
00097         CCSMSequence(const CCSMSequence &in){
00098                 seq= new List_Type();
00099                 *seq=*in.seq;
00100                 count=0;
00101                 if (seq->size()==1)
00102                         singleton=true;
00103                 else singleton=false;
00104                 listSid = in.listSid;
00105                 last = in.last;
00106                 count=in.count;
00107         }
00108
00109         void reduction(std::vector<size_t>& k){
00110                 //elimino dalla lista degli Eventi gli eventi corrispondenti alle transazioni che non
      sono utili al fine del conteggio
00111                 //delle frequenze del livello a cui siamo arrivati in modo da migliorare
00112                 //le performance dell'algoritmo e l'occupazione di memoria
00113
00114                 //time_t init,end;
00115                 //time(&init);
00116          std::vector<Eid_List*>::iterator it;
00117                 size_t i =0;
00118                 size_t j=0;
00119                 it = eid_vector.begin();
00120                 //size_t s=eid_vector.size();
00121          std::vector<Eid_List*> eid_temp;
00122
00123                 for (it=eid_vector.begin();it!=eid_vector.end();++it,i++){
00124                         if (k[j]!=i)
00125                                 eid_temp.push_back((*it));
00126                         else {
00127                                 delete (*it);
00128
00129                                 j++;
00130                             }
00131
00132                 }
00133                 /*for (it = eid_canc.begin();it!=eid_canc.end();++it)
00134                                 delete (*it);
00135                 */
00136                 eid_vector=eid_temp;
00137                 //elimino dalla bitmap le sequenze che non sono utili al fine del conteggio delle
      frequenze delle sequenze, in modo da migliorare
00138                 //le performance dell'algoritmo e l'occupazione di memoria
00139                 //time(&init);
00140                 i=0;
00141                 BitVector ris;
00142                 for (size_t f =0;f<listSid.size();++f){
00143                         if (f!=k[i]){
00144                                 ris.push_back(listSid[f]);
00145                         }
00146                         else i++;
00147                 }
00148                 //time(&end);
00149                 //cout«"ridotta listSid in tempo "«difftime(end,init)«std::endl;
00150                 listSid=ris;
00151
00152         }
00153
00154
00155         //      void read(const std::string& s);
00156
00157         size_t size() {
00158                 return seq->size();
00159         };
00160
00161         bool operator<(const CCSMSequence& in);
00162
00163         bool operator==(const CCSMSequence& in);
00164
```

```
00165         void add(const ItemType& s){
00166                 seq->push_back(s);
00167                 if (seq->size()==1)
00168                         singleton=true;
00169                 else singleton=false;
00170         }
00171
00172         std::string toStdString(){
00173                 std::string ris="";
00174                 List_Type::const_iterator it = seq->begin();
00175                 for (;it!=seq->end();++it)
00176                         ris=ris+(*it).asString()+" ";
00177                 return ris;
00178         }
00179
00180         void setPresent(size_t sid, bool isHere){
00181
00182                 if (sid<listSid.size())
00183                         return;
00184         //questo serve per riempire l'ultimo elemento del vettore degli
00185         //Eid_List per quegli elementi che non sono presenti in ultima posizione
00186                 if (sid>listSid.size()){
00187                         for (size_t i=listSid.size();i < sid;i++){
00188                                 listSid.push_back(ABSENT);
00189                                 eid_vector.push_back(new Eid_List());
00190                         }
00191                 }
00192                 listSid.push_back(isHere);
00193                 eid_vector.push_back(new Eid_List());
00194                 if (isHere)
00195                         count++;
00196         }
00197
00198         void setEid(size_t sid, size_t eid){
00199                 //inserisco la posizione dell occorrenza (eventIdentifier)
00200                 //del singleton per
00201                 // questo determinato sid sequenceIdentifier
00202                 std::pair<int,int> couple(eid,eid);
00203                 if (eid_vector.size() == sid)
00204                         eid_vector.push_back(new Eid_List());
00205                 eid_vector[sid]->push_back(couple);
00206
00207         }
00208
00209         void stampaEid(size_t sid){
00210         std::string ris="";
00211                 Eid_List* temp = eid_vector[sid];
00212                 int a, b;
00213                 for (size_t i=0;i<temp->size();++i){
00214                         a=(*temp)[i].first;
00215                         b=(*temp)[i].second;
00216                         std::cout«"   ("«a«","«b«")";
00217                 }
00218                 std::cout«std::endl;
00219
00220         }
00221
00222         void bitMaptoString(){
00223         std::string ris ="";
00224                 for (size_t i=0;i<listSid.size();++i){
00225                         std::cout«listSid[i]«std::endl;
00226                         stampaEid(i);
00227                 }
00228         }
00229
00230         void svuota(){
00231                 delete seq;
00232                 seq = new List_Type();
00233                 }
00234
00235         ~CCSMSequence() {
00236                 delete seq;
00237          std::vector<Eid_List*>::iterator it_list;
00238                 for (it_list=eid_vector.begin();it_list!=eid_vector.end();++it_list)
00239                         delete (*it_list);
00240
00241         };
00242
00243         int getCount(){
00244                 return count;
00245         }
00246
00247         bool isSingleton(){
00248                 return singleton;
00249         }
00250         /* La sequenza first e una sequenza di lunghezza >=1 mentre la sequenza second è una sequenza
      di lunghezza = 1
```

```
00251          *
00252          *
00253          */
00254         static CCSMSequence* merge(CCSMSequence* first, CCSMSequence* second, int min_gap,int max_gap,
      double threshold);
00255
00256         static std::vector<size_t> reduce_tr(std::vector<CCSMSequence*>* vec){
00257                 //cout«"ENTRO NELLA REDUCE_TR"«std::endl;
00258                 time_t init,end;
00259          std::vector<size_t> k;
00260          std::vector<CCSMSequence*>::iterator it;
00261                 BitVector ris;
00262                 if (vec->size()==0)
00263                         return k;
00264                 it=vec->begin();
00265                 //cout«"ecco le sequenze su cui applico la reduce"«std::endl;
00266                 //cout«(*it)->toStdString()«std::endl;
00267                 ris=(*it)->listSid;
00268
00269                 ++it;
00270                 time(&init);
00271                 for (;it!=vec->end();++it){
00272                         //cout«(*it)->toStdString()«std::endl;
00273                         ris=ris|(*it)->listSid;
00274                 }
00275                 time(&end);
00276
      //cout«"----------"«vec->size()«"------"«difftime(end,init)«"----------------------"«std::endl;
00277                 BitVector::iterator b_it;
00278                 b_it=ris.begin();
00279                 size_t i =0;
00280
00281                 time(&init);
00282                 for (;b_it!=ris.end();++b_it,++i){
00283                         if (!(*b_it))
00284                                 k.push_back(i);
00285                 }
00286                 time(&end);
00287                 //cout«"----------secondo ciclo
      interno------"«difftime(end,init)«"----------------------"«std::endl;
00288                 return k;
00289
00290         }
00291
00292         ItemType getFirstItem(){
00293                 List_Type::iterator it=seq->begin();
00294                 return (*it);
00295         }
00296
00297         CCSMSequence* getLastItem(){
00298                 return last;
00299         }
00300
00301         void setLastItem(CCSMSequence* in){
00302                 last=in;
00303         }
00304
00305         ResultItems getSequenceItems(){
00306                 ResultItems ris(*seq,count);
00307                 return ris;
00308         }
00309
00310         // inserisco nella lista le sequenze di cui questa e il prefisso
00311         //
00312         void addPrefixSequence(CCSMSequence* in){
00313                 prefix.push_back(in);
00314         }
00315
00316         std::list<CCSMSequence*> getPrefixSequence(){
00317                 return prefix;
00318         }
00319
00320         // inserisco nella lista le sequence di cui questa e il suffisso
00321         //
00322         void addSuffixSequence(CCSMSequence* in){
00323                 suffix.push_back(in);
00324         }
00325
00326         std::list<CCSMSequence*> getSuffixSequence(){
00327                 return suffix;
00328         }
00329
00330         //da questa sequenza lunga k-2 puo essere contenuta in una possibile sequenza frequente di
      lunghezza k
00331         //se ha almeno un elemento nella Suffix list e uno nella Prefix list
00332         bool canGenerate(){
00333                 return suffix.size()>0 && prefix.size()>0;
```

```
00334         }
00335
00336         void read(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes);
00337
00338 };
00339
00340
00341 }//END NAMESPACE
00342
00343 #endif
00344
```

7.19 /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrItemSetsExtraction.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Algorithms/Bodymap.hpp"
```

Data Structures

- class minerule::ConstrItemSetsExtraction

Namespaces

- namespace minerule

7.20 ConstrItemSetsExtraction.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __CONSTRITEMSETSEXTRACTION_H
00017 #define __CONSTRITEMSETSEXTRACTION_H
00018
00019 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00020 #include "minerule/Database/Connection.hpp"
00021 #include "minerule/Algorithms/Bodymap.hpp"
00022
00023 namespace minerule {
00024
00025   class ConstrItemSetsExtraction : public MiningAlgorithm {
00026   private:
00027     AlgorithmsOptions options;
00028     Connection connection;
00029         SourceTable* sourceTable;
00030         SourceTable::Iterator bodyIterator;
00031
00032
00033     void prepareData();
```

```
00034
00035   public:
00036     ConstrItemSetsExtraction(const OptimizedMinerule& mr) :
00037       MiningAlgorithm(mr) {}
00038
00039     virtual ~ConstrItemSetsExtraction() {}
00040
00041     virtual void mineRules();
00042
00043     virtual bool canHandleMinerule() const {
00044       return
00045         !minerule.getParsedMinerule().hasCrossConditions() &&
00046         !minerule.getParsedMinerule().requiresClusters() &&
00047         !minerule.getParsedMinerule().hasDisjuctionsInMC();
00048     }
00049
00050   };
00051 }
00052
00053
00054
00055 #endif
```

7.21 /Users/esposito/Software/minerule/include/minerule/Algorithms/ConstrTree.hpp File Reference

```
#include <vector>
#include <set>
#include "minerule/Algorithms/IncrementalAlgorithm.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Algorithms/IncrAlgoClasses.hpp"
```

Data Structures

- class minerule::ConstrTree

Namespaces

- namespace minerule

7.22 ConstrTree.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __CONSTRUCTIVE_INCREMENTAL_ALGO_H__
00017 #define __CONSTRUCTIVE_INCREMENTAL_ALGO_H__
00018
00019 #include<vector>
```

```
00020 #include<set>
00021
00022 #include "minerule/Algorithms/IncrementalAlgorithm.hpp"
00023 #include "minerule/Database/ItemType.hpp"
00024 #include "minerule/Algorithms/IncrAlgoClasses.hpp"
00025
00026 namespace minerule {
00027
00032 class ConstrTree : public IncrementalAlgorithm {
00033  protected:
00034   Body* root;
00035   size_t ngroups;
00036   mrdb::ResultSet* rb2;
00037   mrdb::ResultSet* rh2;
00038   SourceRowColumnIds bodyDes;
00039   SourceRowColumnIds headDes;
00040   mrdb::Statement* stateb2;
00041   mrdb::Statement* stateh2;
00042
00043   void adjustSuppRSet();
00044   void adjustSuppMIndex();
00045   void insertRulesInStructure();
00046   void adjustSupp();
00047   void prepareData();
00048
00049   size_t buildAttrStr(const ParsedMinerule::AttrVector& attr,
00050                       size_t startIndex,
00051                       std::string& attrStr,
00052                       std::vector<int>& des) const;
00053
00054   std::string buildQry( const std::string& groupAttrStr,
00055                         const std::string& attrStr,
00056                         const std::string& constraints) const;
00057
00058   Body* getRoot(){return root;}
00059  public:
00060   ConstrTree(const OptimizedMinerule& mr) : IncrementalAlgorithm(mr), root(new Body()), ngroups(0)
      /*,mb2(NULL), mh2(NULL)*/, rb2(NULL), rh2(NULL) { }
00061
00062   virtual ~ConstrTree() {
00063     // Trashing the trashable
00064     if(rh2!=NULL) delete rh2;
00065     if(stateh2!=NULL) delete stateh2;
00066     if(rb2!=NULL) delete rb2;
00067     if(stateb2!=NULL) delete stateb2;
00068
00069     rb2=rh2=NULL;
00070     stateh2=stateb2=NULL;
00071
00072         delete root;
00073   }
00074
00075   virtual void execute() ;
00076 };
00077
00078 } // namespace
00079
00080 #endif
```

7.23 /Users/esposito/Software/minerule/include/minerule/Algorithms/DestrTree.hpp File Reference

```
#include <vector>
#include <set>
#include "IncrementalAlgorithm.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Algorithms/IncrAlgoClasses.hpp"
```

Data Structures

- class minerule::DestrTree

Namespaces

- namespace minerule

7.24 DestrTree.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __DESTRUCTIVE_INCREMENTAL_ALGO_H__
00017 #define __DESTRUCTIVE_INCREMENTAL_ALGO_H__
00018
00019 #include<vector>
00020 #include<set>
00021
00022 #include "IncrementalAlgorithm.hpp"
00023 #include "minerule/Database/ItemType.hpp"
00024 #include "minerule/Algorithms/IncrAlgoClasses.hpp"
00025
00026 namespace minerule {
00027
00032 class DestrTree : public IncrementalAlgorithm {
00033  protected:
00034   Body* root;
00035   size_t ngroups;
00036   mrdb::ResultSet* rb1;
00037   mrdb::ResultSet* rh1;
00038   mrdb::ResultSet* rb1nb2;
00039   mrdb::ResultSet* rh1nh2;
00040   SourceRowColumnIds bodyDes;
00041   SourceRowColumnIds headDes;
00042
00043   mrdb::Statement* stateb1, *stateb1nb2;
00044   mrdb::Statement* stateh1, *stateh1nh2;
00045
00046
00047   void adjustSuppRSet();
00048   void adjustSuppMIndex();
00049   void insertRulesInStructure();
00050   void adjustSupp();
00051   void prepareData();
00052
00053   size_t buildAttrStr(const ParsedMinerule::AttrVector& attr,
00054                       size_t startIndex,
00055                       std::string& attrStr,
00056                       std::vector<int>& des) const;
00057
00058   std::string buildQry( const std::string& groupAttrStr,
00059                         const std::string& attrStr,
00060                         const std::string& constraints) const;
00061
00062   std::string buildQry1NotQry2( const std::string& groupAttrStr,
00063                                 const std::string& attrStr,
00064                                 const std::string& constraint1,
00065                                 const std::string& constraint2) const;
00066
00067   Body* getRoot(){return root;}
00068  public:
00069   DestrTree(const OptimizedMinerule& mr) : IncrementalAlgorithm(mr), root(new Body()), ngroups(0),
      rb1(NULL), rh1(NULL), rb1nb2(NULL), rh1nh2(NULL) { }
00070
00071   virtual ~DestrTree() {
00072   // Trashing the trashable
00073         if(rh1!=NULL) delete rh1;
00074         if(stateh1!=NULL) delete stateh1;
00075         if(rb1!=NULL) delete rb1;
00076         if(stateb1!=NULL) delete stateb1;
```

```
00077         rb1=rh1=NULL;
00078         stateh1=stateb1=NULL;
00079
00080     delete root;
00081   }
00082
00083   virtual void execute() ;
00084 };
00085
00086 } // namespace
00087
00088 #endif
```

7.25 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSMiner.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
```

Data Structures

- class minerule::FSMiner

Namespaces

- namespace minerule

7.26 FSMiner.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __FSMINER_H__
00017 #define __FSMINER_H__
00018
00019 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00020 #include "minerule/Database/Connection.hpp"
00021 #include "minerule/Utils/AlgorithmTypes.hpp"
00022
00023 namespace minerule {
00024
00025   class FSMiner : public MiningAlgorithm {
00026   private:
00027     AlgorithmsOptions options;
00028
00029     Connection connection;
00030     SourceRowColumnIds rowDes;
00031     mrdb::PreparedStatement* statement;
00032
```

```
00033     void prepareData();
00034
00035   public:
00036     FSMiner(const OptimizedMinerule& mr, const AlgorithmsOptions& opts) :
00037        MiningAlgorithm(mr), options(opts), statement(NULL) {}
00038
00039     virtual ~FSMiner() {}
00040
00041     virtual void mineRules();
00042
00043     virtual bool canHandleMinerule() const {
00044       return minerule.getParsedMinerule().miningTask == MTMineSequences;
00045     }
00046
00047                 virtual SourceTableRequirements sourceTableRequirements() const {
00048                         return SourceTableRequirements(SourceTableRequirements::SortedGids);
00049                 };
00050
00051
00052   };
00053 }
00054
00055
00056
00057 #endif
```

7.27 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTree.hpp File Reference

```
#include <fstream>
#include <map>
#include <iostream>
#include <string>
#include <vector>
#include "FSTreeNode.hpp"
#include "FSTreeSequence.hpp"
#include "minerule/Optimizer/OptimizedMinerule.hpp"
```

Data Structures

- class minerule::FSTree

Namespaces

- namespace minerule

7.28 FSTree.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
```

```
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef FSTREE_H
00017 #define FSTREE_H
00018 #include <fstream>
00019 #include <map>
00020 #include <iostream>
00021 #include <string>
00022 #include <vector>
00023 #include "FSTreeNode.hpp"
00024 #include "FSTreeSequence.hpp"
00025 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00026 //#include "sequenceList.hpp"
00027
00028 namespace minerule {
00029
00034 class FSTree{
00035  private:
00036
00040   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* link_S;
00046   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* link_NS;
00047
00051   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* result;
00052
00057   //sequenceList* m_list;
00058   std::map<FSTreeSequence, std::vector<FSTreeNode*>*, FSTreeSequence::less_sequence> m_list;
00062   std::string source;
00063
00067   FSTreeNode* root;
00068
00072   int num;
00076   double threshold;
00077
00078   int n_nodi;
00079
00080   int max_length;
00081  public:
00082
00087   FSTree(const OptimizedMinerule& opt) :
00088     root(new FSTreeNode()),
00089     num(0),
00090     threshold(0) {
00091     root= new FSTreeNode();
00092     num=0;
00093     threshold=0;
00094
00095     max_length = opt.getParsedMinerule().bodyCardinalities.getMax();
00096     link_S= new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00097     link_NS=new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00098     result = new std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00099   }
00104   void setThreshold(double t){
00105     threshold=t*num;
00106     //threshold=t;
00107   }
00108
00109
00113   int getN_nodi();
00114
00118   void resetN_nodi();
00119
00130   std::vector<FSTreeSequence*>*  fraziona(FSTreeSequence *s);
00134   void createLinkSTable(mrdb::ResultSet*, const SourceRowColumnIds&);
00135
00139   void createLinkNSTable();
00140
00146   void insertLink(FSTreeSequence* s);
00147
00153   void addResult(FSTreeNode* n);
00161   int countPath(FSTreeNode* n);
00162
00163   void stampa(FSTreeNode* n);
00164
00171   void addResults(std::vector<FSTreeNode*>* vec);
00172
00178   void mine();
00179
00187   FSTreeNode* appendChild(FSTreeNode* r, const ItemType& t);
00188
00195   void addList(FSTreeNode* father, FSTreeNode* children);
00196
00201   void insertTree(FSTreeNode* r, FSTreeSequence* t);
00202
00208   void construct_Tree(mrdb::ResultSet*, const SourceRowColumnIds&);
00209
00213   ~FSTree();
```

```
00214
00218   double getThreshold();
00219
00220
00221   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* getResult();
00222
00223
00224
00225 };
00226
00227 } //minerule
00228
00229 #endif
00230
```

7.29 /Users/esposito/Software/minerule/include/minerule/Algorithms/FSTreeNode.hpp File Reference

```
#include <vector>
#include "minerule/Database/ItemType.hpp"
```

Data Structures

- class minerule::FSTreeNode

Namespaces

- namespace minerule

7.30 FSTreeNode.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef NODE_H
00017 #define NODE_H
00018
00019 #include <vector>
00020 #include "minerule/Database/ItemType.hpp"
00021
00022
00023 namespace minerule {
00027   class FSTreeNode{
00028   private:
00032     std::vector<FSTreeNode* >* child;
00036     ItemType label;
00040     int count;
00044     FSTreeNode* parent;
00045   public:
00050     FSTreeNode(const ItemType& l);
00051
```

```
00055     FSTreeNode();
00056
00061     FSTreeNode(FSTreeNode* p);
00062
00066     ~FSTreeNode();
00067
00071     const ItemType& getLabel();
00072
00076     std::vector<FSTreeNode*>* getChild();
00077
00081     int getCount();
00082
00086     FSTreeNode* getParent();
00087
00091     void setCount(int n);
00092
00096     void setParent(FSTreeNode* s);
00097
00101     void insertChild(FSTreeNode* s);
00102   };
00103
00104 } // minerule
00105 #endif
00106
```

7.31 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ FSTreeSequence.hpp File Reference

```
#include <list>
#include <string>
#include <iostream>
#include "minerule/Database/ItemType.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
```

Data Structures

- class minerule::FSTreeSequence
- struct minerule::FSTreeSequence::less_sequence

Namespaces

- namespace minerule

7.32 FSTreeSequence.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef Sequence_H
```

```
00017 #define Sequence_H
00018 #include <list>
00019 #include <string>
00020 #include <iostream>
00021 #include "minerule/Database/ItemType.hpp"
00022 #include "minerule/Database/SourceRowColumnIds.hpp"
00023
00024 namespace minerule {
00025
00029   class FSTreeSequence{
00030   public:
00031     typedef std::list<ItemType> ItemVector;
00032   private:
00036       ItemVector* seq;
00037   public:
00041     struct less_sequence{
00042       bool operator()(const FSTreeSequence& first, const FSTreeSequence& second) const {
00043               return (first < second);
00044       }
00045
00046       bool comp(const FSTreeSequence& first, const FSTreeSequence& second) const {
00047         return (first < second);
00048       }
00049
00050     };
00051
00055     FSTreeSequence() {
00056       seq=new ItemVector();
00057     };
00058
00063     FSTreeSequence(const FSTreeSequence &in){
00064       seq=new ItemVector();
00065       *seq = *in.seq;
00066     };
00067
00075     FSTreeSequence(const FSTreeSequence &in, size_t start, size_t n_item);
00076
00080     ~FSTreeSequence();
00081
00086     ItemVector* getFSTreeSequence();
00087
00092     void add(const ItemType& s);
00093
00094
00098     void stampa();
00099
00104     size_t size();
00105
00110     ItemType removeHead();
00111
00116     void insertHead(const ItemType& s);
00117
00121     std::string toStdString();
00122
00126     void svuota();
00127
00132     bool operator==(FSTreeSequence& s);
00133
00139     void read(mrdb::ResultSet* rs, const SourceRowColumnIds&);
00140
00141     // ItemType pre_proc();
00146     bool operator<(const FSTreeSequence& in) const;
00147
00148   };
00149
00150 } //namespace
00151
00152 #endif
```

7.33 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ IDIncrementalAlgorithm.hpp File Reference

```
#include <vector>
#include <set>
#include "IncrementalAlgorithm.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Result/QueryResult.hpp"
```

Data Structures

- class minerule::IDIncrementalAlgorithm

Namespaces

- namespace minerule

7.34 IDIncrementalAlgorithm.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __IDINCREMENTAL_ALGO_H__
00017 #define __IDINCREMENTAL_ALGO_H__
00018
00019 #include<vector>
00020 #include<set>
00021
00022 #include "IncrementalAlgorithm.hpp"
00023 #include "minerule/Database/ItemType.hpp"
00024 #include "minerule/Result/QueryResult.hpp"
00025
00026 namespace minerule {
00027
00039 class IDIncrementalAlgorithm : public IncrementalAlgorithm {
00040   typedef std::pair< std::set<ItemType>*, std::set<ItemType>* > ValidRule;
00041   typedef std::vector< ValidRule > ValidRules;
00042  protected:
00043   const ParsedMinerule::AttrVector* attrList;
00044
00045   std::set<ItemType>* fillValidItems(const std::string& constraints) const ;
00046
00047   void getItemInfos( std::string& itemDescr, SourceRowColumnIds& hbsr ) const;
00048   bool checkInclusion(const std::set<ItemType>& validOnes, const ItemSet& foundOnes) const;
00049   bool checkInValidRules(const ValidRules& validRules, Rule& r) const;
00050   void filterQueries(const ValidRules& validRules) ;
00051
00052  public:
00053
00054   IDIncrementalAlgorithm(const OptimizedMinerule& mr) : IncrementalAlgorithm(mr), attrList(NULL) { }
00055   virtual ~IDIncrementalAlgorithm() {}
00056
00057   virtual void execute() ;
00058 };
00059
00060 } // namespace
00061
00062 #endif
```

7.35 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ IncrAlgoClasses.hpp File Reference

```
#include "minerule/Database/ItemType.hpp"
#include <vector>
#include <iostream>
#include <map>
#include "minerule/Database/Connection.hpp"
```

Data Structures

- class minerule::Body
- class minerule::Head
- class minerule::NodeRow
- class minerule::NodeRowB

Namespaces

- namespace minerule

7.36 IncrAlgoClasses.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017
00018
00019 #ifndef INCRALGOCLASSI_H
00020 #define INCRALGOCLASSI_H
00021 #include "minerule/Database/ItemType.hpp"
00022 #include <vector>
00023 #include <iostream>
00024 #include <map>
00025 #include "minerule/Database/Connection.hpp"
00026
00027 namespace minerule {
00028
00029  class NodeRow;
00030  class NodeRowB;
00031  class Head;
00032
00033  /*******************class Body******/
00034  class Body
00035  {public:
00036   typedef std::map<ItemType,NodeRowB*> RowBContainer;
00037
00038   private:
00039    ItemType ancestor;
00040    NodeRowB* findBody(ItemSet:: iterator ite,
00041                       ItemSet:: iterator itend);
00042    NodeRowB* findHead(ItemSet:: iterator ite,
00043                            ItemSet:: iterator itend);
00044    Head* insertItemSetB(ItemSet::iterator ite,
00045                         ItemSet::iterator itend,
00046                         double d);
00047   public:
00048   static size_t countb;
00049   Body();
00050   ~Body();
00051
00052   RowBContainer* NRB;
00053
00054   void setAncestor(ItemType b){ancestor=b;}
00055   ItemType getAncestor(){return ancestor;}
00056   Head* insertItemSetB(ItemSet& SREV, double supp);
00057   void findBodiesInTree(ItemSet* body);
00058   void findChildInTree(ItemSet* b1, ItemSet* h1);
00059   void findRulesInTree(ItemSet* itemset_body,ItemSet* itemset_head);
00060   void findRulesInTree(ItemSet* b1,ItemSet* b1nb2,
```

```
00061                       ItemSet* h1,ItemSet* h1nh2);
00062   void extractRules(std::vector<ItemType>& body,
00063                     double thrR, double thrB,int ngroups,Connection* pconnection);
00064   void provaStampaLiv1();
00065  };
00066
00067  /*******************class Head******/
00068  class Head
00069  {public:
00070   typedef std::map<ItemType,NodeRow*> RowContainer;
00071
00072   private:
00073    ItemType ancestor;
00074
00075   public:
00076    static size_t counth;
00077    Head();
00078    ~Head();
00079
00080    RowContainer* NR;
00081    void setAncestor(ItemType h){ancestor=h;}
00082    ItemType getAncestor(){return ancestor;}
00083    void insertItemSetH(ItemSet& SREV, double supp);
00084    void insertItemSetH(ItemSet:: iterator ite,
00085                        ItemSet:: iterator itend,
00086                        double s);
00087    void extractRules(const std::vector<ItemType>& body,
00088                      std::vector<ItemType>& head,
00089                      double thrR, double thrB, double suppB, int ngroups,Connection* pconnection);
00090    void findHead(ItemSet* head);
00091    void findHead2bis(ItemSet* h1nh2,ItemSet* h1);
00092    void findHead2(ItemSet* h1);
00093  };
00094
00095
00096  /*******************class NodeRow******/
00097
00098  class NodeRow{
00099
00100   private:
00101   Head* child;
00102   double support;
00103   //marca per algo distr
00104   bool marked;
00105
00106   public:
00107   NodeRow() : child(NULL),support(0) {marked=0;}
00108   NodeRow(const NodeRow& nr) : support(nr.support) {}
00109   ~NodeRow(){delete child;}
00110
00111   double getSupp(){return support;}
00112   void setSupp(double& s){support=s;/*cout«"ho fissato il supp a "«support«"  ";*/}
00113   void incremSupp(){support++;/*cout«"ora il supp e' "«support«"  ";*/}
00114   void decremSupp(){support--;/*cout«"ora il supp e' "«support«"  ";*/}
00115   Head* getChild() {return child;}
00116   Head* setChild() {child = new Head(); return child;}
00117   void mark(){marked=1;}
00118   void demark(){marked=0;}
00119   bool isMarked(){return marked;}
00120  };
00121
00122
00123
00124  /*******************class NodeRowB******/
00125
00126  //classe per le righe del Body
00127  //ogni riga del Body potrebbe avere un puntatore ad un Node Head
00128  //nota che una riga potrebbe anche non avere un puntatore head.In questo caso
00129  //avrebbe almeno un figlio...(caso con |body|>1)
00130
00131  class NodeRowB : public NodeRow
00132  {
00133   private:
00134   Head* head;
00135   Body* child;
00136   public:
00137   NodeRowB(): head(NULL),child(NULL){ }
00138   NodeRowB(const NodeRowB& nr){}
00139   ~NodeRowB(){delete head; delete child;}
00140   Body* getChild() {return child;}
00141   Body* setChild() {child = new Body(); return child;}
00142   Head* setHead() {
00143     if (head==NULL)
00144       head = new Head();
00145     return head;}
00146   Head* getHead() {return head;}
00147
```

```
00148  };
00149
00150
00151
00152 }
00153 #endif
```

7.37 /Users/esposito/Software/minerule/include/minerule/Algorithms/IncrementalAlgorithm.hpp File Reference

```
#include "minerule/Optimizer/OptimizedMinerule.hpp"
```

Data Structures

- class minerule::IncrementalAlgorithm

Namespaces

- namespace minerule

7.38 IncrementalAlgorithm.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __INCREMENTAL_ALGORITHM_H__
00017 #define __INCREMENTAL_ALGORITHM_H__
00018
00019 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00020
00021 namespace minerule {
00022
00023 class IncrementalAlgorithm {
00024  protected:
00025   const OptimizedMinerule* minerule;
00026  public:
00027   IncrementalAlgorithm(const OptimizedMinerule& mr) :
00028     minerule(&mr) {}
00029
00030   virtual ~IncrementalAlgorithm() {}
00031
00032   virtual void execute() =0;
00033
00034   static IncrementalAlgorithm*
00035   newIncrementalAlgorithm(const OptimizedMinerule& mr);
00036 };
00037
00038
00039 } //namespace
00040
00041 #endif
```

7.39 /Users/esposito/Software/minerule/include/minerule/Algorithms/MiningAlgorithmBase.hpp File Reference

```
#include "AlgorithmsOptions.hpp"
#include "minerule/Optimizer/OptimizedMinerule.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Database/SourceTableRequirements.hpp"
#include "minerule/Utils/Progress.hpp"
```

Data Structures

- class minerule::MiningAlgorithmBase
- class minerule::MiningAlgorithm

Namespaces

- namespace minerule

7.40 MiningAlgorithmBase.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MININGALGORITHM_BASE_H__
00017 #define __MININGALGORITHM_BASE_H__
00018
00019 #include "AlgorithmsOptions.hpp"
00020 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00021 #include "minerule/Utils/AlgorithmTypes.hpp"
00022 #include "minerule/Database/Connection.hpp"
00023 #include "minerule/Database/SourceTableRequirements.hpp"
00024 #include "minerule/Utils/Progress.hpp"
00025
00026 namespace minerule {
00027
00032         class MiningAlgorithmBase {
00033         protected:
00034                 const OptimizedMinerule& minerule;
00035         public:
00036                 MiningAlgorithmBase( const OptimizedMinerule& mr ) : minerule(mr) {}
00037                 virtual ~MiningAlgorithmBase() {}
00038
00039                 virtual void execute() {
00040                         throw MineruleException( MR_ERROR_INTERNAL, "This method should never be
      executed!");
00041                 }
00042
00043                 virtual SourceTableRequirements sourceTableRequirements() const {
00044                         return SourceTableRequirements();
00045                 };
00046
```

```
00047
00048                 virtual bool canHandleMinerule() const {
00049                         return false;
00050                 }
00051
00052                 virtual const OptimizedMinerule& optimizedMinerule() const { return minerule; }
00053
00054   // Instantiate the algorithm specified by t
00055                 static MiningAlgorithmBase* algorithmForType(AlgorithmTypes t, const
      OptimizedMinerule&)
00056                         ;
00057         };
00058
00059         class MiningAlgorithm : public MiningAlgorithmBase {
00060         protected:
00061                 AlgorithmsOptions options;
00062                 Connection connection;
00063         public:
00064                 MiningAlgorithm(const OptimizedMinerule& m) : MiningAlgorithmBase(m) {}
00065
00066                 virtual void initialize() {
00067                         MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00068
00069                         options.setSupport( minerule.getParsedMinerule().sup );
00070                         options.setConfidence( minerule.getParsedMinerule().conf );
00071                         options.setBodyCardinalities( minerule.getParsedMinerule().bodyCardinalities);
00072                         options.setHeadCardinalities( minerule.getParsedMinerule().headCardinalities);
00073
00074                         MinMaxPair bodyCards( options.getBodyCardinalities() );
00075                         bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00076                         options.setBodyCardinalities(bodyCards);
00077
00078                         MinMaxPair headCards( options.getHeadCardinalities() );
00079                         headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00080                         options.setHeadCardinalities(headCards);
00081
00082
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00083                         connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00084
      connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00085
      connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00086                         if(minerule.getParsedMinerule().ha.size()>0)
00087
      connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00088                         else
00089                                 connection.createResultTables();
00090                         connection.init();
00091                 }
00092
00093                 virtual void execute() {
00094                         initialize();
00095                         mineRules();
00096                 }
00097
00098                 virtual void mineRules() {
00099                                 throw MineruleException(MR_ERROR_INTERNAL, "This method should be
      implemented in sub-classes!");
00100                 }
00101
00102         };
00103
00104
00105
00106 }
00107 #endif
```

7.41 /Users/esposito/Software/minerule/include/minerule/Algorithms/↩ ResultCombinator.hpp File Reference

```
#include "minerule/Algorithms/IncrementalAlgorithm.hpp"
```

Data Structures

- class minerule::ResultCombinator

Namespaces

- namespace minerule

7.42 ResultCombinator.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __RESULT_COMBINATOR_H__
00017 #define __RESULT_COMBINATOR_H__
00018
00019 #include "minerule/Algorithms/IncrementalAlgorithm.hpp"
00020
00021 namespace minerule {
00022
00023   class ResultCombinator : public IncrementalAlgorithm {
00024   public:
00025     ResultCombinator(const OptimizedMinerule& mr) : IncrementalAlgorithm(mr) {}
00026     virtual ~ResultCombinator() {}
00027
00028     virtual void execute() ;
00029   };
00030
00031 }
00032
00033 #endif
```

7.43 /Users/esposito/Software/minerule/include/minerule/Algorithms/STSMiner.hpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
#include "minerule/Utils/Bitstring.hpp"
#include "minerule/Utils/StringUtils.hpp"
#include <fstream>
#include <locale>
#include <map>
#include <math.h>
#include <unistd.h>
```

Data Structures

- struct minerule::Attr_def
- struct minerule::Constraints
- class minerule::STSMiner

Namespaces

- namespace minerule

Typedefs

- typedef std::map< long, int > minerule::GroupMap
- typedef std::pair< long, int > minerule::GroupPair

7.44 STSMiner.hpp

Go to the documentation of this file.

```
00001 #ifndef STSMiner_H
00002 #define STSMiner_H
00003
00004 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00005 #include "minerule/Database/MRResultSet.hpp"
00006 #include "minerule/Database/PrepareDataUtils.hpp"
00007 #include "minerule/Utils/Converter.hpp"
00008 #include "minerule/Utils/AlgorithmTypes.hpp"
00009 #include "minerule/Utils/Bitstring.hpp"
00010 #include "minerule/Utils/StringUtils.hpp"
00011 #include <fstream>
00012 #include <locale>
00013 #include <map>
00014 #include <math.h>
00015 #include <unistd.h>
00016
00017 namespace minerule {
00018
00019 typedef std::map<long, int> GroupMap;
00020 typedef std::pair<long, int> GroupPair;
00021
00022 struct Attr_def {
00023     std::string attr_name;
00024     std::string attr_type;
00025
00026     Attr_def(std::string n, std::string t): attr_name(n), attr_type(t) { }
00027 };
00028
00032 struct Constraints {
00033     std::vector<BitString> bem;
00034     std::vector<std::vector<std::pair<int,int> > > mid_counters;
00035
00036     void clear() {
00037         bem.clear();
00038         mid_counters.clear();
00039     }
00040
00041     Constraints() {}
00042 };
00043
00044 class STSMiner : public MiningAlgorithm{
00045
00046 private:
00047     int seq_count;
00048     AlgorithmsOptions options;
00049     ParsedMinerule pm;
00050
00051     std::string groupAttrList;
00052     std::string ordAttrList;
00053     std::string bodyAttrList;
00054     std::vector<Attr_def> attr_types;
00055     std::vector<std::string> attr_list;
00056     GroupMap hashtable;
00057     std::vector<std::vector<std::string> > hash_vect;
00058     std::vector<BitString*> bem_vect;
00059     std::vector<std::vector<Constraints*> >  constr_vect;
00060     std::vector<std::vector<Constraints*> >  constr_vect_singleton;
00061     std::vector<BitString*> bvptr;
00062     std::vector<MinMaxPair> min_max_mid_count;
00063     std::vector<std::vector<std::pair<int,int> > > mid_count;
00064     std::vector<std::vector<std::string> > results;
00065     std::vector<std::vector<std::vector<int> > > prefix_pos_lists;//[id-1][pos_list_entry]
00066     std::vector<std::vector<std::pair<int,int>* > > pos_lists;
00067     std::vector<std::vector<std::pair<int,int>* > > pos_lists_singleton;
```

```
00068     std::vector<std::vector<int> > pos_d_lists;
00069
00070     bool context_dep_dist;
00071     bool context_dep_BEM;
00072     bool context_free_BEM;
00073     bool active_count;
00074     bool DISTINCT;
00075     bool log;
00076     int max_MIN_COUNT;
00077     int grpId_count;//number of values found for group by attribute in source table
00078     int bemCD_constraints;
00079     int bemCF_constraints;
00080     int bvptr_length;
00081     int min_trackid;
00082     int max_trackid;
00083     int bem_vect_length;
00084     int max_seq_length;
00085     int min_seq_length;
00086
00087     std::ofstream matcher_cache;
00088     std::ofstream seqicache;
00089
00090     bool extractKSequences(int k);
00091     bool extract2Sequences();
00092     bool extract1Sequences();
00093     std::string union_rows();
00094     void createBitVectors();
00095     void createBitVectors_context_dep();
00096     void saveSequenceData(int k,int& id1, int& id2, bool check_bem,
00097                           BitString tmp0,
00098                           std::vector<std::pair<int,int>* > ris_l,
00099                           std::vector<int> pos_d_l,
00100                           BitString* new_bem_string,
00101                           std::vector<bool>* closed_seq,
00102                           std::vector<std::vector<int> > prefix_pos_l,
00103                           std::vector<std::pair<int,int> > new_mid_count_vect,
00104                           std::vector<Constraints*> c_v);
00105     Constraints* init_c_v(std::vector<std::string> AttrList, bool* check_bem_cd);
00106     std::vector<std::vector<std::string> > init_dist_val_vect(std::vector<std::string> AttrList);
00107     BitString initbemvect_context_dep(std::vector<std::string> bodyAttrVal);
00108     void initbemvect(int id,std::vector<std::string> attrVal);
00109     bool checkBem(int id, int k);
00110     int pruning1Seq();
00111     bool pruningSeq2(std::string sqlQuery);
00112     bool pruningSeqK(int k,std::string sqlQuery);
00113     void updateMidCounters_singleton(Constraints* c);
00114     void updateMidCounters(Constraints* constr,int id1,int id2, int i, int j);
00115     bool check_pos_lists_with_gap(int id1, int id2,
00116                                   std::vector<std::pair<int,int>* >* ris_l,
00117                                   BitString* bv_constr,
00118                                   std::vector<std::vector<int> >* new_prefix_pos_list,
00119                                   std::vector<int>* new_pos_d_l,
00120                                   bool* BEM_context_free,
00121                                   std::vector<Constraints*>* c_v,
00122                                   BitString* bem_cd);
00123     bool sourceFilter();
00124     void check_go_on(Constraints* c,BitString* ris);
00125     int findIndex(int id, int pos, int grpId);
00126     bool init_bemCD(int id1, int index_id1, int id2, int pos2, int grpId, Constraints*
      output,BitString* bem_cd);
00127     BitString inferBemCD(BitString bs1,BitString bs2);
00128     BitString inferBem(int id1,int id2,int i, int j);
00129     void init_bem_bit_vector(int k,int id1,int id2,bool* check_bem,
00130                              BitString* bem_cf,
00131                              BitString* new_bem_vect,
00132                              std::vector<std::pair<int,int> >* new_count);
00133     bool checkMidCounters(std::vector<std::pair<int,int> >mc_v);
00134     void updateBemData(Constraints* constr,int id1,int id2,int prefix_pos,
00135                                int pos,int grp,bool* check_bem_cd,BitString* bem_cd);
00136     void saveSequences(int k);
00137
00138     //clean memory functions
00139     void delete_all(int k);
00140     void dropResultsTables(int k);
00141     void resetStructs(int k,
00142                               int size_before_results,
00143                               int size_before_bvptr,
00144                               int size_before_pos,
00145                               int size_before_prefix_pos,
00146                               int size_before_bem_vect,
00147                               int size_before_mid_count,
00148                               int size_before_pos_d,
00149                               int size_before);
00150
00151 public:
00152     STSMiner(const OptimizedMinerule& mr, const AlgorithmsOptions& opts) :
00153         MiningAlgorithm(mr),pm(ParsedMinerule(mr.getParsedMinerule())) {
```

```
00154         MRDebug()«"STSMiner initialization..."«std::endl;
00155        /* double vm, rss;
00156         process_mem_usage(vm, rss);
00157         std::cout « "VM: " « vm « "; RSS: " « rss « std::endl;*/
00158
00159         this->initialize();
00160         connection.createResultTables();
00161
00162         std::string temp = "/tmp/" + pm.tab_result + ".s";
00163         seqicache.open(temp.c_str(), std::ofstream::trunc);
00164
00165         temp = "/tmp/" + pm.tab_result + ".m";
00166         matcher_cache.open(temp.c_str(), std::ofstream::trunc);
00167
00168         options = opts;
00169         log=true;
00170         max_seq_length = pm.length.getMax();
00171         min_seq_length = pm.length.getMin();
00172         active_count=false;
00173
00174         groupAttrList = "";
00175         ordAttrList = "";
00176         bodyAttrList = "";
00177
00178         for(int i=0; i<pm.ga.size()-1; ++i)
00179             groupAttrList+=pm.ga[i]+",";
00180         groupAttrList+=pm.ga[pm.ga.size()-1];
00181
00182         for(int i=0; i<pm.oa.size()-1; ++i)
00183             ordAttrList+=pm.oa[i]+",";
00184         ordAttrList+=pm.oa[pm.oa.size()-1];
00185
00186         for(int i=0; i<pm.ba.size()-1; ++i)
00187             bodyAttrList+=pm.ba[i]+",";
00188         bodyAttrList+=pm.ba[pm.ba.size()-1];
00189
00190         seq_count=1;
00191         DISTINCT=pm.distinct;
00192         context_dep_dist=false;
00193         context_dep_BEM=false;
00194         context_free_BEM=false;
00195         max_MIN_COUNT=1;
00196         bemCD_constraints=0;
00197         bemCF_constraints=0;
00198
00199         initAttrNameList();
00200
00201         for(int i=0;i<pm.seq_bem_vect.size();++i) {
00202             Bem_cond* bc = pm.seq_bem_vect[i];
00203             bool cf=false;
00204             bool cd=false;
00205
00206             do {
00207                 if(!containedIn(bc->attr,pm.ba)) {
00208                     context_dep_BEM=true;
00209                     cd=true;
00210                 }
00211                 else {
00212                     context_free_BEM=true;
00213                     cf=true;
00214                 }
00215                 bc=bc->and_c;
00216             }while(bc);
00217             if(cd)
00218                 bemCD_constraints++;
00219             if(cf)
00220                 bemCF_constraints++;
00221         }
00222
00223         for(int i=0;i<pm.seq_dist_vect.size();++i) {
00224             for(int j=0; j<pm.seq_dist_vect[i]->attr.size();++j) {
00225                 if(!containedIn(pm.seq_dist_vect[i]->attr[j],pm.ba))
00226                     context_dep_dist=true;
00227             }
00228         }
00229
00230         int x=0;
00231         for(int i=0;i<pm.seq_bem_vect.size();++i)
00232             if(pm.seq_bem_vect[i]->type.compare("MID")==0)
00233                 x++;
00234         bem_vect_length= 3 >= 2+x ? 3 : 2+x;
00235
00236         if(bem_vect_length>0) {
00237
00238             int max_int_val=std::numeric_limits<int>::max();
00239             for(int i=1;i<bem_vect_length-1;++i)
00240                 min_max_mid_count.push_back(MinMaxPair(1,max_int_val));
```

```
00241             x=0;
00242             for(int i=0;i<pm.seq_bem_vect.size();++i) {
00243                 if(pm.seq_bem_vect[i]->type.compare("MID")==0) {
00244                     Bem_cond* temp=pm.seq_bem_vect[i];
00245                     do {
00246                         max_MIN_COUNT=std::max(max_MIN_COUNT,temp->count_min);
00247
      min_max_mid_count[x].setMin(std::max(min_max_mid_count[x].getMin(),temp->count_min));
00248
      min_max_mid_count[x].setMax(std::min(min_max_mid_count[x].getMax(),temp->count_max));
00249                         temp = temp->and_c;
00250                     } while(temp);
00251
00252                     active_count= active_count || min_max_mid_count[x].getMin()!=1
00253                          || min_max_mid_count[x].getMax()!=std::numeric_limits<int>::max();
00254                     ++x;
00255                 }
00256             }
00257         }
00258         initAttrTypesList(pm.tab_source);
00259     }
00260
00261     virtual bool canHandleMinerule() const {
00262       return pm.miningTask == MTMineSequences;
00263     }
00264
00265     virtual void mineRules();
00266
00267     virtual ~STSMiner() { }
00268
00269     virtual SourceTableRequirements sourceTableRequirements() const {
00270             return SourceTableRequirements(SourceTableRequirements::SortedGids);
00271     }
00272
00273
00274 private:
00275
00276     void initAttrNameList() {
00277         attr_list.clear();
00278         for(int i=0;i<pm.ga.size();++i)
00279             attr_list.push_back(pm.ga[i]);
00280
00281         for(int i=0;i<pm.ba.size();++i)
00282             if(!containedIn(pm.ba[i],attr_list))
00283                 attr_list.push_back(pm.ba[i]);
00284         for(int i=0;i<pm.seq_bem_vect.size();++i) {
00285             Bem_cond* bc = pm.seq_bem_vect[i];
00286             do {
00287                 if(!containedIn(bc->attr,attr_list))
00288                     attr_list.push_back(bc->attr);
00289                 bc=bc->and_c;
00290             }while(bc);
00291         }
00292         for(int i=0;i<pm.seq_dist_vect.size();++i)
00293             for(int j=0; j<pm.seq_dist_vect[i]->attr.size();++j)
00294                 if(!containedIn(pm.seq_dist_vect[i]->attr[j],attr_list))
00295                     attr_list.push_back(pm.seq_dist_vect[i]->attr[j]);
00296     }
00297
00298     void initAttrTypesList(std::string src) {
00299         attr_types.clear();
00300         std::string sqlQuery = "SELECT "+toString(attr_list)+" FROM " + src + " LIMIT 1";
00301         mrdb::ResultSet* rs = execQr(sqlQuery);
00302         mrdb::ResultSetMetaData* meta = rs->getMetaData();
00303
00304         if(rs->next()) {
00305             for(int i=0; i<attr_list.size();++i) {
00306                 Attr_def tmp(attr_list[i],meta->getColumnTypeName(i+1));
00307                 attr_types.push_back(tmp);
00308             }
00309         }
00310         else exit(1);
00311         delete rs;
00312     }
00313
00314     std::string find_val(std::string attr_name,std::vector<std::string> AttrVal){
00315
00316         for(int i = 0; i < attr_list.size() ; ++i)
00317             if(attr_list[i].compare(attr_name) == 0)
00318                 return AttrVal[i];
00319         return "";
00320     }
00321
00322
00323     std::string pos_list_string(std::vector<std::pair<int,int>* > pos_l) {
00324         std::string str = "";
00325         for(std::vector<std::pair<int, int>* >::iterator it = pos_l.begin();
```

```
00326             it != pos_l.end(); ++it) {
00327             std::pair<int, int>* pair = *it;
00328             str += "(" + Converter(pair->first).toString()+ ","
00329                 + Converter(pair->second).toString() + ") - ";
00330         }
00331         return str;
00332     }
00333
00334     static std::string toString(std::vector<std::string> v) {
00335         std::string ris="";
00336         if(v.size()>0) {
00337             ris += v[0];
00338             for(int i=1;i<v.size();i++)
00339                 ris += ","+v[i];
00340         }
00341         return ris;
00342     }
00343
00344     std::string get_type(std::string attr_n){
00345         for(int i=0;i<attr_types.size();++i)
00346             if(attr_n.compare(attr_types[i].attr_name)==0)
00347                 return attr_types[i].attr_type;
00348         return (std::string)"unknown";
00349     }
00350
00351     bool textAttr(std::string attr) {
00352         std::string type= get_type(attr);
00353         return (type.compare("varchar")==0 || type.compare("VARCHAR")==0 || type.compare("text")==0 ||
    type.compare("TEXT")==0);
00354     }
00355
00356     bool containedIn(std::string el,std::vector<std::string> list){
00357         for(int i=0;i<list.size();++i)
00358             if(list[i].compare(el)==0)
00359                 return true;
00360         return false;
00361     }
00362
00363     mrdb::ResultSet* execQr(std::string sqlQuery) {
00364
00365       MRDebug() << "Executing query: " + sqlQuery << std::endl;
00366
00367       mrdb::Statement *statement = connection.getMRDBConnection()->createStatement();
00368       mrdb::ResultSet *result = statement->executeQuery(sqlQuery.c_str());
00369
00370       return result;
00371     }
00372
00373     void execQuery(std::string sqlQuery) {
00374
00375       MRDebug() << "Executing statement: " + sqlQuery << std::endl;
00376
00377       mrdb::Statement *statement =
00378           connection.getMRDBConnection()->createStatement();
00379       statement->execute(sqlQuery);
00380
00381       delete statement;
00382     }
00383
00384     void setSeqIDProp() {
00385
00386         std::string query = "SELECT COUNT(DISTINCT (" + groupAttrList + ")) FROM "+ pm.tab_source ;
00387         mrdb::ResultSet* rs = execQr(query);
00388         rs->next();
00389         grpId_count = rs->getInt(1);
00390         delete rs;
00391
00392         query= "ALTER TABLE "+ pm.tab_result + "_Source ADD COLUMN GRPID NUMERIC;";
00393         execQuery(query);
00394
00395         try {
00396
00397             query= "CREATE TABLE TEMP_GRPID(GRPID SERIAL PRIMARY KEY";
00398             for(int i=0; i<pm.ga.size(); ++i)
00399                 query+= ", " + pm.ga[i] + " " +get_type(pm.ga[i]);
00400             query+=");";
00401             execQuery(query);
00402
00403             query= "INSERT INTO TEMP_GRPID(" + groupAttrList + ")"
00404                    " SELECT DISTINCT(" + groupAttrList + ") FROM "+ pm.tab_result + "_Source";
00405             execQuery(query);
00406             execQuery("CREATE INDEX temp_grpid_index ON TEMP_GRPID("+groupAttrList+");");
00407             query="UPDATE "+ pm.tab_result + "_Source src"
00408                    " SET GRPID=(SELECT s.GRPID FROM TEMP_GRPID s WHERE ";
00409             for(int i=0; i<pm.ga.size()-1; ++i)
00410                 query+=" s."+pm.ga[i]+"=src."+pm.ga[i]+" AND ";
00411             query+=" s."+pm.ga[pm.ga.size()-1]+"=src."+pm.ga[pm.ga.size()-1]+");";
```

```
00412             execQuery(query);
00413
00414             //std::vector<std::vector<std::string> > hash_vect;
00415             std::vector<std::string> grpAttrVals;
00416             query= "SELECT DISTINCT GRPID," + groupAttrList + " FROM TEMP_GRPID ORDER BY GRPID";
00417             rs= execQr(query);
00418             while(rs->next()) {
00419                 int grpId= rs->getInt(1);
00420                 for(int i=0; i<pm.ga.size(); ++i)
00421                     grpAttrVals.push_back(rs->getString(2+i));
00422                 hash_vect.push_back(grpAttrVals);
00423                 grpAttrVals.clear();
00424             }
00425             delete rs;
00426
00427             dropTable("TEMP_GRPID");
00428
00429         } catch(...) {
00430             dropTable("TEMP_GRPID");
00431             throw std::exception();
00432         }
00433
00434         pm.ga.clear();
00435         pm.ga.push_back("GRPID");
00436         groupAttrList="GRPID";
00437         //ADD GRPID NUMERIC IN ATTR_TYPE_LIST PER LE VARIE GET_TYPE
00438         initAttrNameList();
00439         initAttrTypesList(pm.tab_result + "_Source");
00440
00441         query = "SELECT MIN("+ groupAttrList + ") FROM "+ pm.tab_result + "_Source;";
00442         rs = execQr(query.c_str());
00443         rs->next();
00444         min_trackid=rs->getInt(1);
00445         delete rs;
00446         query = "SELECT MAX(" + groupAttrList + ") FROM "+ pm.tab_result + "_Source;";
00447         rs = execQr(query);
00448         rs->next();
00449         max_trackid=rs->getInt(1);
00450         delete rs;
00451     }
00452
00453
00454     bool emptyTable(std::string tableName) {
00455         std::string query = "SELECT * FROM " + tableName + " LIMIT 1";
00456         mrdb::ResultSet* rs = execQr(query);
00457         bool x=!(rs->next());
00458         delete rs;
00459         return x;
00460     }
00461
00462     bool tableExists(const char * tableName){
00463                         std::string table(tableName, strlen(tableName));
00464                         std::string query = "SELECT relname FROM pg_class WHERE upper(relname) =
      upper('"+table+"');";
00465                         mrdb::ResultSet* rs = execQr(query);
00466                         bool result = rs->next();
00467                         delete rs;
00468                         return result;
00469         }
00470
00471     void dropTable(std::string table) {
00472             execQuery("DROP TABLE IF EXISTS "+table+";");
00473     }
00474
00475     bool checkConstraint(std::string op, std::string val, std::string range) {
00476       //  std::cout«"checking: "+val+op+range«std::endl;
00477         if(op.compare("=") == 0)
00478             return (val.compare(range)) == 0;
00479         if(op.compare("<") == 0)
00480             return (val.compare(range)) < 0;
00481         if(op.compare(">") == 0)
00482             return (val.compare(range)) > 0;
00483         if(op.compare("<=") == 0)
00484             return (val.compare(range)) <= 0;
00485         if(op.compare(">=") == 0)
00486             return (val.compare(range)) >= 0;
00487         if(op.compare("!=") == 0 || op.compare("<>") == 0)
00488             return (val.compare(range)) != 0;
00489         if(op.compare("BETWEEN") == 0) {
00490             //range contiene: "const1 AND const2"
00491             unsigned pos = range.find("AND");
00492             std::string const1 = range.substr(0,pos-1);
00493             std::string const2 = range.substr(pos+4);
00494             return ((val.compare(const1)) >= 0 && (val.compare(const2)) <= 0);
00495         }
00496     }
00497
```

```
00498     std::vector<std::string> copy_and_merge(std::vector<std::string> v1,std::vector<std::string> v2) {
00499         std::vector<std::string> v;
00500         for(int i=0; i<v1.size();++i)
00501             v.push_back(v1[i]);
00502         v.push_back(v2[v2.size()-1]);
00503         return v;
00504     }
00505 
00506     void updateMatcher(int seqId,int id_el) {
00507         for(int i=0;i<bvptr[id_el-1]->length();++i){
00508             if(bvptr[id_el-1]->test(i)==true){
00509                     matcher_cache « seqId ;
00510                     for(int j=0;j<hash_vect[i].size();++j)
00511                         matcher_cache « "\t"« hash_vect[i][j] ;
00512                     matcher_cache « std::endl;
00513             }
00514         }
00515     }
00516 
00517     void finalize_matcher(bool at_least_one){
00518         matcher_cache.close();
00519         std::string loadstr = "/tmp/" + pm.tab_result + ".m";
00520         if(at_least_one) {
00521             const std::string& dbms = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00522             if( dbms == "postgres" ) {
00523                 loadstr = "COPY " + pm.tab_result  +"_matcher FROM '/tmp/" + pm.tab_result + ".m'";
00524             }
00525             else  {
00526                 throw MineruleException( MR_ERROR_OPTION_CONFIGURATION,
00527                                          "Option for key mrdb::dbms is not set properly, it is set to
      "+dbms+
00528                                          ", but only 'postgres' is supported." );
00529             }
00530             mrdb::Statement* state = connection.getMRDBConnection()->createStatement();
00531             state->execute(loadstr.c_str());
00532             delete state;
00533         }
00534         loadstr = "/tmp/" + pm.tab_result + ".m";
00535         unlink(loadstr.c_str());
00536     }
00537 
00538     void finalize_seqi(int k) {
00539         seqicache.close();
00540         std::string loadstr = "/tmp/" + pm.tab_result + ".s";
00541         if(k>1){
00542             const std::string& dbms = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00543             if( dbms == "postgres" ) {
00544                 loadstr = "COPY " + pm.tab_result  +"_Seq"+Converter(k).toString()+"(ID_1,ID_2) FROM
      '/tmp/" + pm.tab_result + ".s'";
00545             }
00546             else  {
00547                 throw MineruleException( MR_ERROR_OPTION_CONFIGURATION,
00548                                          "Option for key mrdb::dbms is not set properly, it is set to
      "+dbms+
00549                                          ", but only 'postgres' is supported." );
00550             }
00551             mrdb::Statement* state = connection.getMRDBConnection()->createStatement();
00552             state->execute(loadstr.c_str());
00553             delete state;
00554         }
00555         loadstr = pm.tab_result + "_"+Converter(k).toString()+".s";
00556         unlink(loadstr.c_str());
00557     }
00558 
00559     void reset_seqi_cache(int k) {
00560         std::string temp = "/tmp/" + pm.tab_result + ".s";
00561         seqicache.open(temp.c_str(), std::ofstream::trunc);
00562     }
00563 
00564 
00565     void saveSingleton(int i) {
00566         //seq_count counts the returned sequences (seq_count-1==0->no sequences generated
00567         double sup = bvptr[i-1]->count(true)/(double)grpId_count;
00568         connection.insert(seq_count,i,sup,0,true);//pos=0 && first=true perche' sono singoletti
00569         updateMatcher(seq_count,i);
00570         seq_count++;
00571     }
00572 
00573     void saveSequence(int id, std::vector<std::string> elements, double sup) {
00574         for(int h = 0; h < elements.size(); ++h)
00575             connection.insert(seq_count,Converter(elements[h]).toLong(),sup,h,(h==0));
00576         updateMatcher(seq_count,id);
00577         seq_count++;
00578     }
00579 
00580 
00581     template<typename FwdIterator>
```

```
00582     void deleter(FwdIterator from, FwdIterator to)
00583     {
00584        while ( from != to )
00585        {
00586            delete *from;
00587            from++;
00588        }
00589     }
00590
00591     void show_Results(bool show) {
00592         if(!show)
00593             return;
00594         if(seq_count==1)
00595             MRLog()«"NO SEQUENCES EXTRACTED"«std::endl;
00596         else
00597             printSeq(printSeq1());
00598      }
00599
00600     std::vector<std::string> printSeq1() {
00601
00602         std::vector<std::string> seq1_str;
00603         std::string el_val = "";
00604         mrdb::ResultSet* rs = execQr("SELECT * FROM "+ pm.tab_result + "_Seq1");
00605
00606         MRDebug() « "Seq1 : " « std::endl;
00607
00608         while(rs->next()) {
00609
00610             int id = rs->getInt(1);
00611
00612                 el_val = "";
00613                 for(int i = 0 ; i < pm.ba.size() ; ++i){
00614                     el_val += rs->getString(i+2);
00615                     if(i<pm.ba.size()-1) el_val += ",";
00616                 }
00617                 seq1_str.push_back("[" + el_val + "] ");
00618        }
00619         delete rs;
00620         return seq1_str;
00621     }
00622
00623     void printSeq(std::vector<std::string> seq_ris) {
00624         std::string q;
00625         mrdb::ResultSet* rs;
00626         int id_el, id,id_old;
00627         std::vector<float> sup;
00628
00629         q = "SELECT * FROM " + pm.tab_result + "_Seq_support ORDER BY ID";
00630         rs= execQr(q);
00631         while(rs->next())
00632             sup.push_back(rs->getFloat(2));
00633
00634         delete rs;
00635
00636         q = "SELECT * FROM " + pm.tab_result + "_Seq ORDER BY ID,POS";
00637         rs= execQr(q);
00638         MRDebug() « "Sequences extracted : " « std::endl;
00639         id_old=-1;
00640
00641         while(rs->next()) {
00642             id = rs->getInt(1);
00643             id_el = rs->getInt(2);
00644             if(id>id_old)
00645                 std::cout « std::endl « std::endl «  "ID: " + Converter(id).toString() « + " with
      support: "
00646                             + Converter(round(sup[id-1]*grpId_count)).toString()
00647                             + "/" + Converter(grpId_count).toString() + " elements: " « std::endl;
00648
00649             std::cout « seq_ris[id_el-1]  ;
00650             id_old=id;
00651         }
00652         delete rs;
00653     }
00654
00655 /*****************  UTILITIES   *******************/
00656
00657 void print_singleton_data() {
00658     /*******************/
00659     for(int i = 0; i < bvptr_length; ++i) {
00660             std::cout«"Bitvectors for Seq1 "+Converter(i).toString()+" of ID
      "+Converter(i+1).toString()+":"«std::endl;
00661             operator«(std::cout,*bvptr[i]);
00662             std::cout«std::endl«std::endl;
00663     }
00664     /*******************/
00665     for(int i = 0; i < pos_lists_singleton.size(); ++i) {
00666         std::cout«"POS_LIST for Seq1 "+Converter(i).toString()+" of ID
```

```
      "+Converter(i+1).toString()+":"«std::endl;
00667         std::cout«pos_list_string(pos_lists_singleton[i]);
00668         std::cout«std::endl;
00669     }
00670     /********************/
00671     if(context_dep_BEM)
00672     for(int i = 0; i < constr_vect_singleton.size(); ++i) {
00673         std::cout«"Constr_vectors for Seq1 "+Converter(i).toString()+" of ID
      "+Converter(i+1).toString()+": "«std::endl;
00674         for(int k = 0; k < constr_vect_singleton[i].size(); ++k)
00675             for(int j = 0; j < constr_vect_singleton[i][k]->bem.size(); ++j) {
00676                 operator«(std::cout,(constr_vect_singleton[i][k]->bem[j]));
00677                 if(active_count) {
00678                     std::cout«std::endl«"Mid counters: ";
00679                     for(int h=0;h<constr_vect_singleton[i][k]->mid_counters[j].size();++h)
00680                         std::cout«Converter(constr_vect_singleton[i][k]->mid_counters[j][h].first
00681
      +constr_vect_singleton[i][k]->mid_counters[j][h].second).toString()+", ";
00682                 }
00683             std::cout«std::endl;
00684         }
00685     }
00686     /********************/
00687     if(context_free_BEM)
00688     for(int i = 0; i < bem_vect.size(); ++i) {
00689         std::cout«"BEMvectors for Seq1 "+Converter(i).toString()+" of ID
      "+Converter(i+1).toString()+": "«std::endl;
00690             operator«(std::cout,(*bem_vect[i]));
00691             std::cout«std::endl;
00692     }
00693     /********************/
00694     if(context_free_BEM && active_count)
00695         for(int i=0;i<mid_count.size();++i) {
00696             std::cout«"Mid counters for id: "+ Converter(i+1).toString()«std::endl;
00697             for(int j=0;j<mid_count[i].size();++j)
00698
      std::cout«Converter(mid_count[i][j].first).toString()+","+Converter(mid_count[i][j].second).toString()+"-";
00699             std::cout«std::endl;
00700         }
00701     /**********************/
00702 }
00703
00704 void print_structs_values() {
00705
00706     /********************/
00707     for(int i = 0; i < bvptr.size(); ++i){
00708         std::cout«"Bitvectors for Seq "+Converter(i).toString()+" of ID
      "+Converter(i+1).toString()+":"«std::endl;
00709         operator«(std::cout,*bvptr[i]);
00710         std::cout«std::endl;
00711     }
00712
00713     /********************/
00714     if(context_free_BEM)
00715     for(int i = 0; i < bem_vect.size(); ++i){
00716             std::cout«"BEMvectors of ID "+Converter(i+1).toString()+":"«std::endl;
00717             operator«(std::cout,*bem_vect[i]);
00718             std::cout«std::endl;
00719     }
00720     /********************/
00721     if(context_dep_BEM) {
00722         for(int k=0;k<constr_vect.size();++k) {
00723             std::cout«"Constr_vectors for Seq of ID "+Converter(k+1).toString()+": "«std::endl;
00724             for(int j =0; j<constr_vect[k].size();++j)
00725                 for(int i=0;i<constr_vect[k][j]->bem.size();++i) {
00726                     operator«(std::cout,constr_vect[k][j]->bem[i]);
00727                     if(active_count) {
00728                         std::cout«std::endl«"Mid counters: ";
00729                         for(int h=0;h<constr_vect[k][j]->mid_counters[i].size();++h)
00730                             std::cout«Converter(constr_vect[k][j]->mid_counters[i][h].first
00731
      +constr_vect[k][j]->mid_counters[i][h].second).toString()+", ";
00732
00733                     }
00734                     std::cout«std::endl;
00735                 }
00736         }
00737     }
00738     /********************/
00739     for(int i = 0; i < pos_lists.size(); ++i){
00740         std::cout«"POS_LIST for Seq of ID "+Converter(i+1).toString()+":"«std::endl;
00741         std::cout«pos_list_string(pos_lists[i])«std::endl;
00742        // std::cout«"pos_length: "+Converter(pos_lists[i].size()).toString();
00743         std::cout«std::endl;
00744     }
00745     /************************/
00746     if(context_dep_dist)
```

```
00747     for(int i = 0; i<prefix_pos_lists.size(); ++i){
00748         std::cout« "Prefix-list for id:"+Converter(i+1).toString()«std::endl;
00749         for(int j = 0; j<prefix_pos_lists[i].size(); ++j){
00750             std::cout« "pos entry "+Converter(j).toString()«std::endl;
00751             for(int h = 0; h<prefix_pos_lists[i][j].size(); ++h){
00752                     std::cout«Converter(prefix_pos_lists[i][j][h]).toString()+",";
00753             }
00754             std::cout«std::endl;
00755         }
00756     }
00757     /*************************/
00758     if(context_free_BEM && active_count)
00759         for(int i=0;i<mid_count.size();++i) {
00760             std::cout«"Mid BEM_CF counters for id: "+ Converter(i+1).toString()«std::endl;
00761             for(int j=0;j<mid_count[i].size();++j)
00762
       std::cout«Converter(mid_count[i][j].first).toString()+","+Converter(mid_count[i][j].second).toString()+"-";
00763             std::cout«std::endl;
00764         }
00765     /*********************/
00766 }
00767
00768 };//end class
00769 }//end namespace
00770 #endif // STSMiner_H
```

7.45 /Users/esposito/Software/minerule/include/minerule/Database/Connection.hpp File Reference

```
#include <string.h>
#include <minerule/mrdb/Connection.hpp>
#include <minerule/mrdb/ResultSet.hpp>
#include <minerule/mrdb/Statement.hpp>
#include <minerule/mrdb/ResultSetMetaData.hpp>
#include <minerule/mrdb/DatabaseMetaData.hpp>
#include <minerule/mrdb/PreparedStatement.hpp>
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Database/SourceRowMetaInfo.hpp"
#include <iostream>
#include <stdio.h>
```

Data Structures

- class minerule::Connection

Namespaces

- namespace minerule

7.46 Connection.hpp

Go to the documentation of this file.

```
00001 /* Database/Connection.h*/
00002
00003 #ifndef CONNECTION_H_3H6SFDND
00004 #define CONNECTION_H_3H6SFDND
00005
00006 #include <string.h>
```

```
00007 #include <minerule/mrdb/Connection.hpp>
00008 #include <minerule/mrdb/ResultSet.hpp>
00009 #include <minerule/mrdb/Statement.hpp>
00010 #include <minerule/mrdb/ResultSetMetaData.hpp>
00011 #include <minerule/mrdb/DatabaseMetaData.hpp>
00012 #include <minerule/mrdb/PreparedStatement.hpp>
00013
00014 #include "minerule/Utils/MineruleOptions.hpp"
00015 #include "minerule/Database/SourceRow.hpp"
00016 #include "minerule/Database/SourceRowMetaInfo.hpp"
00017
00018 #include <iostream>
00019 #include <stdio.h>
00020
00021 // #include "minerule/Algorithms/Algorithms.h"
00022
00023 namespace minerule {
00024
00025         class Connection  {
00026         public:
00027         typedef enum { RulesTable, HeadsTable, BodiesTable, SeqTable, SeqElTable, ElemTable }
      TableKind;
00028
00029         private:
00030                 class DBInserter; // forward declaration
00031
00032                 std::string outTableName;
00033                 MinMaxPair bodyCard;
00034                 MinMaxPair headCard;
00035                 DBInserter * dbInserter;
00036                 mrdb::Connection* connection;
00037
00038                 class DBInserter {
00039                 protected:
00040                         Connection& connection;
00041                         mrdb::PreparedStatement* headInserter;
00042                         mrdb::PreparedStatement* bodyInserter;
00043                         mrdb::PreparedStatement* seqInserter;
00044       mrdb::PreparedStatement* seqElInserter;
00045                 public:
00046                         DBInserter(Connection& cc) : connection(cc), headInserter(NULL),
      bodyInserter(NULL), seqInserter(NULL), seqElInserter(NULL) {};
00047                         virtual ~DBInserter() {
00048                                 if(headInserter) delete headInserter;
00049                                 if(bodyInserter) delete bodyInserter;
00050                                 if(seqInserter) delete seqInserter;
00051         if(seqElInserter) delete seqElInserter;
00052                         }
00053                         virtual void setHeadInserter(mrdb::PreparedStatement* inserter) { headInserter
      = inserter; }
00054                         virtual void setBodyInserter(mrdb::PreparedStatement* inserter) { bodyInserter
      = inserter; }
00055                         virtual void setSeqInserter(mrdb::PreparedStatement* inserter) { seqInserter =
      inserter; }
00056       virtual void setSeqElInserter(mrdb::PreparedStatement* inserter) { seqElInserter = inserter; }
00057                         virtual void insert(const ItemSet&, const ItemSet&, double support,  double
      confidence, bool saveBody = true) =0;
00058                         virtual void insert(int seqId, int seqEl, double support, int pos, bool
      saveBody = true) =0;
00059                         virtual void insertHeadBodyElems(TableKind,const ItemSet& elems, size_t
      counter) =0;
00060                         virtual void init() {};
00061                         virtual void finalize() {};
00062                         virtual void finalize(bool b) {};
00063                 };
00064
00065                 class DirectDBInserter : public DBInserter {
00066                 public:
00067                         DirectDBInserter(Connection& cc) : DBInserter(cc) {};
00068                         virtual void insert(const ItemSet&, const ItemSet&, double support, double
      confidence,       bool saveBody = true);
00069                         virtual void insert(int seqId, int seqEl, double support, int pos, bool
      saveSeqId = true);
00070                         virtual void insertHeadBodyElems(TableKind, const ItemSet& elems, size_t
      counter);
00071                         virtual ~DirectDBInserter() {};
00072                 };
00073
00074                 class CachedDBInserter : public DBInserter {
00075                 private:
00076                         std::ofstream outR, outH, outB;
00077                         std::string filename;
00078                 public:
00079                         CachedDBInserter(Connection& cc) : DBInserter(cc) {};
00080                         virtual void insert(const ItemSet&, const ItemSet&, double support, double
      confidence,       bool saveBody = true);
00081                         virtual void insert( int seqId, int seqEl,double support, int pos, bool
```

```
      saveSeqId = true);
00082                         virtual void insertHeadBodyElems(TableKind, const ItemSet& elems, size_t
      counter);
00083                         virtual ~CachedDBInserter() {};
00084                         virtual void init();
00085                         virtual void finalize();
00086                         virtual void finalize(bool seq);
00087                 };
00088
00089         public:
00090                 // Costruttore
00091                 Connection() : bodyCard(1,1000), headCard(1,1000)  /*algoOptions(NULL)*/ {
00092                         if (MineruleOptions::getSharedOptions().getMRDB().getCacheWrites())
00093                                 dbInserter = new CachedDBInserter(*this);
00094                         else dbInserter = new DirectDBInserter(*this);
00095                 }
00096                 // Distruttore
00097                 ~Connection() {       delete dbInserter; }
00098
00099                 void setBodyCardinalities( const MinMaxPair& rhs ) { bodyCard=rhs; }
00100                 void setHeadCardinalities( const MinMaxPair& rhs ) { headCard=rhs; }
00101                 void useMRDBConnection(mrdb::Connection* newConnection);
00102
00103                 mrdb::Connection* getMRDBConnection() { return connection; }
00104
00105                 bool tableExists(const char * tableName);
00106                 void deleteTable(const char * tableName);
00107                 void deleteDestTables();
00108                 void createResultTables(const SourceRowMetaInfo&) ;
00109                 void createResultTables();
00110                 void insert(const char * what);
00111
00112                 // this function should be systematically used in order to
00113                 // write acquired rules to the DB
00114                 // It uses algorithmOptions.get????Cardinalities() to filter rules
00115                 // having wrong cardinalities
00116                 void insert(const ItemSet& body,  const ItemSet& head, double support, double
      confidence, bool saveBody = true) {
00117                         dbInserter->insert(body, head, support, confidence, saveBody);
00118                 }
00119                 //overload for sequences
00120       void insert( int seqId, int seqEl,double support, int pos, bool saveSeqId) {
00121           dbInserter->insert(seqId, seqEl, support, pos, saveSeqId);
00122       }
00123
00124                 void delete_tmp_db();
00125                 void create_tmp_db(int sintax, const SourceRowAttrCollectionDescriptor& body, const
      SourceRowAttrCollectionDescriptor& head);
00126
00127                 void setOutTableName(const std::string& str) {
00128                         outTableName=str;
00129                 }
00130
00131                 void init() {
00132                         dbInserter->init();
00133                 }
00134
00135                 void finalize() { dbInserter->finalize(); }
00136                 void finalize(bool b) { dbInserter->finalize(b); }
00137
00138                 std::string getTableName(TableKind kind) const;
00139         };
00140 } // end namespace
00141
00142 #endif /* end of include guard: CONNECTION_H_3H6SFDND */
```

7.47 /Users/esposito/Software/minerule/include/minerule/mrdb/Connection.hpp File Reference

```
#include "minerule/Utils/MineruleException.hpp"
#include "Statement.hpp"
#include "PreparedStatement.hpp"
#include "SQLException.hpp"
#include "DatabaseMetaData.hpp"
```

Data Structures

- class mrdb::Connection

Namespaces

- namespace mrdb

Typedefs

- typedef mrdb::Connection *(* MRDBConnectFun) (const char *, const char *, const char *)

7.47.1 Typedef Documentation

7.47.1.1 MRDBConnectFun

```
typedef mrdb::Connection *(* MRDBConnectFun) (const char *, const char *, const char *)
```

Definition at line 48 of file Connection.hpp.

7.48 Connection.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_CONNECTION_H_KEKCN3MCK2__
00002 #define MRDB_CONNECTION_H_KEKCN3MCK2__
00003
00004 #include "minerule/Utils/MineruleException.hpp"
00005 #include "Statement.hpp"
00006 #include "PreparedStatement.hpp"
00007 #include "SQLException.hpp"
00008 #include "DatabaseMetaData.hpp"
00009
00010 namespace mrdb {
00011
00012 class Connection {
00013 public:
00014   virtual ~Connection() {}
00015
00022   virtual Statement *createStatement() = 0;
00023
00032   virtual PreparedStatement *prepareStatement(const std::string &sql) = 0;
00033
00038   virtual DatabaseMetaData *getMetaData() = 0;
00039 };
00040
00041 } // namespace
00042
00043
00044 // The MRDBConnectFun type represents factory functions for mrdb::Connection
00045 // objects. This is the type of the entry point of dynamic libraries implementing
00046 // mrdb adapters.
00047 extern "C" {
00048 typedef mrdb::Connection *(*MRDBConnectFun)(const char*,
00049                                             const char*,
00050                                             const char*);
00051 }
00052
00053 #endif /* end of include guard: MRDB_CONNECTION_H_KEKCN3MCK2__ */
```

7.49 /Users/esposito/Software/minerule/include/minerule/Database/ItemType.hpp File Reference

```
#include <iostream>
#include "minerule/Database/SourceRowElement.hpp"
```

Data Structures

- class minerule::ItemType

Namespaces

- namespace minerule

Typedefs

- typedef std::vector< ItemType > minerule::ItemSet

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const ItemType &it)

7.50 ItemType.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __ITEMTYPE_H__
00017 #define __ITEMTYPE_H__
00018
00019 #include <iostream>
00020 #include "minerule/Database/SourceRowElement.hpp"
00021
00022 namespace minerule {
00032         class ItemType; // forward declaration
00033
00037         typedef std::vector<ItemType> ItemSet;
00038
00039
00040         class ItemType {
00041                 friend std::ostream& operator«(std::ostream&,const ItemType&);
00042                 SourceRowElement* el;
00043         public:
00044                 ItemType() : el(NULL) {
00045                 }
00046
00047                 virtual ~ItemType() {
```

```
00048                         if(el!=NULL)
00049                                 delete el;
00050                 }
00051
00052                 ItemType( const ItemType& i)  {
00053                         if( i.el==NULL)
00054                                 el=NULL;
00055                         else
00056                                 el=i.el->copy();
00057                 }
00058
00059                 // This constructor makes a copy of the given srel. The caller is free to delete it
00060                 // as needed.
00061                 ItemType( const SourceRowElement& srel ) {
00062                         el = srel.copy();
00063                 }
00064
00065                 // By calling this constructor one delegates to this class the management of srel.
00066                 ItemType(SourceRowElement* srel) : el(srel) {}
00067
00068                 virtual void setPreparedStatementParameters(mrdb::PreparedStatement* state, size_t
    start_index) const {
00069                         el->setPreparedStatementParameters(state, start_index);
00070                 }
00071
00072                 std::string getFullElementType() const { return el->getFullElementType(); }
00073
00074
00075   // Notice the srel given as argument to this function
00076   // will be managed by this object. In particular it
00077   // will be destroyed when the object is destroyed.
00078                 void setSourceRowElement( SourceRowElement& srel ) {
00079                         el = &srel;
00080                 }
00081
00082                 ItemType& operator=(const ItemType& i) {
00083                         if(el!=NULL)
00084                                 delete el;
00085
00086                         if(i.el!=NULL)
00087                                 el = i.el->copy();
00088                         else
00089                                 el = NULL;
00090
00091                         return *this;
00092                 }
00093
00094                 ItemType& operator=(const SourceRowElement& i) {
00095                         if(el!=NULL)
00096                                 delete el;
00097
00098                         el = i.copy();
00099
00100                         return *this;
00101                 }
00102
00103
00104                 bool operator()(const ItemType& it1, const ItemType& it2) const {
00105                         if(it1.el==NULL)
00106                                 return it2.el!=NULL;
00107
00108                         if(it2.el==NULL)
00109                                 return false;
00110
00111                         return *it1.el < *it2.el;
00112                 }
00113
00114                 bool operator<(const ItemType& it) const {
00115                         if(el==NULL)
00116                                 return it.el!=NULL;
00117
00118                         if(it.el==NULL)
00119                                 return false;
00120
00121                         return *el < *it.el;
00122                 }
00123
00124                 bool operator<(const SourceRowElement& elem) const {
00125                         if(el==NULL)
00126                                 return !elem.empty();
00127
00128                         if(elem.empty())
00129                                 return false;
00130
00131                         return *el < elem;
00132                 }
00133
```

```
00134                 bool operator>(const ItemType& it) const {
00135                         if(el==NULL)
00136                                 return false;
00137
00138                         if(it.el==NULL)
00139                                 return true;
00140
00141                         return *it.el < *el;
00142                 }
00143
00144                 bool operator>(const SourceRowElement& elem) const {
00145                         if(el==NULL)
00146                                 return false;
00147
00148                         if(elem.empty())
00149                                 return true;
00150
00151                         return elem < *el;
00152                 }
00153
00154                 bool operator==(const ItemType& it) const {
00155                         if(el==NULL)
00156                                 return it.el==NULL;
00157
00158                         if(it.el==NULL)
00159                                 return false;
00160
00161                         return *el == *it.el;
00162                 }
00163
00164                 bool operator==(const SourceRowElement& elem) const {
00165                         if(el==NULL)
00166                                 return elem.empty();
00167
00168                         if(elem.empty())
00169                                 return false;
00170
00171                         return *el == elem;
00172                 }
00173
00174                 bool operator!=(const ItemType& it) const {
00175                         if(el==NULL)
00176                                 return it.el!=NULL;
00177
00178                         if(it.el==NULL)
00179                                 return true;
00180
00181                         return *el != *it.el;
00182                 }
00183
00184                 bool operator!=(const SourceRowElement& elem) const {
00185                         if(el==NULL)
00186                                 return !elem.empty();
00187
00188                         if(elem.empty())
00189                                 return true;
00190
00191                         return *el != elem;
00192                 }
00193
00194                 std::string getSQLData() const {
00195                         assert(el!=NULL);
00196                         return el->getSQLData();
00197                 }
00198
00199                 std::string asString() const {
00200                         assert(el!=NULL);
00201                         return el->asString();
00202                 }
00203
00204                 minerule::SourceRowElement& getElement() const {
00205                         if(el==NULL)
00206                                 throw MineruleException(MR_ERROR_INTERNAL, "Accessing a NULL
      element!");
00207                         return *el;
00208                 }
00209
00210                 static bool lessThan( const ItemType&, const ItemType&);
00211
00212
00213         };
00214
00215         inline std::ostream& operator«(std::ostream& os, const ItemType& it) {
00216                 if(it.el==NULL)
00217                         os«"NULL";
00218                 else
00219                         os « *it.el;
```

```
00220
00221                 return os ;
00222         }
00223
00224
00225         inline bool ItemType::lessThan( const ItemType& it1, const ItemType& it2) {
00226                 return it1<it2;
00227         }
00228
00229 } // namespace
00230 #endif
```

7.51 /Users/esposito/Software/minerule/include/minerule/Database/MRResultSet.hpp File Reference

```
#include "minerule/mrdb/PreparedStatement.hpp"
#include "minerule/mrdb/ResultSet.hpp"
```

Data Structures

- class MRResultSetIterator

7.52 MRResultSet.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MRRESULTSETITERATOR_H__
00017 #define __MRRESULTSETITERATOR_H__
00018
00019 #include "minerule/mrdb/PreparedStatement.hpp"
00020 #include "minerule/mrdb/ResultSet.hpp"
00021 // #include "MIndex.hpp"
00022
00023 class MRResultSetIterator {
00024   mrdb::PreparedStatement *query;
00025   mrdb::ResultSet *rs;
00026
00027 public:
00028   MRResultSetIterator(mrdb::PreparedStatement *q) : query(q), rs(NULL) {
00029     rs = query->executeQuery();
00030   }
00031
00032   bool next() { return rs->next(); }
00033
00034   void reset() {
00035     if (rs != NULL) {
00036       delete rs;
00037     }
00038
00039     rs = query->executeQuery();
00040   }
00041
00042   mrdb::ResultSet *getResultSet() { return rs; }
00043 };
00044
00045 #endif
```

7.53 /Users/esposito/Software/minerule/include/minerule/Database/↩ PrepareDataUtils.hpp File Reference

```
#include <string>
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/Optimizer/OptimizedMinerule.hpp"
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/mrdb/Connection.hpp"
```

Data Structures

- class minerule::PrepareDataUtils

Namespaces

- namespace minerule

7.54 PrepareDataUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __PREPAREDATAUTILS_H__
00017 #define __PREPAREDATAUTILS_H__
00018
00019 #include <string>
00020 #include "minerule/Parsers/ParsedMinerule.hpp"
00021 #include "minerule/Database/SourceRowColumnIds.hpp"
00022 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00023 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00024 #include "minerule/mrdb/Connection.hpp"
00025
00026 namespace minerule {
00027
00028 class PrepareDataUtils {
00029 private:
00030   const ParsedMinerule &mr;
00031   SourceTableRequirements sourceTableRequirements;
00032
00033   std::string buildAndList(const list_AND_node *l) const;
00034
00035   std::string buildConditionFilter(const list_OR_node *) const;
00036
00037   std::string createSourceTable() const;
00038
00039   static std::string buildAttrListAlias(const ParsedMinerule::AttrVector &attrs,
00040                                         const std::string &alias = "",
00041                                         bool addColAlias = false);
00042
00043   std::string buildAttrListEquiJoin(const std::string &alias1,
00044                                     const std::string &alias2) const;
00045
```

```
00046 public:
00047   PrepareDataUtils(const ParsedMinerule &m,
00048                    const SourceTableRequirements &requirements)
00049       : mr(m), sourceTableRequirements(requirements) {}
00050
00051   std::string buildBodyTableQuery(SourceRowColumnIds &rowDes,
00052                                   const std::string &condition) const;
00053   std::string buildHeadTableQuery(SourceRowColumnIds &rowDes,
00054                                   const std::string &condition) const;
00055   std::string buildExtendedSourceTableQuery(SourceRowColumnIds &rowDes) const;
00056
00057   std::string buildSourceTableQuery(SourceRowColumnIds &rowDes) const;
00058
00059   static std::string
00060   buildAttrListDescription(const ParsedMinerule::AttrVector &attrs,
00061                            const std::string &alias = "",
00062                            bool addColAlias = false);
00063
00064   static void dropTableIfExists(mrdb::Connection *conn,
00065                                 const std::string &tname);
00066
00077   static size_t
00078   evaluateTotGroups(const ParsedMinerule &pmr);
00079
00080   size_t evaluateTotGroups() const
00081       {
00082     return evaluateTotGroups(mr);
00083   }
00084 };
00085
00086 } // namespace
00087
00088 #endif
```

7.55 /Users/esposito/Software/minerule/include/minerule/Database/SourceRow.hpp File Reference

```
#include <vector>
#include <string.h>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/Database/SourceRowAttribute.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/Database/SourceRowAttributeCollection.hpp"
```

Data Structures

- class minerule::SourceRow

Namespaces

- namespace minerule

Macros

- #define ELEM_OR_EMPTY(elem) if((elem)==NULL) return emptyElement; else return *elem;

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const SourceRow &sr)

7.55.1 Macro Definition Documentation

7.55.1.1 ELEM_OR_EMPTY

```
#define ELEM_OR_EMPTY(
              elem ) if((elem)==NULL) return emptyElement; else return *elem;
```

Definition at line 62 of file SourceRow.hpp.

7.56 SourceRow.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __ITEM_H__
00017 #define __ITEM_H__
00018
00019 #include <vector>
00020 #include <string.h>
00021
00022 #include "minerule/mrdb/ResultSet.hpp"
00023 #include "minerule/Database/SourceRowAttribute.hpp"
00024 #include "minerule/Database/SourceRowColumnIds.hpp"
00025 #include "minerule/Database/SourceRowAttributeCollection.hpp"
00026
00027
00028
00029 namespace minerule {
00030
00037         class SourceRow {
00038                 friend  std::ostream& operator«(std::ostream& os, const SourceRow& item);
00039         private:
00040                 SourceRowElement* group;
00041                 SourceRowElement* clusterBody;
00042                 SourceRowElement* body;
00043                 SourceRowElement* clusterHead;
00044                 SourceRowElement* head;
00045
00046                 /* The element that is returned when the value cannot be retrieved (e.g., when it is
      null)*/
00047                 static const SourceRowEmptyElement emptyElement;
00048
00049         public:
00050                 SourceRow() : group(NULL), clusterBody(NULL), body(NULL), clusterHead(NULL),
      head(NULL) { }
00051     // create a new item as a copy of the current row of the result set
00052                 SourceRow(mrdb::ResultSet* resultSet, const SourceRowColumnIds& srd);
00053
00054     // copy constructor
00055                 SourceRow(const SourceRow& item);
00056
00057
00058                 virtual ~SourceRow();
00059
00060                 void init(mrdb::ResultSet* resultSet,const SourceRowColumnIds& srd);
00061
00062                 #define ELEM_OR_EMPTY(elem) if((elem)==NULL) return emptyElement; else return *elem;
00063
```

```
00064                 // --- GETTERS ---
00065                 const SourceRowElement&
00066                 getGroup() const {
00067                         ELEM_OR_EMPTY(group)
00068                 }
00069
00070                 const SourceRowElement&
00071                 getClusterBody() const {
00072                         ELEM_OR_EMPTY(clusterBody);
00073                 }
00074
00075                 const SourceRowElement&
00076                 getBody() const {
00077                         ELEM_OR_EMPTY(body);
00078                 }
00079
00080                 const SourceRowElement&
00081                 getClusterHead() const {
00082                         ELEM_OR_EMPTY(clusterHead);
00083                 }
00084
00085                 const SourceRowElement&
00086                 getHead() const {
00087                         ELEM_OR_EMPTY(head);
00088                 }
00089         };
00090
00091
00092         std::ostream& operator«(std::ostream& os, const SourceRow& sr);
00093
00094
00095 }; // END NAMESPACE MINERULE
00096
00097
00098 #endif
```

7.57 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttribute.hpp File Reference

```
#include <string.h>
#include <iostream>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/mrdb/Types.hpp"
#include "minerule/Database/SourceRowElement.hpp"
```

Data Structures

- class minerule::SourceRowAttribute
- class minerule::GenericSourceRowAttribute
- class minerule::MemDebugGenericSourceRowAttribute
- class minerule::NumericSourceRowAttribute

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const SourceRowAttribute &attr)

7.58 SourceRowAttribute.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __ITEMATTRIBUTE_H__
00017 #define __ITEMATTRIBUTE_H__
00018
00019 #include <string.h>
00020 #include <iostream>
00021
00022 #include "minerule/mrdb/ResultSet.hpp"
00023 #include "minerule/mrdb/PreparedStatement.hpp"
00024 #include "minerule/mrdb/ResultSetMetaData.hpp"
00025 #include "minerule/mrdb/Types.hpp"
00026
00027 #include "minerule/Database/SourceRowElement.hpp"
00028
00029 namespace minerule {
00036 class SourceRowAttribute : public SourceRowElement {
00037 public:
00044   static SourceRowAttribute *createAttribute(mrdb::ResultSetMetaData *rsmd,
00045                                              mrdb::ResultSet *rs, int elem);
00046
00047   virtual ~SourceRowAttribute() {}
00048
00049   // ---- Methods to be implemented in sub-classes
00053   virtual SourceRowElement *copy() const = 0;
00054
00058   virtual mrdb::Types::SQLType getType() const = 0;
00059
00067   virtual void setValue(mrdb::ResultSet *rs, int col) = 0;
00068
00075   virtual void setValue(const std::string & value) = 0;
00076
00083   virtual int compareTo(const SourceRowAttribute &) const = 0;
00084
00092   virtual std::string asString(const std::string &sep = ",") const = 0;
00093
00094   // INHERITED FROM SOURCE ROW ELEMENT
00095
00097   virtual bool operator()(const SourceRowElement &s1,
00098                           const SourceRowElement &s2) const {
00099     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00100     const SourceRowAttribute &attr2 = dynamic_cast<const SourceRowAttribute &>(s2);
00101
00102     return attr1.compareTo(attr2) < 0;
00103   }
00104
00106   virtual bool operator==(const SourceRowElement &s1) const {
00107     const SourceRowAttribute &attr1 = dynamic_cast<const SourceRowAttribute &>(s1);
00108
00109     return this->compareTo(attr1) == 0;
00110   }
00111
00113   virtual bool empty() const { return false; }
00114
00120   virtual std::string getSQLData() const = 0;
00121
00129   virtual std::ostream &operator«(std::ostream &os) const {
00130     os « asString();
00131     return os;
00132   }
00133 };
00134
00141 class GenericSourceRowAttribute : public SourceRowAttribute {
00142   friend std::ostream &operator«(std::ostream &os,const SourceRowAttribute &ia);
00143
00144 private:
00145   std::string value;
00146   mrdb::Types::SQLType type;
00147
```

```
00148 public:
00149   GenericSourceRowAttribute(mrdb::ResultSet *_rs, int _elem,
00150                             mrdb::Types::SQLType _type)
00151       : value(_rs->getString(_elem)), type(_type) {}
00152
00153   GenericSourceRowAttribute()
00154       : value(""), type((mrdb::Types::SQLType)0)
00155         /*, colId(-1) */ {};
00156
00157   // copy constructor
00158   GenericSourceRowAttribute(const GenericSourceRowAttribute &rhs)
00159       : value(rhs.value), type(rhs.type) /*, colId(rhs.colId)*/ {}
00160
00161   virtual ~GenericSourceRowAttribute() {
00162     //      std::cout « "Generic... destructor" « std::endl;
00163   }
00164
00165   // Implementing AttributeType methods...
00166
00167   virtual SourceRowElement *copy() const {
00168     SourceRowAttribute *newRow = new GenericSourceRowAttribute(*this);
00169     return newRow;
00170   }
00171
00172   virtual void setPreparedStatementParameters(mrdb::PreparedStatement *state,
00173                                               size_t start_index) const {
00174     state->setString(start_index, value);
00175   }
00176
00177   virtual mrdb::Types::SQLType getType() const;
00178
00179
00180   virtual void setValue(mrdb::ResultSet *, int);
00181   virtual void setValue(const std::string &);
00182   virtual int compareTo(const SourceRowAttribute &) const;
00183
00184   virtual std::string asString(const std::string &sep = ",") const;
00185
00186   virtual std::string getSQLData() const;
00187
00188   virtual SourceRowElement &operator=(const SourceRowElement &_rhs) {
00189     const GenericSourceRowAttribute &rhs =
00190         dynamic_cast<const GenericSourceRowAttribute &>(_rhs);
00191     value = rhs.value;
00192     type = rhs.type;
00193     return *this;
00194   }
00195
00199   virtual ElementType getElementType() const { return 'g'; }
00200   virtual void serialize(std::ostream &os) const ;
00201   virtual void deserialize(std::istream &is) ;
00202 };
00203
00209 class MemDebugGenericSourceRowAttribute : public GenericSourceRowAttribute {
00210   static size_t instanceCount;
00211
00212 private:
00213   void incCounter() {
00214     instanceCount++;
00215   }
00216
00217   void decCounter() {
00218     instanceCount--;
00219   }
00220
00221 public:
00222   MemDebugGenericSourceRowAttribute() : GenericSourceRowAttribute() {
00223     incCounter();
00224   }
00225
00226   MemDebugGenericSourceRowAttribute(mrdb::ResultSet *_rs, int _elem,
00227                                     mrdb::Types::SQLType _type)
00228       : GenericSourceRowAttribute(_rs, _elem, _type) {
00229     incCounter();
00230   }
00231
00232   MemDebugGenericSourceRowAttribute(
00233       const MemDebugGenericSourceRowAttribute &rhs)
00234       : GenericSourceRowAttribute(rhs) {
00235     incCounter();
00236   }
00237
00238   virtual ~MemDebugGenericSourceRowAttribute() { decCounter(); }
00239
00240   virtual SourceRowElement *copy() const {
00241     SourceRowAttribute *newRow = new MemDebugGenericSourceRowAttribute(*this);
00242     return newRow;
```

```
00243   }
00244
00245   static size_t getInstanceCounter() { return instanceCount; }
00246 };
00247
00253 class NumericSourceRowAttribute : public SourceRowAttribute {
00254   friend std::ostream &operator«(std::ostream &os,
00255                                  const SourceRowAttribute &ia);
00256
00257 private:
00258   long int value;
00259
00260 public:
00261   const static SourceRowAttribute::ElementType elementType;
00262
00263   NumericSourceRowAttribute(mrdb::ResultSet *_rs, int _elem)
00264       : value(_rs->getInt(_elem)) {}
00265
00266   NumericSourceRowAttribute() : value(0){};
00267
00268   // copy constructor
00269   NumericSourceRowAttribute(const NumericSourceRowAttribute &rhs)
00270       : value(rhs.value) {}
00271
00272   virtual ~NumericSourceRowAttribute() {
00273     //      std::cout « "Numeric... destructor" « std::endl;
00274   }
00275
00276   // Implementing AttributeType methods...
00277
00278   virtual SourceRowElement *copy() const {
00279     SourceRowAttribute *newRow = new NumericSourceRowAttribute(*this);
00280     return newRow;
00281   }
00282
00283   virtual void setPreparedStatementParameters(mrdb::PreparedStatement *state,
00284                                               size_t start_index) const {
00285     state->setLong(start_index, value);
00286   }
00287
00288   virtual mrdb::Types::SQLType getType() const { return mrdb::Types::INTEGER; }
00289
00290   virtual void setValue(mrdb::ResultSet *, int);
00291   virtual void setValue(const std::string &);
00292   virtual int compareTo(const SourceRowAttribute &) const;
00293
00294   virtual std::string asString(const std::string &sep = ",") const;
00295
00296   virtual std::string getSQLData() const;
00297
00298   virtual SourceRowElement &operator=(const SourceRowElement &_rhs) {
00299     const NumericSourceRowAttribute &rhs =
00300         dynamic_cast<const NumericSourceRowAttribute &>(_rhs);
00301     value = rhs.value;
00302     return *this;
00303   }
00304
00308   virtual ElementType getElementType() const { return 'n'; }
00309   virtual void serialize(std::ostream &os) const ;
00310   virtual void deserialize(std::istream &is) ;
00311 };
00312
00313 inline std::ostream &operator«(std::ostream &os,
00314                                const SourceRowAttribute &attr) {
00315   attr.operator«(os);
00316   return os;
00317 }
00318
00319 } // minerule;
00320
00321 #endif
```

7.59 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowAttributeCollection.hpp File Reference

```
#include <vector>
#include <string.h>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
```

```
#include "minerule/Database/SourceRowAttribute.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
```

Data Structures

- class minerule::SourceRowAttributeCollection

Namespaces

- namespace minerule

7.60 SourceRowAttributeCollection.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __SOURCEROWATTRIBUTECOLLECTION_H__
00017 #define __SOURCEROWATTRIBUTECOLLECTION_H__
00018
00019 #include <vector>
00020 #include <string.h>
00021 #include "minerule/mrdb/ResultSet.hpp"
00022 #include "minerule/mrdb/ResultSetMetaData.hpp"
00023
00024 #include "minerule/Database/SourceRowAttribute.hpp"
00025 #include "minerule/Database/SourceRowColumnIds.hpp"
00026
00027
00028 namespace minerule {
00033   class SourceRowAttributeCollection : public SourceRowElement {
00034   private:
00035     typedef std::vector<SourceRowAttribute*> CollectionType;
00036     // storage
00037     CollectionType attributes;
00038
00039     void freeCollection(); // clear the collection
00040
00041     static bool lessThan( const SourceRowAttributeCollection& s1,
00042                           const SourceRowAttributeCollection& s2);
00043   public:
00044     // builds an empty attribute collection
00045     SourceRowAttributeCollection()  {
00046     }
00047
00048     SourceRowAttributeCollection(
00049                                  mrdb::ResultSetMetaData* rsmd,
00050                                  mrdb::ResultSet* rs,
00051                                  std::vector<int> elems);
00052
00053     // copy constructor
00054     SourceRowAttributeCollection(const SourceRowAttributeCollection& rhs);
00055
00056
00057     // destructor
00058     virtual ~SourceRowAttributeCollection() {
00059         freeCollection();
00060     }
00061
```

```
00062     virtual SourceRowElement&
00063       operator=(const SourceRowElement& rhs);
00064
00065     virtual bool operator()(const SourceRowElement& s1,
00066                             const SourceRowElement& s2) const;
00067     virtual bool operator==(const SourceRowElement&) const;
00068
00069     virtual bool empty() const {
00070       return attributes.empty();
00071     }
00072
00073         virtual void setPreparedStatementParameters(mrdb::PreparedStatement* state, size_t
      start_index) const;
00074
00075
00076
00077     virtual std::string getSQLData() const;
00078
00079     virtual std::string asString(const std::string& sep=",") const;
00080
00081     virtual SourceRowElement* copy() const {
00082       return new SourceRowAttributeCollection(*this);
00083     }
00084
00085     virtual std::ostream& operator«(std::ostream& os) const;
00086
00087
00088     virtual ElementType getElementType() const {
00089       return 'A';
00090     }
00091
00092         virtual std::string getFullElementType() const;
00093
00094     virtual void
00095     serialize(std::ostream& os) const ;
00096
00097     virtual void
00098     deserialize(std::istream& is) ;
00099
00100   };
00101 }
00102
00103 #endif
```

7.61 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowColumnIds.hpp File Reference

```
#include <cstdio>
#include <vector>
#include <string>
#include <iterator>
#include <iostream>
```

Data Structures

- class minerule::SourceRowColumnIds

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &o, const SourceRowColumnIds &rowDes)

7.62 SourceRowColumnIds.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __SOURCEROWCOMMON_H__
00017 #define __SOURCEROWCOMMON_H__
00018
00019 #include <cstdio>
00020 #include <vector>
00021 #include <string>
00022 #include <iterator>
00023 #include <iostream>
00024
00025 namespace minerule {
00026
00034 class SourceRowColumnIds {
00035   public:
00036   std::vector<int> groupElems;
00037   std::vector<int> clusterBodyElems;
00038   std::vector<int> bodyElems;
00039   std::vector<int> clusterHeadElems;
00040   std::vector<int> headElems;
00041
00042   SourceRowColumnIds(const SourceRowColumnIds& rhs)
00043           : groupElems(rhs.groupElems), clusterBodyElems(rhs.clusterBodyElems),
00044                 bodyElems(rhs.bodyElems), clusterHeadElems(rhs.clusterHeadElems),
00045                 headElems(rhs.headElems) {}
00046
00047   SourceRowColumnIds() {
00048   }
00049
00050   ~SourceRowColumnIds() {
00051   }
00052
00053
00060
00061   unsigned int setGroupElems(unsigned int start,unsigned int numCols);
00062
00063
00070
00071   unsigned int setClusterBodyElems(unsigned int start, unsigned int numCols);
00072
00079
00080   unsigned int setBodyElems(unsigned int start,unsigned int numCols);
00081
00088
00089   unsigned int setClusterHeadElems(unsigned int start, unsigned int numCols);
00090
00097
00098   unsigned int setHeadElems(unsigned int start, unsigned int numCols);
00099
00100
00101 private:
00112
00113   unsigned int setElems(std::vector<int>& elems, unsigned int start, unsigned int numCols) const;
00114
00115 };
00116
00117 inline std::ostream& operator<<(std::ostream& o, const SourceRowColumnIds& rowDes) {
00118   // o << "test";
00119         // o << "head:";
00120   copy( rowDes.headElems.begin(), rowDes.headElems.end(), std::ostream_iterator<int>(o, " ") );
00121         // o << " - ";
00122         // o << "body:";
00123   copy( rowDes.bodyElems.begin(), rowDes.bodyElems.end(), std::ostream_iterator<int>(o, " ") );
00124         return o;
00125 }
00126
00127 } // namespace
00128
00129 #endif
```

7.63 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowElement.hpp File Reference

```
#include <vector>
#include <iostream>
#include <cassert>
#include <string.h>
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
```

Data Structures

- class minerule::SourceRowElement
- class minerule::SourceRowEmptyElement

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const SourceRowElement &el)

7.64 SourceRowElement.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __SOURCEROWELEMENT_H__
00017 #define __SOURCEROWELEMENT_H__
00018
00019 #include <vector>
00020 #include <iostream>
00021 #include <cassert>
00022 #include <string.h>
00023 #include "minerule/mrdb/ResultSetMetaData.hpp"
00024 #include "minerule/mrdb/ResultSet.hpp"
00025 #include "minerule/Utils/MineruleException.hpp"
00026 #include "minerule/mrdb/PreparedStatement.hpp"
00027
00028 namespace minerule {
00029
00034 class SourceRowElement {
00035
00036 public:
```

```
00037   typedef char ElementType;
00038
00039   static SourceRowElement *createElement(mrdb::ResultSetMetaData *rsmd,
00040                                          mrdb::ResultSet *rs,
00041                                          const std::vector<int> &srd);
00042
00043   virtual ~SourceRowElement() {}
00044
00045   virtual SourceRowElement *copy() const = 0;
00046
00047   // Returns true if s1<s2
00048   virtual bool operator()(const SourceRowElement &s1,
00049                           const SourceRowElement &s2) const = 0;
00050
00051   virtual bool operator==(const SourceRowElement &) const = 0;
00052
00053   virtual bool operator!=(const SourceRowElement &el) const {
00054     return !this->operator==(el);
00055   }
00056
00057   virtual bool operator<(const SourceRowElement &el) const {
00058     return this->operator()(*this, el);
00059   }
00060
00061   //    const CollectionType& getConstAttributes() const=0;
00062
00063   virtual SourceRowElement &operator=(const SourceRowElement &rhs) = 0;
00064
00065   virtual bool empty() const = 0;
00066
00067   virtual std::string asString(const std::string &sep = ",") const = 0;
00068
00069   virtual void setPreparedStatementParameters(mrdb::PreparedStatement *state,
00070                                               size_t start_index) const = 0;
00071
00072   // This function should return a std::string having the sql
00073   // representation of the data in the object (it is used
00074   // by some algorithm to store temporary information about
00075   // the data.
00076
00077   virtual std::string getSQLData() const = 0;
00078
00079   static SourceRowElement *
00080   createElementFromType(ElementType el) ;
00081
00082   virtual ElementType getElementType() const {
00083     throw MineruleException(MR_ERROR_INTERNAL,
00084                             "getElementType() not implemented!");
00085   }
00086
00087   // Returns a std::string description of the type of this element and, in case
00088   // it applies,
00089   // of inner elments. Defaults to a std::string representation of
00090   // getElementType() except
00091   // when the element is composed.
00092   virtual std::string getFullElementType() const {
00093     char chstr[2] = {getElementType(), '\0'};
00094     return std::string(chstr);
00095   }
00096
00097   virtual void serialize(std::ostream &os) const = 0;
00098   virtual void deserialize(std::istream &is) = 0;
00099
00100   static SourceRowElement *deserializeElementFromString(
00101       const std::string &strRepr) ;
00102
00103   static SourceRowElement *
00104   deserializeElementFromResultSet(mrdb::ResultSet *rs,
00105                                   size_t start_index) ;
00106
00107   static void
00108   serializeElementToString(const SourceRowElement &elem,
00109                            std::string &strRepr) ;
00110
00111   virtual std::ostream &operator<<(std::ostream &) const = 0;
00112 };
00113
00114 class SourceRowEmptyElement : public SourceRowElement {
00115
00116 public:
00117   SourceRowEmptyElement() {}
00118
00119   virtual ~SourceRowEmptyElement() {}
00120
00121   virtual SourceRowElement *copy() const { return new SourceRowEmptyElement(); }
00122
00123   virtual void setPreparedStatementParameters(mrdb::PreparedStatement *state,
```

```
00124                                           size_t start_index) const {
00125     throw MineruleException(MR_ERROR_INTERNAL,
00126                             "setPreparedStatementParameters callled on an "
00127                             "empty element. This is a bug! Please report it!");
00128   }
00129
00130   // Returns true if s1<s2
00131   virtual bool operator()(const SourceRowElement &s1,
00132                           const SourceRowElement &s2) const {
00133     const SourceRowEmptyElement *e1 =
00134         dynamic_cast<const SourceRowEmptyElement *>(&s1);
00135
00136     const SourceRowEmptyElement *e2 =
00137         dynamic_cast<const SourceRowEmptyElement *>(&s2);
00138
00139     // if both s1 and s2 are NOT emptyElements, then the
00140     // proper function should be called
00141     if (e1 == NULL && e2 == NULL) {
00142       return s1(s1, s2);
00143       ;
00144     }
00145
00146     // an empty element is always "lesser" than any non
00147     // empty one
00148     if (e1 == NULL)
00149       return true;
00150
00151     // if s1 is empty and s2 is not than s1<s2 is false
00152     // if both are empty then s1==s2 (hence s1<s2 is false)
00153     // in any case we have to return false
00154     return false;
00155   }
00156
00157   virtual bool operator==(const SourceRowElement &s1) const {
00158     const SourceRowEmptyElement *e1 =
00159         dynamic_cast<const SourceRowEmptyElement *>(&s1);
00160
00161     return e1 != NULL;
00162   }
00163
00164   //    const CollectionType& getConstAttributes() const=0;
00165
00166   virtual SourceRowElement &operator=(const SourceRowElement &rhs) {
00167     assert(dynamic_cast<const SourceRowEmptyElement *>(&rhs) != NULL);
00168
00169     return *this;
00170   }
00171
00172   virtual bool empty() const { return false; }
00173
00174   virtual std::string asString(const std::string &sep = ",") const {
00175     return "";
00176   }
00177
00178   virtual std::string getSQLData() const { assert(false); }
00179
00180   virtual std::ostream &operator«(std::ostream &os) const {
00181     os « "(emptyElem)";
00182     return os;
00183   }
00184
00185   virtual ElementType getElementType() const { return '0'; }
00186   virtual void serialize(std::ostream &os) const {
00187     os « " NULL";
00188   }
00189   virtual void deserialize(std::istream &is) {
00190     std::string null;
00191     is » null;
00192     if (null != " NULL")
00193       throw MineruleException(MR_ERROR_INTERNAL,
00194                               "NULL Element expected, but " + null + " found!");
00195   }
00196 };
00197
00198 inline std::ostream &operator«(std::ostream &os, const SourceRowElement &el) {
00199   el.operator«(os);
00200   return os;
00201 }
00202
00203 } // namespace
00204
00205 #endif
```

7.65 /Users/esposito/Software/minerule/include/minerule/Database/SourceRowMetaInfo.hpp File Reference

```
#include <string>
#include <string.h>
#include <vector>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/mrdb/Connection.hpp"
```

Data Structures

- class minerule::SourceRowAttrCollectionDescriptor
- class minerule::SourceRowMetaInfo

Namespaces

- namespace minerule

7.66 SourceRowMetaInfo.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __SOURCEROWDESCRIPTOR_H__
00017 #define __SOURCEROWDESCRIPTOR_H__
00018
00019 #include <string>
00020 #include <string.h>
00021 #include <vector>
00022 #include "minerule/mrdb/ResultSet.hpp"
00023
00024 #include "minerule/Database/SourceRowColumnIds.hpp"
00025 #include "minerule/Parsers/ParsedMinerule.hpp"
00026 #include "minerule/mrdb/Connection.hpp"
00027
00028
00029 namespace minerule {
00030
00031         class SourceRowAttrCollectionDescriptor {
00032                 std::string dataDefinition;
00033                 std::string columnNames;
00034                 int columnsCount;
00035
00036                 void setColumnNames(mrdb::ResultSet* rs, const std::vector<int>& collectionElems);
00037
00038                 std::string dataDefinitionForElem(mrdb::ResultSet* rs, int elem);
00039
00040                 void setDataDefinition(mrdb::ResultSet* rs, const std::vector<int>& collectionElems);
00041
```

```
00042         public:
00043                 SourceRowAttrCollectionDescriptor() {}
00044                 SourceRowAttrCollectionDescriptor( mrdb::ResultSet* rs, const std::vector<int>&
      collectionElems);
00045
00046                 void init(mrdb::ResultSet* rs, const std::vector<int>& collectionElems );
00047
00048                 const std::string& getSQLDataDefinition() const;
00049                 const std::string& getSQLColumnNames() const;
00050                 unsigned int       getColumnsCount() const { return columnsCount; }
00051                 std::string questionMarks() const;
00052         };
00053
00054         class SourceRowMetaInfo {
00055         private:
00056                 SourceRowAttrCollectionDescriptor group;
00057                 SourceRowAttrCollectionDescriptor clusterBody;
00058                 SourceRowAttrCollectionDescriptor body;
00059                 SourceRowAttrCollectionDescriptor clusterHead;
00060                 SourceRowAttrCollectionDescriptor head;
00061
00062         public:
00063                 // Builds a SourceRowMetaInfo from an SourceRowColumnIds and
00064                 // the source dataset
00065                 SourceRowMetaInfo(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes);
00066
00067                 // Builds a SourceRowMetaInfo from a ParsedMinerule
00068                 SourceRowMetaInfo(mrdb::Connection*, const ParsedMinerule& minerule);
00069
00070
00071                 const SourceRowAttrCollectionDescriptor& getGroup() const {
00072                         return group;
00073                 }
00074
00075                 const SourceRowAttrCollectionDescriptor& getClusterBody() const {
00076                         return clusterBody;
00077                 }
00078
00079                 const SourceRowAttrCollectionDescriptor& getBody() const {
00080                         return body;
00081                 }
00082
00083                 const SourceRowAttrCollectionDescriptor& getClusterHead() const {
00084                         return clusterHead;
00085                 }
00086
00087                 const SourceRowAttrCollectionDescriptor& getHead() const {
00088                         return head;
00089                 }
00090         };
00091
00092 }
00093
00094 #endif
```

7.67 /Users/esposito/Software/minerule/include/minerule/Database/SourceTable.hpp File Reference

```
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
```

Data Structures

- class minerule::SourceTable
- class minerule::SourceTable::Iterator

Namespaces

- namespace minerule

7.68 SourceTable.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef SOURCETABLE_H_GE1S8NL9
00017 #define SOURCETABLE_H_GE1S8NL9
00018
00019 #include "minerule/Database/SourceRow.hpp"
00020 #include "minerule/Database/PrepareDataUtils.hpp"
00021 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00022
00023 namespace minerule {
00024 class SourceTable {
00025 public:
00026   typedef enum { BodyIterator, HeadIterator, FullIterator } IteratorKind;
00027
00028   class Iterator {
00029     friend class SourceTable;
00030
00031   private:
00032     mrdb::ResultSet *_resultSet;
00033     SourceRowColumnIds _columnIds;
00034     SourceRow *_sourceRow;
00035     size_t _rowCounter;
00036
00037     Iterator(mrdb::ResultSet *resultSet, const SourceRowColumnIds &columnIds)
00038         : _resultSet(resultSet), _columnIds(columnIds),
00039           _sourceRow(new SourceRow()), _rowCounter(0) {
00040
00041       assert(_resultSet != NULL);
00042       if (_resultSet->isBeforeFirst()) {
00043         next();
00044       }
00045     }
00046
00047   public:
00048     Iterator() : _resultSet(NULL), _columnIds(), _sourceRow(NULL) { }
00049     Iterator(const Iterator &it)
00050         : _resultSet(it._resultSet), _columnIds(it._columnIds),
00051           _sourceRow(it._sourceRow) {  }
00052
00053     virtual ~Iterator() {
00054     }
00055
00056     bool next();
00057     bool isAfterLast() const;
00058
00059     SourceRow *get() {
00060       assert(_sourceRow != NULL);
00061       return _sourceRow;
00062     }
00063     SourceRow *operator->() { return get(); }
00064
00065     Iterator &operator++() {
00066       next();
00067       return *this;
00068     }
00069   };
00070
00071   SourceTable(const MiningAlgorithm &algorithm)
00072       : _minerule(algorithm.optimizedMinerule().getParsedMinerule()),
00073         _sourceTableRequirements(algorithm.sourceTableRequirements()),
00074         _pdu(_minerule, _sourceTableRequirements), _usesCrossProduct(false),
00075         _bodyStatement(NULL), _headStatement(NULL), _fullStatement(NULL) {
00076     init();
00077   };
00078
00079   SourceTable(const ParsedMinerule &minerule,
00080               const SourceTableRequirements &requirements)
00081       : _minerule(minerule), _sourceTableRequirements(requirements),
00082         _pdu(_minerule, _sourceTableRequirements), _usesCrossProduct(false),
```

```
00083                                 _bodyStatement(NULL), _headStatement(NULL), _fullStatement(NULL)  {
00084     init();
00085   }
00086
00087   virtual ~SourceTable();
00088
00089   bool usesCrossProduct() const { return _usesCrossProduct; }
00090
00091   void init();
00092   size_t getTotGroups();
00093
00094   Iterator newIterator(IteratorKind);
00095
00096 private:
00097   /* data */
00098   const ParsedMinerule &_minerule;
00099   SourceTableRequirements _sourceTableRequirements;
00100   PrepareDataUtils _pdu;
00101   bool _usesCrossProduct;
00102
00103   SourceRowColumnIds _columnIds;
00104   mrdb::PreparedStatement *_bodyStatement;
00105   mrdb::PreparedStatement *_headStatement;
00106   mrdb::PreparedStatement *_fullStatement;
00107
00108   std::vector<mrdb::ResultSet *> _managedResults;
00109
00110   /* methods */
00111   void initBodyHeadResultSets();
00112   void initFullResultSet();
00113 };
00114
00115 } // namespace
00116
00117 #endif /* end of include guard: SOURCETABLE_H_GE1S8NL9 */
```

7.69 /Users/esposito/Software/minerule/include/minerule/Database/SourceTableRequirements.hpp File Reference

Data Structures

- class minerule::SourceTableRequirements

Namespaces

- namespace minerule

7.70 SourceTableRequirements.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef SOURCETABLEREQUIREMENTS_H_E3HLQ5I6
00017 #define SOURCETABLEREQUIREMENTS_H_E3HLQ5I6
00018
00019 namespace minerule {
```

```
00020
00021         // Stores the information about how the source table needs to be constructed.
00022         // Usage example:
00023         //              /* Initialize it, by or-ing the constants SortedGids and CrossProducts */
00024         //              SourceTableRequirements reqs( SourceTableRequirements::SortedGids );
00025         //
00026         //              /* access to the requirements using the provided accessors */
00027         //              reqs.sortedGids();                      /* returns true */
00028         //              reqs.crossProduct();            /* returns false */
00029         class SourceTableRequirements {
00030         public:
00031                 typedef enum { SortedGids = 1, CrossProduct = 2 } Requirements;
00032                 typedef unsigned short RequirementsSet;
00033
00034                 SourceTableRequirements() : _requirements(0) {}
00035                 SourceTableRequirements(RequirementsSet requirements) : _requirements(requirements) {
      }
00036                 virtual ~SourceTableRequirements () {}
00037
00038                 bool sortedGids()     const { return _requirements & SortedGids;    }
00039                 bool crossProduct() const { return _requirements & CrossProduct; }
00040
00041                 void set(RequirementsSet requirements) { _requirements = requirements; }
00042         private:
00043                 RequirementsSet _requirements;
00044         };
00045
00046 }
00047
00048 #endif /* end of include guard: SOURCETABLEREQUIREMENTS_H_E3HLQ5I6 */
```

7.71 /Users/esposito/Software/minerule/include/minerule/Database/↩ Transaction.hpp File Reference

```
#include "minerule/Database/SourceTable.hpp"
```

Data Structures

- class minerule::TransactionBase< SetType >
- class minerule::ItemTransaction< ItemSetType >
- class minerule::RuleTransaction< RuleSetType >

Namespaces

- namespace minerule

7.72 Transaction.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it) and Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef ITEMSETLOADER_H_J9K5VPNF
00017 #define ITEMSETLOADER_H_J9K5VPNF
00018
00019 #include "minerule/Database/SourceTable.hpp"
00020
00021 namespace minerule {
00022         // ------------------------------------------------------------------------------
00023         // Classes declared here load source tables into memory in group sized chunks.
00024         // ------------------------------------------------------------------------------
00025
00026         // Base class. Provides the findGid method.
00027         template <class SetType>
00028         class TransactionBase : public SetType {
00029         public:
00030                 TransactionBase() : SetType() {}
00031
00032                 static bool findGid(ItemType& gid, SourceTable::Iterator& it) {
00033                         while( !it.isAfterLast() && gid > it->getGroup() ) {
00034                                 ++it;
00035                         }
00036
00037                         return !it.isAfterLast() && gid == it->getGroup();
00038                 }
00039         };
00040
00041
00042         // Loader for result set organized as item lists (i.e., the underlying
00043         // query does *not* need a dataset in the form of a joint table that
00044         // differentiates between body and head attributes.
00045         template <class ItemSetType>
00046         class ItemTransaction : public TransactionBase<ItemSetType> {
00047         public:
00048                 ItemTransaction() : TransactionBase<ItemSetType>() {}
00049
00050                 void loadBody(ItemType& gid, SourceTable::Iterator& it) {
00051                         while (!it.isAfterLast() && gid == it->getGroup()) {
00052
00053                                 ItemSetType::insert(it->getBody());
00054                                 ++it;
00055                         }
00056                 }
00057
00058                 void loadHead(ItemType& gid, SourceTable::Iterator& it) {
00059                         while (!it.isAfterLast() && gid == it->getGroup()) {
00060
00061                                 ItemSetType::insert(it->getHead());
00062                                 ++it;
00063                         }
00064                 }
00065         };
00066
00067         // Loader for result set organized as item lists (i.e., the underlying
00068         // query needs a dataset in the form of a joint table that differentiates between
00069         // body and head attributes.
00070         template <class RuleSetType>
00071         class RuleTransaction : public TransactionBase<RuleSetType> {
00072         public:
00073                 RuleTransaction() : TransactionBase<RuleSetType>() {}
00074
00075             void load(ItemType& gid, SourceTable::Iterator& it) {
00076                                 while (!it.isAfterLast() && gid == it->getGroup()) {
00077
      RuleSetType::insert(RuleSetType::end(),std::pair<ItemType,ItemType>(it->getBody(),it->getHead()));
00078                                         ++it;
00079                                 }
00080             }
00081         };
00082
00083
00084 }
00085
00086 #endif /* end of include guard: ITEMSETLOADER_H_J9K5VPNF */
```

7.73 /Users/esposito/Software/minerule/include/minerule/mrdb/↩ DatabaseMetaData.hpp File Reference

```
#include <string>
#include "ResultSet.hpp"
```

```
#include "minerule/mrdb/Types.hpp"
```

Data Structures

- class mrdb::DatabaseMetaData

Namespaces

- namespace mrdb

7.74 DatabaseMetaData.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_DATABASE_META_DATA_HPP_J33CHCYHY__
00002 #define MRDB_DATABASE_META_DATA_HPP_J33CHCYHY__
00003
00004 #include <string>
00005
00006 #include "ResultSet.hpp"
00007 #include "minerule/mrdb/Types.hpp"
00008
00009 namespace mrdb {
00010   class DatabaseMetaData {
00011   public:
00012     virtual ~DatabaseMetaData () {}
00013
00019     virtual ResultSet* getTables(const std::string& tableNamePattern) = 0;
00020
00024     virtual Types::SQLType getColumnType(const std::string& tableName, const std::string& columnName)
      = 0;
00025
00031     virtual ResultSet* getColumns() = 0;
00032   };
00033 }
00034
00035
00036 #endif /* end of include guard: MRDB_DATABASE_META_DATA_HPP_J33CHCYHY__ */
```

7.75 /Users/esposito/Software/minerule/include/minerule/mrdb/↩ PreparedStatement.hpp File Reference

```
#include "ResultSet.hpp"
#include "SQLException.hpp"
#include "minerule/Utils/MineruleException.hpp"
```

Data Structures

- class mrdb::PreparedStatement

Namespaces

- namespace mrdb

7.76 PreparedStatement.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_PREPARED_STATEMENT_H_K3I3JD82G__
00002 #define MRDB_PREPARED_STATEMENT_H_K3I3JD82G__
00003
00004 #include "ResultSet.hpp"
00005 #include "SQLException.hpp"
00006 #include "minerule/Utils/MineruleException.hpp"
00007
00008 namespace mrdb {
00009
00010   class PreparedStatement {
00011   public:
00012     virtual ~PreparedStatement() {}
00013
00019     virtual bool execute() = 0;
00020
00028     virtual ResultSet* executeQuery() = 0;
00029
00033     virtual void setDouble (int idx, double val) = 0;
00034
00038     virtual void setInt (int idx, int val) = 0;
00039
00043     virtual void setString (int idx, const std::string &val) = 0;
00044
00048     virtual void setLong (int idx, long val) = 0;
00049   };
00050
00051 }
00052
00053 #endif /* end of include guard: MRDB_PREPARED_STATEMENT_H_K3I3JD82G */
```

7.77 /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSet.hpp File Reference

```
#include <string>
#include "ResultSetMetaData.hpp"
```

Data Structures

- class mrdb::ResultSet

Namespaces

- namespace mrdb

7.78 ResultSet.hpp

Go to the documentation of this file.

```
00001 #ifndef RESULT_SET_H_JKEJNCI3L19__
00002 #define RESULT_SET_H_JKEJNCI3L19__
00003
00004 #include <string>
00005 #include "ResultSetMetaData.hpp"
00006
00007 namespace mrdb {
00008 class ResultSet {
00009 public:
00010   virtual ~ResultSet() {}
00011
00013   virtual bool isBeforeFirst() const = 0;
```

```
00014
00016   virtual bool isAfterLast() const = 0;
00017
00022   virtual bool next() = 0;
00023
00026   virtual bool getBoolean(int columnIndex) = 0;
00027
00030   virtual float getFloat(int columnIndex) = 0;
00031
00034   virtual double getDouble(int columnIndex) = 0;
00035
00038   virtual int getInt(int columnIndex) = 0;
00039
00042   virtual long getLong(int columnIndex) = 0;
00043
00046   virtual std::string getString(int columnIndex) = 0;
00047
00051   virtual mrdb::ResultSetMetaData* getMetaData() = 0;
00052 };
00053 }
00054
00055 #endif /* end of include guard: RESULT_SET_H_JKEJNCI3L19__ */
```

7.79 /Users/esposito/Software/minerule/include/minerule/mrdb/ResultSetMetaData.hpp File Reference

```
#include <string>
```

Data Structures

- class mrdb::ResultSetMetaData

Namespaces

- namespace mrdb

7.80 ResultSetMetaData.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_RESULTSETMETADATA_HPP_3MNCH3KCU32B__
00002 #define MRDB_RESULTSETMETADATA_HPP_3MNCH3KCU32B__
00003
00004 #include <string>
00005
00006 namespace mrdb {
00007 class ResultSetMetaData {
00008 public:
00009   virtual ~ResultSetMetaData() {}
00010
00012   virtual int getColumnCount () const = 0;
00013
00016   virtual std::string getColumnName(int column) const = 0;
00017
00020   virtual int getColumnType(int column) const = 0;
00021
00024   virtual int getPrecision (int column) const = 0;
00025
00028   virtual int getScale (int column) const = 0;
00029
00032   virtual std::string getColumnTypeName (int column) = 0;
00033
00034 };
00035 }
00036
00037 #endif /* end of include guard: MRDB_RESULTSETMETADATA_HPP_3MNCH3KCU32B__ */
```

7.81 /Users/esposito/Software/minerule/include/minerule/mrdb/SQLException.hpp File Reference

```
#include <stdexcept>
```

Data Structures

- class mrdb::SQLException

Namespaces

- namespace mrdb

Macros

- #define _NOEXCEPT throw()

7.81.1 Macro Definition Documentation

7.81.1.1 _NOEXCEPT

```
#define _NOEXCEPT throw()
```

Definition at line 7 of file SQLException.hpp.

7.82 SQLException.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_SQLEXCEPTION_H_MNG3MX83__
00002 #define MRDB_SQLEXCEPTION_H_MNG3MX83__
00003
00004 #include <stdexcept>
00005
00006 #ifndef _NOEXCEPT
00007 #define _NOEXCEPT throw()
00008 #endif
00009
00010 namespace mrdb {
00011
00012 class SQLException : public std::runtime_error {
00013 public:
00014   SQLException(const std::string &message) : std::runtime_error(message){};
00015   virtual ~SQLException() _NOEXCEPT {}
00016 };
00017
00018 }
00019
00020 #endif /* end of include guard: MRDB_SQLEXCEPTION_H_MNG3MX83__ */
```

7.83 /Users/esposito/Software/minerule/include/minerule/mrdb/Statement.hpp File Reference

```
#include "minerule/Utils/MineruleException.hpp"
#include "ResultSet.hpp"
#include "SQLException.hpp"
```

Data Structures

- class mrdb::Statement

Namespaces

- namespace mrdb

7.84 Statement.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_STATEMENT_H_KDJNMCKELKAJC__
00002 #define MRDB_STATEMENT_H_KDJNMCKELKAJC__
00003
00004 #include "minerule/Utils/MineruleException.hpp"
00005 #include "ResultSet.hpp"
00006 #include "SQLException.hpp"
00007
00008
00009 namespace mrdb {
00010
00011 class Statement {
00012 public:
00013   virtual ~Statement() {}
00014
00021   virtual ResultSet *executeQuery(const std::string &sql) = 0;
00022
00028   virtual bool execute(const std::string &sql) = 0;
00029 };
00030
00031 }
00032
00033 #endif /* end of include guard: MRDB_STATEMENT_H_KDJNMCKELKAJC__ */
```

7.85 /Users/esposito/Software/minerule/include/minerule/mrdb/Types.hpp File Reference

Data Structures

- struct mrdb::Types

Namespaces

- namespace mrdb

7.86 Types.hpp

Go to the documentation of this file.

```
00001 #ifndef MRDB_TYPES_HPP_JACKJHCLKAEJCE__
00002 #define MRDB_TYPES_HPP_JACKJHCLKAEJCE__
00003
00004 // The following list of type definition is the same as the one defined by
00005 // libmrdb++. The actual values are taken from unixmrdb header files.
00006
00007 namespace mrdb {
00008 struct Types {
00009   enum SQLType {
00010     BIGINT = -5,
00011     BINARY = -2,
00012     BIT = -7,
00013     CHAR = 1,
00014     DATE = 9,
00015     DECIMAL = 3,
00016     DOUBLE = 8,
00017     FLOAT = 6,
00018     INTEGER = 4,
00019     LONGVARBINARY = -4,
00020     LONGVARCHAR = -1,
00021     NUMERIC = 2,
00022     REAL = 7,
00023     SMALLINT = 5,
00024     TIME = 10,
00025     TIMESTAMP = 11,
00026     TINYINT = -6,
00027     VARBINARY = -3,
00028     VARCHAR = 12
00029   };
00030 };
00031 }
00032
00033 #endif /* end of include guard: MRDB_TYPES_HPP_JACKJHCLKAEJCE__ */
```

7.87 /Users/esposito/Software/minerule/include/minerule/Optimizer/↩ CatalogueInstaller.hpp File Reference

```
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/Utils/MineruleException.hpp"
```

Data Structures

- class minerule::CatalogueInstaller

Namespaces

- namespace minerule

7.88 CatalogueInstaller.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
```

```
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef CATALOGUEINSTALLER_H_1YBJZ8TY
00017 #define CATALOGUEINSTALLER_H_1YBJZ8TY
00018
00019 #include "minerule/mrdb/Connection.hpp"
00020 #include "minerule/mrdb/Statement.hpp"
00021 #include "minerule/Utils/MineruleException.hpp"
00022
00023 namespace minerule {
00024         class CatalogueInstaller {
00025         protected:
00026                 mrdb::Connection* _connection;
00027                 mrdb::Statement* _statement;
00028         public:
00029                 typedef enum { MySql, Postgres } SupportedDbms;
00030
00031                 CatalogueInstaller();
00032                 virtual ~CatalogueInstaller() {
00033                         delete _statement;
00034                 };
00035
00036                 static CatalogueInstaller* newInstaller(SupportedDbms dbms);
00037                 static CatalogueInstaller* newInstaller();
00038
00039                 virtual void installMRQuery()                    = 0;
00040                 virtual void installMRAttList()                 = 0;
00041                 virtual void installMREqKeys()                   = 0;
00042                 virtual void installMREqKeysCol()              = 0;
00043                 virtual void installMRDepFun()                   = 0;
00044                 virtual void installMRDepFunCol()              = 0;
00045                 virtual void installMRAutoincrement()      = 0;
00046                 virtual void initializeAutoincrement()    = 0;
00047
00048                 virtual void dropMRQuery()                     = 0;
00049                 virtual void dropMRAttList()                 = 0;
00050                 virtual void dropMREqKeys()                    = 0;
00051                 virtual void dropMREqKeysCol()             = 0;
00052                 virtual void dropMRDepFun()                    = 0;
00053                 virtual void dropMRDepFunCol()             = 0;
00054                 virtual void dropMRAutoincrement()    = 0;
00055
00056
00057                 virtual void install() {
00058                         uninstall();
00059
00060                         installMRQuery();
00061                         installMRAttList();
00062                         installMREqKeys();
00063                         installMREqKeysCol();
00064                         installMRDepFun();
00065                         installMRDepFunCol();
00066                         installMRAutoincrement();
00067                         initializeAutoincrement();
00068                 }
00069
00070                 virtual void uninstall() {
00071                         dropMRQuery();
00072                         dropMRAttList();
00073                         dropMREqKeys();
00074                         dropMREqKeysCol();
00075                         dropMRDepFun();
00076                         dropMRDepFunCol();
00077                         dropMRAutoincrement();
00078                 }
00079         };
00080 }
00081 #endif /* end of include guard: CATALOGUEINSTALLER_H_1YBJZ8TY */
```

7.89 /Users/esposito/Software/minerule/include/minerule/Optimizer/↩ GAQueryCombinator.hpp File Reference

```
#include <iostream>
#include <ga/GA1DBinStrGenome.h>
```

```
#include <ga/GASimpleGA.h>
#include "minerule/PredicateUtils/Predicate.hpp"
```

Data Structures

- class minerule::TimeOutException
- class minerule::GAQueryCombinator

Namespaces

- namespace minerule

7.90 GAQueryCombinator.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <ga/GA1DBinStrGenome.h>
00018 #include <ga/GASimpleGA.h>
00019 #include "minerule/PredicateUtils/Predicate.hpp"
00020
00046 namespace minerule {
00047   class TimeOutException : public std::exception {
00048   };
00049
00050
00072   class GAQueryCombinator {
00073     float  startingTime;
00074
00075   public:
00079     typedef std::vector<size_t> QueryAndList;
00083     typedef std::vector<QueryAndList> QueryOrList;
00084   private:
00088     Predicate& mr_target;
00092     std::vector<Predicate>& mr_candidates;
00093
00098     QueryOrList result;
00099
00104     std::string tab_source;
00105
00110     float  timeOut;
00111
00117     size_t maxDisjuncts;
00118
00124     size_t maxQueries;
00125
00131     size_t maxDistinctPredicates;
00132
00145     void
00146       buildAssociatedPredicate(const GAGenome& g,
00147                                           Predicate& p,
00148                                           size_t& numUsedDisjuncts,
00149                                           size_t& numUsedQueries,
00150                                           size_t& numAssertedBits) const;
00151
00156     void buildResult(const GAGeneticAlgorithm& ga);
```

```
00157   public:
00172     GAQueryCombinator(Predicate& _mr_target, std::vector<Predicate>& _mr_candidates)
00173       : mr_target(_mr_target),
00174         mr_candidates(_mr_candidates),
00175         timeOut(4.0),
00176         maxDisjuncts(3),
00177         maxQueries(5),
00178         maxDistinctPredicates(10) {}
00179
00180     virtual ~GAQueryCombinator() {}
00181
00189    static float evaluator(GAGenome& g);
00190
00191
00195     static GABoolean
00196       TerminationCriterion(GAGeneticAlgorithm & ga);
00197
00202     void evolve() ;
00203
00207     const QueryOrList& getResult() const;
00208
00209     //
00210     // -- Parameters
00211     //
00212
00213     float getTimeOutThreshold() const { return timeOut; }
00214     size_t getMaxDisjuncts() const {return maxDisjuncts;}
00215     size_t getMaxQueries() const {return maxQueries;}
00216     size_t getMaxDistinctPredicates() const {return maxDistinctPredicates;}
00217
00218     void setMaxDisjuncts(size_t m) { maxDisjuncts=m; }
00219     void setMaxQueries(size_t m) { maxQueries=m; }
00220     void setMaxDistinctPredicates(size_t m) { maxDistinctPredicates=m; }
00221     void setTimeOutThreshold(float to) { timeOut=to; }
00222   };
00223
00224
00225 }
00226
00227
00228
```

7.91 /Users/esposito/Software/minerule/include/minerule/Optimizer/↩ Installers/MySqlCatalogueInstaller.hpp File Reference

```
#include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
```

Data Structures

- class minerule::MySqlCatalogueInstaller

Namespaces

- namespace minerule

7.92 MySqlCatalogueInstaller.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
```

```
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef MYSQLCATALOGUEINSTALLER_H_3QIF6B9O
00017 #define MYSQLCATALOGUEINSTALLER_H_3QIF6B9O
00018
00019 #include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
00020
00021 namespace minerule {
00022
00023         // right now the postgres installer works also for mysql.
00024         // so the mysql installer simply inherits from it.
00025         class MySqlCatalogueInstaller : public PostgresCatalogueInstaller {
00026         public:
00027                 MySqlCatalogueInstaller() : PostgresCatalogueInstaller() {} ;
00028                 virtual ~MySqlCatalogueInstaller () {};
00029         };
00030 }
00031
00032 #endif /* end of include guard: MYSQLCATALOGUEINSTALLER_H_3QIF6B9O */
```

7.93 /Users/esposito/Software/minerule/include/minerule/Optimizer/↩ Installers/PostgresCatalogueInstaller.hpp File Reference

```
#include "minerule/Optimizer/CatalogueInstaller.hpp"
```

Data Structures

- class minerule::PostgresCatalogueInstaller

Namespaces

- namespace minerule

7.94 PostgresCatalogueInstaller.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef POSTGRESCATALOGUEINSTALLER_H_M0QRV1OU
00017 #define POSTGRESCATALOGUEINSTALLER_H_M0QRV1OU
00018
00019 #include "minerule/Optimizer/CatalogueInstaller.hpp"
00020
00021 namespace minerule {
```

```
00022         class PostgresCatalogueInstaller : public CatalogueInstaller {
00023         public:
00024                 PostgresCatalogueInstaller() : CatalogueInstaller() {}
00025                 virtual ~PostgresCatalogueInstaller () {};
00026 
00027                 virtual void installMRQuery() ;
00028                 virtual void installMRAttList() ;
00029                 virtual void installMREqKeys() ;
00030                 virtual void installMREqKeysCol() ;
00031                 virtual void installMRDepFun() ;
00032                 virtual void installMRDepFunCol() ;
00033                 virtual void installMRAutoincrement() ;
00034                 virtual void initializeAutoincrement() ;
00035 
00036                 virtual void dropMRQuery() ;
00037                 virtual void dropMRAttList() ;
00038                 virtual void dropMREqKeys() ;
00039                 virtual void dropMREqKeysCol() ;
00040                 virtual void dropMRDepFun() ;
00041                 virtual void dropMRDepFunCol() ;
00042                 virtual void dropMRAutoincrement() ;
00043 
00044         };
00045 
00046 } /* minerule */
00047 
00048 #endif /* end of include guard: POSTGRESCATALOGUEINSTALLER_H_M0QRV1OU */
```

7.95 /Users/esposito/Software/minerule/include/minerule/Optimizer/↩ OptimizedMinerule.hpp File Reference

```
#include "minerule/mrdb/Connection.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Optimizer/GAQueryCombinator.hpp"
```

Data Structures

- class minerule::OptimizedMinerule
- class minerule::OptimizedMinerule::OptimizationInfo

Namespaces

- namespace minerule

7.96 OptimizedMinerule.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MROPTIMIZER_H__
00017 #define __MROPTIMIZER_H__
00018
00019 #include "minerule/mrdb/Connection.hpp"
00020 #include "minerule/Parsers/ParsedMinerule.hpp"
00021 #include "minerule/Database/SourceRow.hpp"
00022 #include "minerule/Optimizer/GAQueryCombinator.hpp"
00023
00024
00025 namespace minerule {
00026
00027   class OptimizedMinerule  {
00028   public:
00029     typedef enum  {
00030       None,            // No relationship can be found
00031       Dominance,       // There exists a query which dominate the current one
00032       Inclusion,       // There exists a query which include the current one
00033       Equivalence,     // There exists a query equivalent to the current one
00034       Combination      // There exists a set of queries that can be combined
00035                        // into one equivalent to the current one
00036     } MineruleRelationship;
00037
00038     class OptimizationInfo {
00039     public:
00040       MineruleRelationship relationship;
00041       ParsedMinerule minerule;
00042
00043       // used only in case relationship==Combination, it contains the set of queries
00044       // which need to be combined in order to form the equivalent one
00045       std::vector<ParsedMinerule> minerulesToCombine;
00046       // Specifies how the queries have to be combined
00047       GAQueryCombinator::QueryOrList combinationFormula;
00048     };
00049
00050   private:
00051     ParsedMinerule minerule;
00052     OptimizationInfo optInfo;
00053     bool sortSourceTableForGids;
00054
00055   protected:
00056     void
00057       dropTableIfExists(mrdb::Connection*, const std::string& tname) const;
00058
00059     // ??? with respect to what the relationship should be???
00060     MineruleRelationship getRelationshipType();
00061
00062     static void attributesInPredicate( const list_OR_node* l, std::set<std::string>& result );
00063     static bool predicateAttributesAreIncluded( const ParsedMinerule& mr1, const ParsedMinerule& mr2
      );
00064     void checkForCombinedQueries(  );
00065
00066   public:
00067     OptimizedMinerule(const std::string& minerule_text) {
00068       minerule.init(minerule_text);
00069       optInfo.relationship = None;
00070     }
00071
00072     void optimize();
00073
00074     const ParsedMinerule& getParsedMinerule() const {
00075       return minerule;
00076     }
00077
00078     const OptimizationInfo&
00079       getOptimizationInfo() const {
00080       return optInfo;
00081     }
00082
00083     OptimizationInfo&
00084       getOptimizationInfo() {
00085       return optInfo;
00086     }
00087
00088                 bool hasIDConstraints() const ;
00089
00090     static bool isACandidateQuery( const ParsedMinerule& candidate, const ParsedMinerule& target );
00091         static bool firstMineruleIncludesSecondMinerule(const ParsedMinerule&, const ParsedMinerule&);
00092         static bool firstMineruleDominatesSecondMinerule(const ParsedMinerule&, const
     ParsedMinerule&);
00093
00094     // Returns None || Dominance || Inclusion || Equivalence
00095     // the relation is to be intended as: "the first RELATION the second"
00096     // i.e. if Dominance is returned it means: "the first dominates the second".
00097     static MineruleRelationship getMineruleRelationship(const ParsedMinerule&, const ParsedMinerule&);
00098   };
00099
```

```
00100 } // namespace
00101 #endif
```

7.97 /Users/esposito/Software/minerule/include/minerule/Optimizer/OptimizerCatalogue.hpp File Reference

```
#include <set>
#include <map>
#include <string>
#include <string.h>
#include <stdlib.h>
#include "minerule/mrdb/Types.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Result/QueryResult-header.hpp"
#include "minerule/Optimizer/CatalogueInstaller.hpp"
```

Data Structures

- class minerule::CatalogueInfo
- class minerule::OptimizerCatalogue
- class minerule::OptimizerCatalogue::Catalogue
- class minerule::OptimizerCatalogue::EqKeysCatalogue
- class minerule::OptimizerCatalogue::DepFunCatalogue
- class minerule::OptimizerCatalogue::MineruleResultInfo

Namespaces

- namespace minerule

7.98 OptimizerCatalogue.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __OPTIMIZER_CATALOGUE_H__
00017 #define __OPTIMIZER_CATALOGUE_H__
00018
00019
00020 #include <set>
00021 #include <map>
00022 #include <string>
```

```
00023 #include <string.h>
00024 #include <stdlib.h>
00025 #include "minerule/mrdb/Types.hpp"
00026 #include "minerule/Utils/MineruleException.hpp"
00027 #include "minerule/Parsers/ParsedMinerule.hpp"
00028 #include "minerule/Database/ItemType.hpp"
00029 #include "minerule/Result/QueryResult-header.hpp"
00030 #include "minerule/Optimizer/CatalogueInstaller.hpp"
00031
00032
00033
00034
00048 namespace minerule {
00049
00050   class CatalogueInfo {
00051   public:
00052     std::string qryName;
00053     std::string qryText;
00054         std::string resName;
00055     std::vector<std::string> resTables; // name of the tables used by the result
00056     size_t resSize;
00057
00058     void updateQrySize() ;
00059   };
00060
00061
00062   class OptimizerCatalogue {
00063     /* ---------- Public types ---------- */
00064   public:
00065     typedef enum { EQKEYS, DEPFUN } CatalogueType;
00066     typedef enum { Equal, Reversed, None } OrderType;
00067
00068     // a KeyCols instance will store the name of columns that can
00069     // be grouped to form a key
00070     typedef std::set<std::string> KeyCols;
00071
00072     // A CatalogueEntry is a mapping originalAttrList->substAttrList
00073     // with the additional information about the type of ordering that
00074     // exists between riginalAttrList and substAttrList
00075     typedef std::multimap< KeyCols,std::pair<KeyCols,OrderType> > CatalogueEntry;
00076
00077     // A catalogue is a mapping tableName->CatalogueEntry
00078         class Catalogue : public std::map<std::string, CatalogueEntry> {
00079         public:
00080                 Catalogue() : std::map<std::string, CatalogueEntry>() {}
00081                 Catalogue(const Catalogue& catalogue) :
00082                 map<std::string, CatalogueEntry>(catalogue) {}
00083                 virtual ~Catalogue() {}
00084
00085                 void insertMapping(const std::string& table,const KeyCols& origKeyCols,int
      refKeyId,OrderType orderType) ;
00086                 static OrderType stringToOrder(const std::string&);
00087                 void initialize();
00088                 virtual const std::string& getSchemaInfo(const std::string& schemaKey) const=0;
00089         };
00090
00091     class EqKeysCatalogue : public Catalogue {
00092       std::map<std::string,std::string> schema;
00093     public:
00094       EqKeysCatalogue() : Catalogue() {
00095                   schema["tab_main"]="mr_eq_keys";
00096                   schema["tab_main_tab_name"]="tab_name";
00097                   schema["tab_main_lhs_key_id"]="key_id";
00098                   schema["tab_main_rhs_key_id"]="ref_key_id";
00099                   schema["tab_main_order_type"]="order_type";
00100                   schema["tab_cols"]="mr_eq_keys_col";
00101                   schema["tab_cols_col_name"]="col_name";
00102                   schema["tab_cols_key_id"]="key_id";
00103                   initialize();
00104           }
00105
00106       EqKeysCatalogue(const EqKeysCatalogue& catalogue) : Catalogue(catalogue),
      schema(catalogue.schema) { }
00107       virtual ~EqKeysCatalogue() {}
00108
00109       virtual const std::string& getSchemaInfo(const std::string& schemaKey) const {
00110                   std::map<std::string,std::string>::const_iterator it;
00111                   it=schema.find(schemaKey);
00112                   if(it==schema.end())
00113                           throw MineruleException( MR_ERROR_CATALOGUE_ERROR,
00114                                   "Requested for the unknown schema key:"+schemaKey);
00115                   return it->second;
00116       }
00117     };
00118
00119     class DepFunCatalogue : public Catalogue {
00120       std::map<std::string,std::string> schema;
```

```
00121     public:
00122       DepFunCatalogue() : Catalogue() {
00123                   schema["tab_main"]="mr_dep_fun";
00124                   schema["tab_main_tab_name"]="lhs_tab_name";
00125                   schema["tab_main_lhs_key_id"]="lhs_att_list_id";
00126                   schema["tab_main_rhs_key_id"]="rhs_att_list_id";
00127                   schema["tab_main_order_type"]="order_type";
00128                   schema["tab_cols"]="mr_dep_fun_col";
00129                   schema["tab_cols_col_name"]="col_name";
00130                   schema["tab_cols_key_id"]="col_id";
00131                   initialize();
00132       }
00133 
00134       DepFunCatalogue(const DepFunCatalogue& catalogue) :
00135         Catalogue(catalogue), schema(catalogue.schema) { }
00136 
00137       virtual ~DepFunCatalogue() {}
00138       virtual const std::string& getSchemaInfo(const std::string& schemaKey) const {
00139         std::map<std::string,std::string>::const_iterator it;
00140         it=schema.find(schemaKey);
00141         if(it==schema.end())
00142           throw MineruleException( MR_ERROR_CATALOGUE_ERROR,
00143                                    "Requested for the unknown schema key:"+schemaKey);
00144         return it->second;
00145       }
00146     };
00147 
00148     // Input of addMineruleResult. We use a class since it is likely
00149     // that the input of this function will vary as more optimization
00150     // politics are implemented. In this way the interface of the
00151     // function will stay the same while we add functionality
00152     // to the system.
00153     class MineruleResultInfo : public ParsedMinerule {
00154     public:
00155                 std::string resultset;
00156 
00157                 MineruleResultInfo(const ParsedMinerule& mr) : ParsedMinerule(mr),
      resultset(mr.tab_result) {};
00158                 MineruleResultInfo(const MineruleResultInfo& mri) : ParsedMinerule(mri),
      resultset(mri.resultset) {};
00159         };
00160 
00161  private:
00162     EqKeysCatalogue eqKeysCatalogue;
00163     DepFunCatalogue depFunCatalogue;
00164 
00165         static void setMRQueryInfoFromResultSet( mrdb::ResultSet* rs, CatalogueInfo& info, bool
      includeResultSize );
00166  public:
00167     /* ---------- Public Methods ---------- */
00168 
00169     //void addEquivalentKey( const KeyCols&, const KeyCols& );
00170     OptimizerCatalogue() {
00171       /*     eqKeysCatalogue.initialize();
00172              depFunCatalogue.initialize();*/
00173     }
00174 
00175     const Catalogue& getCatalogue(CatalogueType ct=EQKEYS) const {
00176                 switch( ct ) {
00177                         case EQKEYS:
00178                                 return eqKeysCatalogue;
00179                         case DEPFUN:
00180                                 return depFunCatalogue;
00181                         default:
00182                                 throw MineruleException(MR_ERROR_CATALOGUE_ERROR, "Requests for an
      unknown catalogue type!");
00183                 }
00184     }
00185 
00186     bool isIDAttribute(const std::string& tableName,
00187                        const std::vector<std::string>& itemCols,
00188                        const std::string& attribute) const ;
00189 
00190 
00197                 static bool hasIDConstraints(const ParsedMinerule& mineruel) ;
00198 
00199 
00200     // ==================== STATIC METHODS ====================
00201 
00205         static bool checkInstallation();
00206     static void install() ;
00207     static void uninstall() ;
00208 
00209 
00210     static void addMineruleResult( const MineruleResultInfo& mri ) ;
00211         static void addDerivedResult(const std::string& original, const std::string derived) ;
00212     static std::string addMineruleAttributeList(const ParsedMinerule::AttrVector& l) ;
```

```
00213     static std::string getNewAutoincrementValue(const std::string& tableName) ;
00214
00221     static std::string getResultsetName(const std::string& queryname) ;
00222
00228     static bool existsMinerule(const std::string& mrname);
00229
00230     static void deleteMinerule(const std::string& mrname)
00231       ;
00232
00236         static std::string getMRQueryName(size_t i) ;
00237
00242     static void getMRQueryNames(std::vector<std::string>& nameVec)
00243       ;
00244
00249     static void getMRQueryInfos(std::vector<CatalogueInfo>& catInfoVec, bool includeResultSize=true)
00250       ;
00251
00255     static void getMRQueryInfo(const std::string& qryName, CatalogueInfo& catInfo, bool
      includeResultSize=true) ;
00256
00269     static void getMRQueryResultIterator(const std::string& qryName,
00270                                          QueryResult::Iterator& it,
00271                                          double sup=-1,
00272                                          double con=-1)
00273       ;
00274   };
00275
00276
00277 } // namespace
00278
00279 #endif
```

7.99 /Users/esposito/Software/minerule/include/minerule/Optimizer/QueryNormalizer.hpp File Reference

```
#include <vector>
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Utils/SQLUtils.hpp"
```

Data Structures

- class minerule::QueryNormalizer
- class minerule::QueryNormalizer::SubstEntryBody
- class minerule::QueryNormalizer::SubstEntryHead
- class minerule::QueryNormalizer::SubstEntryHead::Elem

Namespaces

- namespace minerule

7.100 QueryNormalizer.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
```

```
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __QUERYNORMALIZER_H__
00017 #define __QUERYNORMALIZER_H__
00018
00019 #include <vector>
00020 #include "minerule/Parsers/ParsedMinerule.hpp"
00021 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00022 #include "minerule/Utils/SQLUtils.hpp"
00023
00024 namespace minerule {
00025
00026   /*
00027    * Allows to normalize ParsedMinerules so that they are more easily
00028    * checked for equivalence,dominance and inclusion relationships.
00029    * Look at the implementation file for more comments about how it
00030    * actually works.
00031    */
00032
00033   class QueryNormalizer {
00034   public:
00035     /* ---------- Public Types ---------- */
00036     /* ---------- (needed for query normalization) ---------- */
00037     class SubstEntryBody {
00038     public:
00039       // pred is the ptr to the 'next' field of the node preceding
00040       // the node from which this struct were built
00041       // (note that it can be used to unlink the current
00042       // node).
00043       list_AND_node** pred;
00044       size_t pos;
00045
00046       SubstEntryBody() {
00047         pred=NULL;
00048         pos=0;
00049       }
00050
00051       SubstEntryBody(const SubstEntryBody& seb) :
00052         pred( seb.pred ),
00053         pos( seb.pos ) { }
00054
00055       bool operator<(const SubstEntryBody& s2) const {
00056         return this->pos < s2.pos;
00057       }
00058     };
00059
00060     class SubstEntryHead {
00061     public:
00062
00063       class Elem {
00064       public:
00065         OptimizerCatalogue::OrderType  order;
00066
00067 std::string colName;        // the column name
00068 std::string value;          // the adjoint value
00069 std::string op;             // the operator involved
00070
00071         Elem() {
00072         }
00073
00074         Elem(const Elem& elem) :
00075           order(elem.order),
00076           colName(elem.colName),
00077           value(elem.value),
00078           op(elem.op)  {}
00079       }; // class Elem
00080
00081       OptimizerCatalogue::KeyCols        refKey;
00082       std::vector<Elem>   elems;
00083
00084       SubstEntryHead() {
00085       }
00086
00087       SubstEntryHead(const SubstEntryHead& seh) :
00088         refKey( seh.refKey ),
00089         elems( seh.elems ) {}
00090
00091       bool operator<(const SubstEntryHead& s2) const {
00092         return this->refKey<s2.refKey;
00093       }
00094     };
```

```
00095
00096     typedef std::set<SubstEntryHead> HeadInfo;
00097     typedef std::set<SubstEntryBody> BodyInfo;
00098   private:
00099       /* ---------- Member Fields ---------- */
00100     const OptimizerCatalogue& optimizerCatalogue;
00101     const OptimizerCatalogue::Catalogue& catalogue;
00102     ParsedMinerule* mr;
00103
00104
00105     /* ---------- Private Methods ---------- */
00106    std::string reverseOperator(std::string op, bool doReverse=true ) const;
00107
00108     void substituteInPredicate(
00109                 list_OR_node* cond,
00110                 const OptimizerCatalogue::CatalogueEntry& catEntry ) const;
00111
00112     void substituteInAndList(
00113                 list_AND_node*& andlist,
00114                 const OptimizerCatalogue::CatalogueEntry& catEntry,
00115                 const std::string& prefix) const;
00116
00117     void performSubstitutionInAndList(
00118                 list_AND_node*& andList,
00119                 const OptimizerCatalogue::KeyCols& Ai,
00120                 const OptimizerCatalogue::KeyCols& ai,
00121                 OptimizerCatalogue::OrderType order,
00122                 const std::string& prefix) const;
00123
00124     bool findSubstitutions( list_AND_node*& andList,
00125                             const OptimizerCatalogue::KeyCols& Ai,
00126                             const OptimizerCatalogue::KeyCols& ai,
00127                             const OptimizerCatalogue::OrderType order,
00128                             BodyInfo& bodies,
00129                             HeadInfo& heads,
00130                             const std::string& prefix) const;
00131
00132
00133     void setSimplePred(simple_pred* pred,
00134                        std::string colName,
00135                        std::string op,
00136                         SQLUtils::Type,
00137                         const std::string& ) const;
00138
00139     void removeAndNodes( BodyInfo& sInfo ) const;
00140     void addAndNode(list_AND_node*& andList,
00141                     simple_pred* pred) const;
00142
00143     void addNewConditions( list_AND_node*& andList,
00144                            const HeadInfo& heads,
00145                            const std::string& prefix) const;
00146
00147     void addNewCondition( list_AND_node*& andList,
00148                           const std::string& aiVal,
00149                           const std::vector<SubstEntryHead::Elem>& elems,
00150                           const std::string& prefix) const;
00151
00152     void substituteInAttrList(
00153                 ParsedMinerule::AttrVector& l,
00154                 const OptimizerCatalogue::CatalogueEntry& catEntry
00155                 ) const;
00156
00157     void compactPredicates( list_AND_node*& l ) const;
00158
00159     void relaxOperatorsInAndList(list_AND_node* andlist)
00160       ;
00161
00162     void relaxOperatorInPred(simple_pred* pred)
00163       ;
00164
00165    std::string getRelaxedValue(const std::string& tabSource,
00166                            const std::string& attr,
00167                            const std::string& op,
00168                            const std::string& value)
00169       ;
00170
00171     void relaxOperators(list_OR_node* cond)
00172       ;
00173
00174     void cleanPredicate(list_OR_node*& cur) const;
00175   public:
00176     /* ---------- Public Methods ---------- */
00177
00178     QueryNormalizer( const OptimizerCatalogue& oc,
00179                      ParsedMinerule& mrule ) : optimizerCatalogue(oc),
00180                                                catalogue( oc.getCatalogue() ),
00181                                                mr(&mrule) {
```

```
00182     };
00183
00184
00185
00186     void setMinerule( ParsedMinerule& mrule ) {
00187       mr = &mrule;
00188     }
00189
00190     // As a side effect it normalizes the minerule mr
00191     void normalize() ;
00192   };
00193
00194 } // namespace
00195
00196 #endif
```

7.101 /Users/esposito/Software/minerule/include/minerule/Parsers/EvaluationMeasure.hpp File Reference

Data Structures

- class minerule::EvaluationMeasure

Namespaces

- namespace minerule

7.102 EvaluationMeasure.hpp

Go to the documentation of this file.

```
00001 #ifndef __SUPPORTMEASURE_H__
00002 #define __SUPPORTMEASURE_H__
00003
00004
00005 namespace minerule {
00006
00007 class EvaluationMeasure {
00008   // private:
00009   //  const ItemDefinitionTable* table;
00010
00011
00012  public:
00013   typedef enum {
00014     MONOTONE,
00015     ANTIMONOTONE,
00016     OTHER
00017   } MeasureType;
00018
00019   typedef enum {
00020     LessEqual,
00021     Less,
00022     Greater,
00023     GreaterEqual
00024   } RelOperator;
00025
00026    // void setItemDefinitionTable( const ItemDefinitionTable& tab ) { table=&tab; } ;
00027
00028   virtual MeasureType getMeasureType()=0;
00029   //virtual bool eval(ItemSet, ItemSet, double, double, double)=0;
00030 };
00031
00032
00033 //esempio utilizzo
00034
00035 //if( support.getMeasureType() == EvaluationMeasure::MONOTONE )
00036
00037 }//end namespace
00038
00039 #endif
```

7.103 /Users/esposito/Software/minerule/include/minerule/Parsers/HeaderQuery.hpp File Reference

```
#include <string>
#include "minerule/Utils/MinMaxPair.hpp"
#include "ParsedMinerule.hpp"
```

Data Structures

- class minerule::HeaderQuery

Namespaces

- namespace minerule

7.104 HeaderQuery.hpp

Go to the documentation of this file.

```
00001 #ifndef __HEADERQUERY_H__
00002 #define __HEADERQUERY_H__
00003 #include <string>
00004 #include "minerule/Utils/MinMaxPair.hpp"
00005 #include "ParsedMinerule.hpp"
00006
00007 namespace minerule {
00008
00012   class HeaderQuery{
00013
00014 private:
00015     std::string name;
00016     MinMaxPair* cardB;
00017     MinMaxPair* cardH;
00018     ParsedMinerule::AttrVector* attrB;
00019     ParsedMinerule::AttrVector* attrH;
00020
00021  public:
00022
00026     HeaderQuery(std::string& na,
00027                 MinMaxPair* cB,
00028                 MinMaxPair* cH,
00029                 ParsedMinerule::AttrVector* aB,
00030                 ParsedMinerule::AttrVector* aH) :
00031                     name(na),
00032                     cardB(cB),
00033                     cardH(cH),
00034                     attrB(aB),
00035                     attrH (aH) {};
00036
00040     HeaderQuery(std::string& na,
00041                 MinMaxPair* cB,
00042                 ParsedMinerule::AttrVector* aB) :
00043       name(na),
00044       cardB(cB),
00045       cardH(NULL),
00046       attrB(aB),
00047       attrH (NULL) {};
00048
00049
00054     std::string getName(){
00055       return name;
00056     }
00057
00058     MinMaxPair* getCardBody(){
00059       return cardB;
00060     }
00061
00062     MinMaxPair* getCardHead(){
```

```
00063       return cardH;
00064     }
00065
00066     ParsedMinerule::AttrVector* getAttrBody(){
00067       return attrB;
00068     }
00069
00070     ParsedMinerule::AttrVector* getAttrHead(){
00071       return attrH;
00072     }
00073
00074   };
00075 }//end namespace
00076 #endif
```

7.105 /Users/esposito/Software/minerule/include/minerule/Parsers/↩ ParsedMinerule.hpp File Reference

```
#include <string>
#include <vector>
#include "minerule/Utils/MinMaxPair.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
#include <limits>
```

Data Structures

- struct __tagsimple_pred
- struct __taglist_AND_node
- struct __taglist_OR_node
- struct __tagminerule
- class Attribute
- class minerule::Bem_cond
- class minerule::Dist_cond
- class minerule::ParsedMinerule

Namespaces

- namespace minerule

Typedefs

- typedef struct __tagsimple_pred simple_pred
- typedef struct __taglist_AND_node list_AND_node
- typedef struct __taglist_OR_node list_OR_node
- typedef struct __tagminerule mineruletag

Functions

- list_OR_node * clone_l_OR (list_OR_node *l)
- list_AND_node * clone_l_AND (list_AND_node *l)
- std::ostream & minerule::operator<< (std::ostream &os, const ParsedMinerule &mr)

7.105.1 Typedef Documentation

7.105.1.1 list_AND_node

```
typedef struct __taglist_AND_node list_AND_node
```

Definition at line 19 of file ParsedMinerule.hpp.

7.105.1.2 list_OR_node

```
typedef struct __taglist_OR_node list_OR_node
```

Definition at line 20 of file ParsedMinerule.hpp.

7.105.1.3 mineruletag

```
typedef struct __tagminerule mineruletag
```

Definition at line 22 of file ParsedMinerule.hpp.

7.105.1.4 simple_pred

```
typedef struct __tagsimple_pred simple_pred
```

Definition at line 18 of file ParsedMinerule.hpp.

7.105.2 Function Documentation

7.105.2.1 clone_l_AND()

```
list_AND_node * clone_l_AND (
            list_AND_node * l )
```

Definition at line 84 of file ParsedMinerule.cpp.

```
00085 {
00086         list_AND_node* result;
00087         simple_pred*   p;
00088
00089         if (l==NULL) return NULL;
00090         result=(list_AND_node*)malloc(sizeof(list_AND_node));
00091
00092         if (result==NULL)
00093                 mrerror("Cannot allocate memory for a new list_AND_node");
00094
00095         p=(simple_pred*)malloc(sizeof(simple_pred));
00096
00097         if (p==NULL)
00098                 mrerror("simple_pred");
00099
00100         p->val1=strdup(l->sp->val1);
00101         p->op=strdup(l->sp->op);
00102         p->val2=strdup(l->sp->val2);
00103
00104         result->sp=p;
00105         result->next=clone_l_AND(l->next);
00106
00107         return result;
00108 }
```

7.105.2.2 clone_l_OR()

```
list_OR_node * clone_l_OR (
            list_OR_node * l )
```

Definition at line 111 of file ParsedMinerule.cpp.

```
00112 {
00113         list_OR_node* result;
00114
00115         if (l==NULL) return NULL;
00116
00117         result=(list_OR_node*)malloc(sizeof(list_OR_node));
00118         if (result==NULL)
00119                 mrerror("Cannot allocate memory for a new list_OR_node");
00120
00121         result->l_and=clone_l_AND(l->l_and);
00122         result->next=clone_l_OR(l->next);
00123         return result;
00124 }
```

7.106 ParsedMinerule.hpp

Go to the documentation of this file.

```
00001 // ParsedMinerule.h.h
00002 //#include"Ottimizzatore/ottimizzatore.h"
00003 //#include<stdio>
00004
00005 #ifndef _ParsedMinerule_
00006 #define _ParsedMinerule_
00007
00008 #include<string>
00009 #include<vector>
00010 #include"minerule/Utils/MinMaxPair.hpp"
00011 #include"minerule/Utils/MineruleLogs.hpp"
00012 #include"minerule/Utils/MineruleException.hpp"
00013 #include"minerule/Utils/AlgorithmTypes.hpp"
00014
00015 #include <limits>
```

```
00016
00017
00018 typedef struct __tagsimple_pred simple_pred;
00019 typedef struct __taglist_AND_node list_AND_node;
00020 typedef struct __taglist_OR_node list_OR_node;
00021
00022 typedef struct __tagminerule mineruletag;
00023
00024 struct __tagsimple_pred
00025 {
00026   char* val1;
00027   char* op;
00028   char*   val2;
00029 };
00030
00031 struct __taglist_AND_node
00032 {
00033   simple_pred*  sp;
00034   list_AND_node*  next;
00035 };
00036
00037 struct __taglist_OR_node
00038 {
00039   list_AND_node*  l_and;
00040   list_OR_node* next;
00041 };
00042
00043 struct __tagminerule
00044 {
00045   struct att_list*  ga;
00046   struct att_list*  ca;
00047   struct att_list*  ra;
00048   list_OR_node* mc;
00049   list_OR_node* gc;
00050   list_OR_node* cc;
00051 };
00052
00053 list_OR_node* clone_l_OR(list_OR_node* l);
00054 list_AND_node* clone_l_AND(list_AND_node* l);
00055
00056 class Attribute
00057 {
00058   public:
00059    std::string  name;
00060     Attribute(const char* s);
00061     bool operator<(const Attribute& a) const;
00062 };
00063
00064
00065 namespace minerule {
00066
00067 class Bem_cond{
00068 public:
00069     std::string type;
00070     std::string attr;
00071     std::string op;
00072     std::string val;
00073     int count_min;
00074     int count_max;
00075     Bem_cond* and_c;
00076
00077     Bem_cond(){
00078         type="";
00079         attr="";
00080         op="";
00081         val="";
00082         count_min=1;
00083         count_max=std::numeric_limits<int>::max();
00084         and_c=NULL;
00085     }
00086
00087     Bem_cond(const minerule::Bem_cond& copy_me){
00088         type=copy_me.type;
00089         attr=copy_me.attr;
00090         op=copy_me.op;
00091         val=copy_me.val;
00092         count_min=copy_me.count_min;
00093         count_max=copy_me.count_max;
00094         if(copy_me.and_c!=NULL)
00095             and_c= new Bem_cond(*(copy_me.and_c));
00096         else
00097             and_c=NULL;
00098     }
00099
00100     static std::vector<Bem_cond*> copyBemCond(std::vector<Bem_cond*> copy_me) {
00101         std::vector<Bem_cond*> out;
00102         for(int i=0; i<copy_me.size(); ++i)
```

```
00103             out.push_back(new Bem_cond(*copy_me[i]));
00104         return out;
00105     }
00106 };
00107
00108
00109 class Dist_cond {
00110 public:
00111     std::string function;
00112     std::vector<std::string> attr;
00113     std::string range;
00114
00115     Dist_cond(){
00116         function="";
00117         range="";
00118     }
00119
00120     Dist_cond(const Dist_cond& copy_me){
00121         function=copy_me.function;
00122         range=copy_me.range;
00123         for(int i=0; i<copy_me.attr.size(); ++i) {
00124             attr.push_back(copy_me.attr[i]);
00125         }
00126     }
00127
00128     static std::vector<Dist_cond*> copyDistCond(std::vector<Dist_cond*> copy_me) {
00129         std::vector<Dist_cond*> out;
00130         for(int i=0; i<copy_me.size(); ++i)
00131             out.push_back(new Dist_cond(*copy_me[i]));
00132         return out;
00133     }
00134 };
00135
00136 class ParsedMinerule
00137 {
00138  public:
00139   typedef std::vector<std::string> AttrVector;
00140   typedef Bem_cond* bem_c;
00141
00142  private:
00143   void fillAttrList(AttrVector& dest, struct att_list* source);
00144
00145  std::string getAttrText(const AttrVector& l) const;
00146  std::string getSimplePredText(const simple_pred* pred) const;
00147  std::string getAndListText(const list_AND_node* cond)const;
00148  std::string getCondText(const list_OR_node* cond)const;
00149  std::string getCardsText(const MinMaxPair& mm) const;
00150
00151  public:
00152
00153   AttrVector  ga;  // group attr list
00154   AttrVector  oa;  // ordering attr list (useful for mining sequences).
00155   AttrVector  ca;  // cluster attr list
00156   AttrVector  ra;  // rule attr list
00157
00158   AttrVector  ba;  // body attr list
00159   AttrVector  ha;  // head attr list
00160
00161   /* le seguenti liste dovrebbero essere sostituite
00162      da oggetti c++ (come quelle sopra per intenderci)*/
00163   list_OR_node* mc;  // mining condition
00164   list_OR_node* gc;  // group condition
00165   list_OR_node* cc;  // cluster condition
00166
00167   AttrVector  c_aggr_list; // cluster aggregate list
00168
00169   float     sup;
00170   float     conf;
00171   MinMaxPair bodyCardinalities;
00172   MinMaxPair headCardinalities;
00173  std::string    tab_source;
00174  std::string    tab_result;
00175
00176   bool tautologies;
00177   bool body_coincident_head;
00178   bool distinct;
00179
00180  /* Attributes used only by the sequence mining algorithm */
00181   MinMaxPair sequenceAllowedGaps;
00182   MinMaxPair length;
00183   std::vector<Dist_cond*> seq_dist_vect;
00184   std::vector<Bem_cond*> seq_bem_vect;
00185   std::string filter_condition;
00186   //std::string getFilterText(const list_OR_node* cond)const;
00187
00188   MiningTasks miningTask;
00189
```

```
00190
00191   ParsedMinerule() :
00192     mc(NULL),
00193     gc(NULL),
00194     cc(NULL),
00195     sup(0.0),
00196     conf(0.0),
00197     bodyCardinalities( MinMaxPair(1,1000)),
00198     headCardinalities( MinMaxPair(1,1000)),
00199     sequenceAllowedGaps( MinMaxPair(0,0)),
00200     length(MinMaxPair(1,std::numeric_limits<int>::max())) {
00201   };
00202
00203   explicit ParsedMinerule(const std::string& minerule_text ):
00204     mc(NULL),
00205     gc(NULL),
00206     cc(NULL),
00207     sup(0.0),
00208     conf(0.0),
00209     bodyCardinalities(1,1000),
00210     headCardinalities(1,1000),
00211     sequenceAllowedGaps( MinMaxPair(0,0)),
00212     length(MinMaxPair(1,std::numeric_limits<int>::max())) {
00213     init(minerule_text);
00214   }
00215
00216   ~ParsedMinerule() {
00217 #ifdef MRUSERWARNING
00218 #warning CAPIRE PERCHE" DA SEGMENTATION FAULT NEL CASO LE SEGUENTI FREE SIANO DECOMMENTATE!
00219           // possible explanation: not all char* pointers in simple predicates have
00220           // been dynamically allocated...
00221 #endif
00222     //free_l_OR(mc);
00223     //free_l_OR(gc);
00224     //free_l_OR(cc);
00225   }
00226
00227   ParsedMinerule( const ParsedMinerule& mr ) :
00228       oa(mr.oa),
00229       filter_condition(mr.filter_condition),
00230       distinct(mr.distinct),
00231     ga(mr.ga),
00232     ca(mr.ca),
00233     ra(mr.ra),
00234     ba(mr.ba),
00235     ha(mr.ha),
00236     c_aggr_list(mr.c_aggr_list),
00237     sup(mr.sup),
00238     conf(mr.conf),
00239     bodyCardinalities( mr.bodyCardinalities ),
00240     headCardinalities( mr.headCardinalities ),
00241     tab_source(mr.tab_source),
00242     tab_result(mr.tab_result),
00243     tautologies(mr.tautologies),
00244     body_coincident_head(mr.body_coincident_head),
00245     sequenceAllowedGaps(mr.sequenceAllowedGaps),
00246     miningTask(mr.miningTask),
00247     length(mr.length){
00248     mc=clone_l_OR(mr.mc);
00249     gc=clone_l_OR(mr.gc);
00250     cc=clone_l_OR(mr.cc);
00251     seq_bem_vect= Bem_cond::copyBemCond(mr.seq_bem_vect);
00252     seq_dist_vect= Dist_cond::copyDistCond(mr.seq_dist_vect);
00253   }
00254
00255   void init(const std::string& minerule_text);
00256
00257   bool requiresClusters() const {
00258     return !ca.empty();
00259     //  return !(body_coincident_head && ca.empty());
00260   }
00261
00262   bool hasCrossConditions(const list_OR_node* cond) const;
00263
00264   bool hasCrossConditions() const {
00265     return hasCrossConditions(mc) || hasCrossConditions(gc) || hasCrossConditions(cc);
00266   }
00267
00268   bool hasDisjuctionsInMC() const {
00269     return mc!=NULL && mc->next!=NULL;
00270   }
00271
00272   bool hasSameBodyHead() const {
00273           return ba == ha;
00274   }
00275
00276   std::string getMinesequenceText() const;
```

```
00277   std::string getMineruleText() const;
00278   std::string getMineitemsetsText() const;
00279   std::string getText() const;
00280   std::string getBEMText() const;
00281
00282 };
00283
00284   std::ostream& operator«(std::ostream& os, const ParsedMinerule& mr);
00285 } // namespace minerule
00286
00287 #endif
```

7.107 /Users/esposito/Software/minerule/include/minerule/Parsers/↩ ParsedPredicate.hpp File Reference

```
#include <iostream>
#include <list>
#include <memory>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <vector>
#include <set>
#include <typeinfo>
#include "PredicateBase.hpp"
#include "ParsedMinerule.hpp"
```

Data Structures

- class minerule::ParsedSimplePredicate
- class minerule::ParsedPredConjunction
- class minerule::ParsedPredicate

Namespaces

- namespace minerule

7.108 ParsedPredicate.hpp

Go to the documentation of this file.

```
00001 #ifndef __PREDICATEPARSER_H__
00002 #define __PREDICATEPARSER_H__
00003
00004 #include <iostream>
00005 #include <list>
00006 #include <memory>
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <vector>
00011 #include <set>
00012 #include <typeinfo>
00013
00014 #include "PredicateBase.hpp"
00015 #include "ParsedMinerule.hpp"
00016
00017 namespace minerule {
00018
```

```
00024   class ParsedSimplePredicate: public SimplePredicateBase {
00025     bool partOfBorH;
00026     bool aggr_f;
00027   public:
00032     ParsedSimplePredicate() : SimplePredicateBase(), partOfBorH(false), aggr_f(false) {}
00033
00041     ParsedSimplePredicate(const std::string& val1, const std::string& op, const std::string& val2,
      bool BorH, bool aggr):
00042       SimplePredicateBase(val1, op, val2), partOfBorH(BorH), aggr_f(aggr){}
00043
00049     ParsedSimplePredicate(const ParsedSimplePredicate& rhs) :
00050       SimplePredicateBase(rhs), partOfBorH(rhs.partOfBorH), aggr_f(rhs.aggr_f) {}
00051
00058     virtual SimplePredicateBase* cloneInstance() const {
00059       return new ParsedSimplePredicate(*this);
00060     }
00061
00068     virtual SimplePredicateBase* newInstance() const {
00069         return new ParsedSimplePredicate();
00070     }
00074     PredicateBase* newPredicate() const;
00075
00079     PredConjunctionBase* newPredConjunction() const;
00080
00081
00082
00087     bool isPartOfBorH() const {
00088       return partOfBorH;
00089     }
00090
00095     bool isAggr_function() const {
00096       return aggr_f;
00097     }
00098   };
00099
00100   //-------------------------------------------------------------------------
00101   // ParsedPredConjunction
00102   //-------------------------------------------------------------------------
00103
00108   class ParsedPredConjunction : public PredConjunctionBase{
00109
00113   public:
00114
00118     ParsedPredConjunction(): PredConjunctionBase() {};
00119
00124     ParsedPredConjunction(const ParsedPredConjunction& rhs) {
00125       ParsedPredConjunction::const_iterator it;
00126       for (it=rhs.begin();it!=rhs.end();it++) {
00127         this->push_back( new ParsedSimplePredicate(dynamic_cast<const ParsedSimplePredicate&>(**it))
      );
00128       }
00129     }
00130
00133     virtual PredConjunctionBase* cloneInstance() const {
00134       return new ParsedPredConjunction( *this );
00135     }
00138     virtual PredConjunctionBase* newInstance() const {
00139       return new ParsedPredConjunction( );
00140     }
00141
00145     PredicateBase* newPredicate()const ;
00146
00150     PredConjunctionBase* newPredConjunction()const;
00151
00156     bool areAllBorH() const {
00157       ParsedPredConjunction::const_iterator it;
00158       for (it=this->begin();
00159            it!=this->end() && dynamic_cast<const ParsedSimplePredicate&>(**it).isPartOfBorH();
00160            it++); /* note the empty body of the for loop */
00161
00162       return it==this->end();
00163     }
00164
00169     bool areAllAggr_f() const {
00170       ParsedPredConjunction::const_iterator it;
00171       for (it=this->begin();
00172            it!=this->end() && dynamic_cast<const ParsedSimplePredicate&>(**it).isAggr_function();
00173            it++); /* note the empty body of the for loop */
00174       return it==this->end();
00175     }
00176
00182     bool atLeastOneBorH()const {
00183       ParsedPredConjunction::const_iterator it;
00184       for (it=this->begin();
00185            it!=this->end() && !dynamic_cast<const ParsedSimplePredicate&>(**it).isPartOfBorH();
00186            it++); /* note the empty body of the for loop */
00187
```

```
00188       return it!=this->end();
00189     }
00190
00191
00192   };
00193
00194   //-------------------------------------------------------------------
00195   // ParsedPredicate
00196   //-------------------------------------------------------------------
00197
00201   class ParsedPredicate : public PredicateBase {
00202   public:
00206     ParsedPredicate() : PredicateBase() {}
00212     ParsedPredicate(const ParsedPredicate& rhs) : PredicateBase(rhs) {}
00213
00216     virtual ParsedPredicate* cloneInstance() const {
00217       return new ParsedPredicate(*this);
00218     }
00219
00222     virtual ParsedPredicate* newInstance() const {
00223       return new ParsedPredicate();
00224     }
00225
00229     PredicateBase* newPredicate()const;
00230
00234     PredConjunctionBase* newPredConjunction() const;
00235
00240     bool areAllBorH() const {
00241       ParsedPredicate::const_iterator it;
00242
00243       for (it=this->begin();
00244            it!=this->end() && dynamic_cast<const ParsedPredConjunction&>(**it).areAllBorH();
00245            it++); /* note the empty body of the for loop */
00246
00247       return it==this->end();
00248     }
00249
00250
00255     bool areAllAggr_f() const {
00256       ParsedPredicate::const_iterator it;
00257       for (it=this->begin();
00258            it!=this->end() && dynamic_cast<const ParsedPredConjunction&>(**it).areAllAggr_f();
00259            it++); /* note the empty body of the for loop */
00260       return it==this->end();
00261     }
00262
00263
00268     bool atLeastOneBorH() const {
00269       ParsedPredicate::const_iterator it;
00270
00271       for (it=this->begin();
00272            it!=this->end() && !dynamic_cast<const ParsedPredConjunction&>(**it).atLeastOneBorH();
00273            it++); /* note the empty body of the for loop */
00274
00275       return it!=this->end();
00276     }
00277
00278
00283     list_OR_node* convert() const;
00284     list_AND_node* convert_and_list(const PredConjunctionBase&) const;
00285
00286   };
00287
00288 } // namespace
00289
00290 #endif
```

7.109 /Users/esposito/Software/minerule/include/minerule/Parsers/↩ ParserLibrary.hpp File Reference

```
#include "ParsedMinerule.hpp"
```

Namespaces

- namespace minerule

Functions

- void minerule::parseMinerule (std::string minerule_text, ParsedMinerule &output)
- ParsedMinerule & minerule::getParserOutputObj ()

7.110 ParserLibrary.hpp

Go to the documentation of this file.

```
00001 #ifndef __PARSER_LIBRARY_H__
00002 #define __PARSER_LIBRARY_H__
00003
00004 #include "ParsedMinerule.hpp"
00005
00006 namespace minerule {
00007
00008 void parseMinerule(std::string minerule_text, ParsedMinerule& output);
00009 ParsedMinerule& getParserOutputObj();
00010
00011 }
00012
00013 #endif
```

7.111 /Users/esposito/Software/minerule/include/minerule/Parsers/↩ PredicateBase.hpp File Reference

```
#include <iterator>
#include <vector>
#include <iostream>
#include <cassert>
```

Data Structures

- class minerule::SimplePredicateBase
- class minerule::PredConjunctionBase
- class minerule::PredicateBase

Namespaces

- namespace minerule

7.112 PredicateBase.hpp

Go to the documentation of this file.

```
00001 #ifndef __PREDICATEBASE_H__
00002 #define __PREDICATEBASE_H__
00003
00004
00005 #include <iterator>
00006 #include <vector>
00007 #include <iostream>
00008 #include <cassert>
00014 namespace minerule {
00015
00016   class PredicateBase;
```

```
00017   class PredConjunctionBase;
00018
00019   // ------------------------------------------------------------------------
00020   // SimplePredicateBase
00021   // ------------------------------------------------------------------------
00022
00028   class SimplePredicateBase  {
00029     std::string val1;
00030     std::string val2;
00031     std::string op;
00032   public:
00033
00040     SimplePredicateBase( const std::string& v1,
00041                          const std::string& o,
00042                          const std::string& v2 ) :
00043       val1(v1), val2(v2),  op(o) {  }
00044
00048     SimplePredicateBase() {}
00049
00054     SimplePredicateBase( const SimplePredicateBase& s) :
00055       val1(s.val1), val2(s.val2), op(s.op) {}
00056
00057
00060     virtual SimplePredicateBase* cloneInstance() const {
00061       return new SimplePredicateBase(*this);
00062     }
00065     virtual SimplePredicateBase* newInstance() const {
00066       return new SimplePredicateBase();
00067     }
00068
00072     virtual ~SimplePredicateBase() {}
00073
00077     virtual PredicateBase* newPredicate() const;
00078
00082     virtual PredConjunctionBase* newPredConjunction() const;
00087     const std::string& getVal1() const {
00088       return val1;
00089     }
00090
00095     const std::string& getVal2() const {
00096       return val2;
00097     }
00098
00103     const std::string& getOp() const {
00104       return op;
00105     }
00106
00111     virtual bool operator==( const SimplePredicateBase& rhs ) const {
00112       return
00113         val1==rhs.val1 &&
00114         val2==rhs.val2 &&
00115         op==rhs.op;
00116     }
00117
00122     virtual bool operator!=( const SimplePredicateBase& rhs ) const {
00123       return
00124         val1!=rhs.val1 ||
00125         val2!=rhs.val2 ||
00126         op!=rhs.op;
00127     }
00128
00133     virtual bool operator<(const SimplePredicateBase& rhs) const {
00134       return val1+op+val2 < rhs.val1+rhs.op+rhs.val2;
00135     }
00136
00141     virtual PredicateBase& operator!() const;
00142   };
00143
00144
00145   // ------------------------------------------------------------------------
00146   // PredConjunctionBase
00147   // ------------------------------------------------------------------------
00151   class PredConjunctionBase : public std::vector<SimplePredicateBase*> {
00152   public:
00156     PredConjunctionBase() {};
00157
00161     PredConjunctionBase( const PredConjunctionBase& );
00162
00166     virtual ~PredConjunctionBase() ;
00167
00171     virtual PredicateBase* newPredicate()const ;
00172
00176     virtual PredConjunctionBase* newPredConjunction()const;
00177
00178
00181     virtual PredConjunctionBase* cloneInstance() const {
00182       return new PredConjunctionBase(*this);
```

```
00183     }
00184
00186     virtual PredConjunctionBase* newInstance() const {
00187       return new PredConjunctionBase();
00188     }
00189
00190
00194     PredConjunctionBase& operator&=(const PredConjunctionBase& p);
00195
00200     PredicateBase& operator!() const;
00201
00205     bool find(SimplePredicateBase* sp) const;
00206
00213     virtual PredConjunctionBase* copy_and_append(SimplePredicateBase* sp){
00214       PredConjunctionBase* conj =  this->cloneInstance();
00215       conj->push_back( sp->cloneInstance() );
00216       return conj;
00217     }
00218   };
00219
00220   // ------------------------------------------------------------------------
00221   // Predicate. Notice: this is a vector of Predicate Conjunctions with
00222   //       another name and some handy functions.
00223   // ------------------------------------------------------------------------
00228   class PredicateBase : public std::vector<PredConjunctionBase*> {
00229   public:
00233     PredicateBase() {};
00234
00238     PredicateBase(const PredicateBase&);
00239
00242     virtual PredicateBase* cloneInstance() const {
00243       return new PredicateBase(*this);
00244     }
00245
00248     virtual PredicateBase* newInstance() const {
00249       return new PredicateBase();
00250     }
00251
00255     virtual ~PredicateBase();
00256
00260     virtual PredicateBase* newPredicate()const;
00261
00265     virtual PredConjunctionBase* newPredConjunction() const;
00266
00270     virtual PredicateBase& operator=(const PredicateBase& rhs);
00271
00275     PredicateBase& operator&=( const PredicateBase& p  );
00276
00280     PredicateBase& operator|=(const PredicateBase& p );
00281
00282
00286     void merge(PredicateBase* l2) {
00287       copy( l2->begin(), l2->end(), std::back_insert_iterator<PredicateBase>(*this));
00288       erase(l2->begin(), l2->end());
00289
00290       /*    Predicate::iterator it_and;
00291             it_and=l2->begin();
00292             while (it_and!=l2->end()){
00293             this->push_back(*it_and);
00294             *it_and=NULL;
00295             it_and++;
00296             }*/
00297     }
00298
00300     PredicateBase&  operator!() const;
00301
00305     void stamp(){
00306       if (this!=NULL){
00307         int c_and=0;
00308         int c_or=0;
00309         PredicateBase::iterator it_and = this->begin();
00310         PredConjunctionBase::iterator it_sp;
00311         while (it_and!=this->end()){
00312           if (c_or) std::cout<<"\n OR \n";
00313           it_sp=(*it_and)->begin();
00314           c_and=0;
00315           while(it_sp!=(*it_and)->end()){
00316             if (c_and) std::cout<<" AND ";
00317             std::cout<<(*it_sp)->getVal1()<<" ";
00318             std::cout<<(*it_sp)->getOp()<<" ";
00319             std::cout<<(*it_sp)->getVal2();
00320             it_sp++;
00321             c_and++;
00322           }
00323           //delete *it_and;
00324           it_and++;
00325           c_or++;
```

```
00326         }
00327         std::cout«"\n\n\n\n";
00328       }
00329
00330     }
00331
00332   };
00333
00334
00335 } // namespace
00336
00337 #endif
```

7.113 /Users/esposito/Software/minerule/include/minerule/Parsers/SupportMeasure.hpp File Reference

```
#include "EvaluationMeasure.hpp"
```

Data Structures

- class minerule::SupportMeasure

Namespaces

- namespace minerule

7.114 SupportMeasure.hpp

Go to the documentation of this file.

```
00001 #ifndef _SUPPORTMEASURE__H_
00002 #define _SUPPORTMEASURE__H_
00003
00004 #include "EvaluationMeasure.hpp"
00005
00006
00007 namespace minerule{
00008 /*
00009  * this class is used to encapsule the support
00010  **/
00011   class SupportMeasure:  public EvaluationMeasure {
00012
00013  private:
00014   EvaluationMeasure::RelOperator relOp;
00015   double threshold;
00016
00017  public:
00018
00019   SupportMeasure(EvaluationMeasure::RelOperator rel, double th): relOp(rel), threshold(th) {};
00020
00021   //virtual bool eval(ItemSet, ItemSet, double, double, double ) { assert(false); }
00022
00023   /*
00024    * this method return if a frequency of an itemset satisfy the support or not satisfy this measure
00025    **/
00026   bool evalSupport(double freq);
00027
00028
00029   /*
00030    * this method return the type of this comparison
00031    * check the relational opertor to establish if a measure is monotone or antimonotone
00032    **/
00033   EvaluationMeasure::MeasureType getMeasureType();
00034
00035
```

```
00036   /*
00037    * This method return the threshold of this support
00038    **/
00039   double getThreshold(){
00040     return threshold;
00041   }
00042
00043 };
00044
00045
00046 }//end namespace
00047 //esempio utilizzo
00048
00049 //if( support.getMeasureType() == EvaluationMeasure::MONOTONE )
00050
00051 #endif
```

7.115 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/ExpressionNFCoder.hpp File Reference

```
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/PredicateUtils/Predicate.hpp"
#include "minerule/PredicateUtils/InvalidConfigurationFilter.hpp"
```

Data Structures

- class minerule::EncodedNF
- class minerule::EncodedNFIterator
- class minerule::ExpressionNFCoder

Namespaces

- namespace minerule

Typedefs

- typedef unsigned int minerule::EncodingBaseType

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const EncodedNF &nf)

7.116 ExpressionNFCoder.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __EXPRESSIONNFCODER_H__
00017 #define __EXPRESSIONNFCODER_H__
00018
00019 #include "minerule/Utils/MineruleException.hpp"
00020 #include "minerule/PredicateUtils/Predicate.hpp"
00021 #include "minerule/PredicateUtils/InvalidConfigurationFilter.hpp"
00022
00023 // This class provide facilities to encode/decode the normal form
00024 // of an Expression contained in an Expression Evaluator
00025
00026 namespace minerule {
00027
00028   typedef unsigned int EncodingBaseType;
00029
00030   // ------------------------------------------------------------------------
00031   // EncodedNF
00032   // ------------------------------------------------------------------------
00033
00034   class EncodedNF {
00035   public:
00036      typedef enum {
00037       FirstMoreGeneral,
00038       FirstMoreSpecific,
00039       Equivalent,
00040       Unrelated
00041     } CodesRelationship;
00042
00043     const EncodingBaseType* encVector;
00044     size_t numVars;     // num of variables present in the expression that
00045     // originated this EncodedNF
00046     size_t vecSize;     // the size of encVector
00047     size_t codingLen;  // the exact number of elements that were encoded
00048
00049     bool operator==(const EncodedNF&) const;
00050
00051     static CodesRelationship
00052       getCodesRelationship( const EncodedNF&, const EncodedNF& );
00053
00054
00055     static size_t
00056       computeHammingDistance(const EncodedNF& nf1,
00057                              const EncodedNF& nf2);
00058   };
00059
00060   std::ostream& operator<<(std::ostream& os, const EncodedNF&);
00061
00062   // ------------------------------------------------------------------------
00063   // EncodedNFIterator
00064   // ------------------------------------------------------------------------
00065
00066   class EncodedNFIterator {
00067     const EncodedNF& target;
00068     VarSet currentVarSet;
00069     size_t cellCounter;
00070     size_t elemCounter;
00071     bool itok;
00072   public:
00073     EncodedNFIterator(const EncodedNF& er):
00074       target(er), currentVarSet(er.numVars), cellCounter(0), elemCounter(0), itok(false) {
00075
00076     }
00077
00078     bool ok() const {
00079       return itok;
00080     }
00081
00082     void setOk(bool val) {
```

```
00083       itok=val;
00084     }
00085
00086     bool operator++(int);
00087
00088     const VarSet& operator*() const {
00089       return currentVarSet;
00090     }
00091   };
00092
00093   // ----------------------------------------------------------------------
00094   // ExpressionNFCoder
00095   // ----------------------------------------------------------------------
00096
00097   class ExpressionNFCoder {
00098   protected:
00099     static const size_t bitsPerCell;
00100     InvalidConfigurationFilter& filter;
00101
00102   private:
00103     EncodingBaseType* buf;
00104
00105     void
00106       cleanBuf() {
00107       if(buf!=NULL)
00108         free(buf);
00109       buf=NULL;
00110     }
00111   public:
00112     ExpressionNFCoder(InvalidConfigurationFilter& f):  filter(f), buf(NULL) {
00113
00114     }
00115
00116     ~ExpressionNFCoder() {
00117       cleanBuf();
00118     }
00119
00120     // Return the encoded representation for the given expression normal
00121     // form. The returned pointer will be deleted as soon a new encoding
00122     // is requested or upon object deallocation.
00123     EncodedNF
00124       encode(Predicate&) ;
00125   };
00126
00127 } // namespace minerule
00128
00129 #endif
```

7.117 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp File Reference

```
#include <string>
#include "minerule/Parsers/ParsedMinerule.hpp"
```

Data Structures

- class minerule::HeadBodyPredicatesSeparator

Namespaces

- namespace minerule

7.118 HeadBodyPredicatesSeparator.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __HEADBODYPREDICATESEPARATOR_H__
00017 #define __HEADBODYPREDICATESEPARATOR_H__
00018
00019 #include <string>
00020 #include "minerule/Parsers/ParsedMinerule.hpp"
00021
00022 namespace minerule {
00023
00024   // ------------------------------------------------------------------------
00025   // HeadBodyPredicatesSeparator
00026   // ------------------------------------------------------------------------
00027
00037   class HeadBodyPredicatesSeparator {
00038   private:
00039     typedef enum { NO_INFO, BODY_CONSTR, HEAD_CONSTR, ERROR} ConstraintClass;
00040     static void
00041       setState( ConstraintClass& current, ConstraintClass newone )
00042       ;
00043
00044     static ConstraintClass
00045       setStateAccordinglyToString(std::string& str, ConstraintClass state)
00046       ;
00047
00048     static void
00049       updateConstraint( std::string& str,
00050                         const std::string& v1,
00051                         const std::string& op,
00052                         const std::string& v2);
00053   public:
00054     static void
00055       separate(list_AND_node* l_and,
00056                std::string& bodyConstraints,
00057                std::string& headConstraints) ;
00058   };
00059
00060 }
00061
00062 #endif
```

7.119 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Interval.hpp File Reference

```
#include <string.h>
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/mrdb/Types.hpp"
#include <set>
#include <map>
#include <string>
#include "minerule/PredicateUtils/VarSet.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Utils/SQLUtils.hpp"
```

Data Structures

- class minerule::StringCompare
- class minerule::Interval
- class minerule::IntervalChecker

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const Interval &i)

7.120 Interval.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __INTERVAL_CHECKER__
00017 #define __INTERVAL_CHECKER__
00018
00019 #include <string.h>
00020 #include "minerule/Utils/MineruleException.hpp"
00021 #include "minerule/mrdb/Types.hpp"
00022 #include <set>
00023 #include <map>
00024 #include <string>
00025 #include "minerule/PredicateUtils/VarSet.hpp"
00026 #include "minerule/Parsers/ParsedMinerule.hpp"
00027 #include "minerule/Utils/SQLUtils.hpp"
00028
00029
00030 namespace minerule {
00031
00032   /* ========================================================================
00033    * StringCompare
00034    * ======================================================================== */
00035   class StringCompare {
00036   public:
00037     bool operator()(const char* lhs, const char* rhs) const {
00038       return strcmp(lhs,rhs)<0;
00039     }
00040   };
00041
00042
00043   /* ========================================================================
00044    * Interval
00045    * ========================================================================
00046    */
00047
00048   class Interval {
00049     friend std::ostream& operator<<(std::ostream&, const Interval&);
00050   public:
00054     typedef enum {
00055       Less,
00056       LessEq,
00057       Grt,
```

```
00058       GrtEq,
00059       Eq,
00060       NotEq
00061     } Operator;
00062
00063   protected:
00069     static const char* POSINFTY;
00075     static const char* NEGINFTY;
00076
00081     const char* lwr; // lower limit
00086     const char* upp; // upper limit
00091     bool lwrOpen;
00092
00097     bool uppOpen;
00098
00104     SQLUtils::Type type;
00109     bool typeOk;
00110
00115     std::set<const char*, StringCompare> openPoints;
00116
00117
00130     Operator getOperator( const char* op, bool negateIt ) const ;
00131
00136     int compareValues( const char* val1, const char* val2) const ;
00137
00148     const char* getMaxLwr( Interval& lhs, Interval& rhs, bool& maxIsOpen ) const ;
00149
00160     const char* getMinUpp( Interval& lhs, Interval& rhs, bool& minIsOpen ) const  ;
00161
00162
00163   public:
00164     Interval() : lwr(NEGINFTY), upp(POSINFTY), lwrOpen(true), uppOpen(true), typeOk(false) { }
00165     Interval(const Interval& rhs) :
00166       lwr(rhs.lwr),
00167       upp(rhs.upp),
00168       lwrOpen(rhs.lwrOpen),
00169       uppOpen(rhs.uppOpen),
00170       type(rhs.type),
00171       typeOk(rhs.typeOk) {}
00172     Interval( Operator op, const char* val, SQLUtils::Type t );
00173
00174
00179     void intersect(Interval& i) {
00180       lwr = getMaxLwr(*this,i,lwrOpen);
00181       upp = getMinUpp(*this,i,uppOpen);
00182       openPoints.insert( i.openPoints.begin(), i.openPoints.end() );
00183     }
00184
00185
00189     void update(const list_AND_node* l, bool negateIt) ;
00190
00191
00195     void setType( SQLUtils::Type t ) {
00196       type=t;
00197       typeOk = true;
00198     }
00199
00200
00204     bool isEmpty() {
00205       int order = compareValues(lwr,upp);
00206       if( order < 0 )  return false;
00207       if( order == 0 ) return lwrOpen || uppOpen || (openPoints.find(lwr)!=openPoints.end());
00208       return true;
00209     }
00210
00211
00221     static const char* getValue( const char* val1, const char* val2 );
00222
00223
00233     static const char* getAttribute(const char* val1, const char* val2 );
00234   };
00235
00236   std::ostream& operator«(std::ostream&, const Interval&);
00237
00238   /* ============================================================================
00239    * IntervalChecker
00240    * ============================================================================
00241    */
00242
00243
00244   class IntervalChecker {
00245     std::map<const char*, SQLUtils::Type> typesMap;
00246     const std::string& tab_source;
00247
00248   public:
00249     IntervalChecker( const std::string& t ) : tab_source(t) {}
00250
```

```
00260     SQLUtils::Type typeForAttribute(const char* attr) ;
00261
00270     bool impossibleVariableSetting( const VarSet& vset, const list_AND_node* l);
00271   };
00272 }
00273
00274 #endif
```

7.121 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/InvalidConfigurationFilter.hpp File Reference

```
#include <string>
#include "minerule/PredicateUtils/VarSet.hpp"
#include "minerule/PredicateUtils/Interval.hpp"
```

Data Structures

- class minerule::InvalidConfigurationFilter

Namespaces

- namespace minerule

7.122 InvalidConfigurationFilter.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __INVALID_CONFIGURATION_FILTER_H__
00017 #define __INVALID_CONFIGURATION_FILTER_H__
00018
00019 #include <string>
00020
00021 #include "minerule/PredicateUtils/VarSet.hpp"
00022 #include "minerule/PredicateUtils/Interval.hpp"
00023
00024 namespace minerule {
00025
00026   class InvalidConfigurationFilter {
00027     const std::string& tab_source;
00028     list_AND_node* preds;
00029     IntervalChecker ic;
00030   public:
00031     InvalidConfigurationFilter( const std::string& t,
00032                                 list_AND_node* l ) :
00033       tab_source(t),
00034       preds(l),
00035       ic(t) {}
00036
00040     bool operator()(const VarSet& vset) {
00041       return ic.impossibleVariableSetting( vset, preds );
00042     }
00043   };
00044 };
00045
00046
00047
00048 #endif
```

7.123 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/Predicate.hpp File Reference

```
#include <vector>
#include <set>
#include <string>
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/PredicateUtils/VarSet.hpp"
```

Data Structures

- class minerule::SimplePredicate
- class minerule::PtrSimplePredComp
- class minerule::PredConjunction
- class minerule::Predicate

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const SimplePredicate &s)
- std::ostream & minerule::operator<< (std::ostream &os, const PredConjunction &p)
- std::ostream & minerule::operator<< (std::ostream &os, const Predicate &p)

7.124 Predicate.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __PREDICATE_H__
00017 #define __PREDICATE_H__
00018
00019 #include <vector>
00020 #include <set>
00021 #include <string>
00022
00023 #include "minerule/Parsers/ParsedMinerule.hpp"
00024 #include "minerule/PredicateUtils/VarSet.hpp"
00025
00026 namespace minerule {
00027   // forward declaration needed by SimplePredicate
00028   class PtrSimplePredComp;
00029
00030   // ----------------------------------------------------------------------
```

```
00031   // SimplePredicate
00032   // ----------------------------------------------------------------------
00033
00034   class SimplePredicate {
00035     typedef std::set<SimplePredicate*, PtrSimplePredComp> PredicatePool;
00036
00037     static  PredicatePool predicatePool;
00038
00039
00040     SimplePredicate(const simple_pred* pred ) :
00041       val1(pred->val1),
00042       val2(pred->val2),
00043       op(pred->op),
00044       varId(0) {
00045     }
00046
00047     SimplePredicate( const std::string& v1,
00048                      const std::string& o,
00049                      const std::string& v2 ) :
00050       val1(v1),
00051       val2(v2),
00052       op(o) {
00053     }
00054
00055     std::string val1;
00056     std::string val2;
00057     std::string op;
00058
00059     size_t varId;
00060   public:
00061
00062
00063     // We need to maintain a unique copy of identical predicates.
00064     // All new predicates will be inserted in the predicate pool,
00065     // when we try to construct a new predicate which is identical
00066     // to one olready known, we simply return the known version
00067
00068     static SimplePredicate&
00069       newSimplePredicate( const simple_pred* pred );
00070
00071     static SimplePredicate&
00072       newSimplePredicate(const std::string& val1,
00073                          const std::string& op,
00074                          const std::string& val2);
00075
00076     // It can be called to reclaim the memory used by the pool
00077     // when it is no longer used.
00078     static void
00079       freeSimplePredicatePool();
00080
00081     const std::string& getVal1() const {
00082       return val1;
00083     }
00084
00085     const std::string& getVal2() const {
00086       return val2;
00087     }
00088
00089     const std::string& getOp() const {
00090       return op;
00091     }
00092
00093     bool operator==( const SimplePredicate& rhs ) const {
00094       return
00095         val1==rhs.val1 &&
00096         val2==rhs.val2 &&
00097         op==rhs.op;
00098     }
00099
00100    bool operator<(const SimplePredicate& rhs) const {
00101      return val1+op+val2 < rhs.val1+rhs.op+rhs.val2;
00102    }
00103
00104    void setVarId( size_t id ) {
00105      varId = id;
00106    }
00107
00108    size_t getVarId() const {
00109      return varId;
00110    }
00111  };
00112
00113  // ----------------------------------------------------------------------
00114  // PtrSimplePredComp
00115  // ----------------------------------------------------------------------
00116
00126  class PtrSimplePredComp {
```

```
00127   public:
00128     bool operator()(const SimplePredicate* lhs,
00129                     const SimplePredicate* rhs) const {
00130       return *lhs < *rhs;
00131     }
00132   };
00133
00134   // ----------------------------------------------------------------------
00135   // PredConjunction
00136   // ----------------------------------------------------------------------
00137
00138   class PredConjunction : public std::vector<SimplePredicate*> {
00139   public:
00140     PredConjunction() {};
00141     PredConjunction( const list_AND_node*);
00142     PredConjunction( const PredConjunction& );
00143     ~PredConjunction() { }
00144
00145     bool evaluate(const VarSet&) const;
00146
00150     PredConjunction& operator&=(const PredConjunction& p);
00151   };
00152
00153   // ----------------------------------------------------------------------
00154   // Predicate. Notice: this is a vector of Predicate Conjunctions with
00155   //       another name and some handy functions.
00156   // ----------------------------------------------------------------------
00157
00158   class Predicate : public std::vector<PredConjunction*> {
00159   private:
00160     std::set<SimplePredicate*, PtrSimplePredComp>* predList;
00161     size_t numVariables;
00162
00163   public:
00164     Predicate() :
00165       predList(NULL),
00166       numVariables(0)  {};
00167
00168     Predicate(const Predicate&);
00169
00170     Predicate( const list_OR_node* node );
00171     ~Predicate();
00172
00173     Predicate& operator=(const Predicate& rhs) {
00174       //freeing used memory
00175       for( iterator it=begin(); it!=end(); it++ ) {
00176         delete *it;
00177       }
00178
00179       // initializing variables
00180       predList=NULL;
00181       numVariables=0;
00182       // copying rhs contents
00183       for(Predicate::const_iterator it=rhs.begin(); it!=rhs.end(); it++) {
00184         push_back( new PredConjunction(**it) );
00185       }
00186
00187       return *this;
00188     }
00189
00190
00194     Predicate& operator&=( const Predicate& p  );
00195
00199     Predicate& operator|=(const Predicate& p );
00200
00201     // Get the std::set of all simple predicates contained in
00202     // in this predicate. The user should call this function
00203     // passing true for parameter rebuildSet if the object
00204     // changed since the last time this function has been
00205     // called (otherwise, a cached, probably wrong, result
00206     // will be returned).
00207
00208     std::set<SimplePredicate*,PtrSimplePredComp>&
00209       getPredicateList(bool rebuildSet=false) ;
00210
00211     bool evaluate(const VarSet&) const;
00212
00213     void  setNumVariables(size_t n) {
00214       numVariables=n;
00215     }
00216     size_t getNumVariables() const {
00217       return numVariables;
00218     }
00219   };
00220
00221   inline std::ostream& operator«(std::ostream& os, const SimplePredicate& s) {
00222     os « s.getVal1() « s.getOp() « s.getVal2();
```

```
00223     return os;
00224   }
00225
00226   inline std::ostream& operator«(std::ostream& os, const PredConjunction& p) {
00227     PredConjunction::const_iterator it2;
00228     for(it2=p.begin(); it2!=p.end(); it2++) {
00229       if(it2!=p.begin()) {
00230         os « " AND ";
00231       }
00232
00233       os « **it2;
00234     }
00235
00236     return os;
00237   }
00238
00239   inline std::ostream& operator«(std::ostream& os, const Predicate& p) {
00240     Predicate::const_iterator it;
00241     for( it=p.begin(); it!=p.end(); it++) {
00242       if(it!=p.begin()) {
00243         os « " OR ";
00244       }
00245
00246       os « **it;
00247     }
00248
00249     return os;
00250   }
00251
00252 } // namespace
00253
00254 #endif
```

7.125 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/PredicateUtils.hpp File Reference

```
#include <vector>
#include <set>
#include <string>
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/PredicateUtils/VarSet.hpp"
#include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
#include "minerule/PredicateUtils/Predicate.hpp"
#include <iterator>
```

Data Structures

- class minerule::CountingIterator
- class minerule::PredicateUtils

Namespaces

- namespace minerule

7.126 PredicateUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
```

```
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __PREDICATE_UTILS_H__
00017 #define __PREDICATE_UTILS_H__
00018
00019 #include <vector>
00020 #include <set>
00021 #include <string>
00022
00023 #include "minerule/Parsers/ParsedMinerule.hpp"
00024 #include "minerule/PredicateUtils/VarSet.hpp"
00025 #include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
00026 #include "minerule/PredicateUtils/Predicate.hpp"
00027 #include <iterator>
00028
00029 namespace minerule {
00030   // ------------------------------------------------------------------------
00031   // CountingIterator
00032   // ------------------------------------------------------------------------
00045   class CountingIterator : public std::iterator<std::output_iterator_tag, void, void, void, void> {
00046     size_t& curNum;
00047     list_AND_node*& l_and;
00048   public:
00049     static list_AND_node* new_list_AND_node( const char* v1,
00050                                              const char* op,
00051                                              const char* v2,
00052                                              list_AND_node* oldList );
00053
00054     static void delete_list_AND_node(list_AND_node*&);
00055
00056     CountingIterator(size_t& counter, list_AND_node*& l) : curNum(counter), l_and(l) {
00057     }
00058
00059     CountingIterator(const CountingIterator& c) : curNum(c.curNum), l_and(c.l_and) {}
00060
00061     CountingIterator& operator=(SimplePredicate* const& pred) {
00062       pred->setVarId(curNum);
00063       list_AND_node* tmp = new_list_AND_node( pred->getVal1().c_str(),
00064                                               pred->getOp().c_str(),
00065                                               pred->getVal2().c_str(), l_and );
00066       l_and =tmp;
00067
00068       return *this;
00069     }
00070
00071     CountingIterator& operator*() {
00072       return *this;
00073     }
00074
00075     CountingIterator& operator++(int) {
00076       curNum++;
00077       return *this;
00078     }
00079
00080     CountingIterator& operator++() {
00081       curNum++;
00082       return *this;
00083     }
00084   };
00085
00086
00087
00088
00089   // ------------------------------------------------------------------------
00090   // PredicateUtils
00091   // ------------------------------------------------------------------------
00092
00093   class PredicateUtils {
00094   public:
00095     typedef EncodedNF::CodesRelationship PredicateRelationship;
00096
00097     static bool predicatesAreEquivalent(Predicate& p1,
00098                                         Predicate& p2,
00099                                         const std::string& tab_source);
00100     static PredicateRelationship
00101       getPredicateRelationship( Predicate& p1,
00102                                 Predicate& p2,
```

```
00103                                const std::string& tab_source);
00104
00105
00106
00107   };
00108
00109 } // namespace
00110
00111 #endif
```

7.127 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/SimplePredAnalyzer.hpp File Reference

```
#include <string.h>
#include "minerule/mrdb/Types.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/SQLUtils.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
```

Data Structures

- class minerule::SimplePredAnalyzer

Namespaces

- namespace minerule

7.128 SimplePredAnalyzer.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __SIMPLEPREDANALYZER_H__
00017 #define __SIMPLEPREDANALYZER_H__
00018
00019 #include <string.h>
00020 #include "minerule/mrdb/Types.hpp"
00021 #include "minerule/Utils/MineruleException.hpp"
00022 #include "minerule/Utils/SQLUtils.hpp"
00023 #include "minerule/Parsers/ParsedMinerule.hpp"
00024
00025
00026 namespace minerule {
00027
00028   class SimplePredAnalyzer {
00029   protected:
00030     // operators are ordered as follows: "< <= = >= > <>"
00031     static size_t getOperatorIndex(const char* op) ;
00032     // relations are ordered as follows "v1<v2 v1=v2 v1>v2"
```

```
00033     static size_t getValuesRelationshipIndex(
00034                       const char* value1,
00035                       const char* value2,
00036                       SQLUtils::Type type) ;
00037
00038      /*
00039      * Given two predicates X=(x opx a) and Y=(y opy b)
00040      * the opRelationTable returns a char which identifies
00041      * the relation between X and Y. In order to access the
00042      * table the two operators and the relation between a and b
00043      * must be encoded (the encodingis given by 'getOperatorIndex'
00044      * and 'getValuesRelationshipCode' methods), the result is
00045      * a character which identifiy the type of the relation. For
00046      * instance let us fix X=(A<5) and Y=(A<3), and hypothetize
00047      * that the encoding for '<' is 0, and the encoding for (A,3,5)
00048      * is 2, then opRelationTable[0][0][2] returns '1' which has
00049      * to be interpreted as a left arrow ("<-") which means that
00050      * the second predicate implies the first. More details about
00051      * returned values are described in the implementation file.
00052      */
00053     static const char* opRelationTable[6][6];
00054   public:
00055     static char
00056       getRelation(const char* Xop, const char* Xvalue,
00057                   const char* Yop, const char* Yvalue,
00058                   SQLUtils::Type)  ;
00059
00060     static bool
00061       isAttrOpValuePredicate(simple_pred* X,
00062                              char*& attr,
00063                              char*& value,
00064                              bool& reverseOp);
00065   };
00066 }
00067
00068 #endif
```

7.129 /Users/esposito/Software/minerule/include/minerule/PredicateUtils/VarSet.hpp File Reference

```
#include <iostream>
#include <cassert>
```

Data Structures

- class minerule::VarSet
- class minerule::VarSetEnumerator

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const VarSet &vset)

7.130 VarSet.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __VARSET_H__
00017 #define __VARSET_H__
00018
00019 #include <iostream>
00020 #include <cassert>
00021
00022
00023 /* ************************************************
00024  * In this file two classes are defined:
00025  * - VarSet : allows to define and store  efficiently the truth value of a
00026  *            set of boolean variables (limitation: the current implementation
00027  *            only deals with variable  sets whose size is less than 32 elements)
00028  * - VarSetEnumerator : allows to iterates through all possible subsets
00029  *            that can be formed starting from a VarSet of a given size.
00030  *            It comes handy since the most frequent operation on a ExpressionEvaluator
00031  *            is to iterate through all possible subsets of a VarSet
00032  *            evaluating the expression * in the context provided by each subset.
00033  * ************************************************ */
00034
00035
00036 namespace minerule {
00037
00038 class VarSet {
00039   typedef unsigned int SetType;
00040
00041   SetType vset;
00042   unsigned short vsetSize;
00043
00044  public:
00045   VarSet(size_t numVars) :vset(0), vsetSize(numVars) {
00046     assert( vsetSize < sizeof(SetType)*8 );
00047   }
00048
00049   VarSet() : vset(0), vsetSize(0) {
00050
00051   }
00052
00053   void setSize(size_t numVars) {
00054     assert( numVars < sizeof(SetType)*8 );
00055     vsetSize = numVars;
00056   }
00057
00058   void setVar(size_t varNum, bool value) {
00059     assert(varNum < vsetSize);
00060     size_t mask = 1<<varNum;
00061
00062     if(value) {
00063       vset |= mask;
00064     } else {
00065       vset &= ~mask;
00066     }
00067   }
00068
00069   bool getVar(size_t varNum) const {
00070     assert(varNum < vsetSize);
00071
00072     return (1<<varNum) & vset;
00073   }
00074
00075
00076   // This function is an helper function for VarSetEnumerator class.
00077   // It assumes that the present instance is used to enumerate all
00078   // possible subsets of a set having vsetSize elements. If the
00079   // current instance does NOT represent the last of the subsets that
00080   // can be enumerated it returns true and move to the next subset, otherwise
00081   // it return false.
00082
```

```
00083   bool operator++(int) {
00084     if(vset+1 < (unsigned int)(1«vsetSize)) {
00085       vset++;
00086       return true;
00087     } else
00088       return false;
00089   }
00090
00091   VarSet& operator=(const VarSet& v) {
00092     vset=v.vset;
00093     vsetSize=v.vsetSize;
00094     return *this;
00095   }
00096
00097   bool operator<(const VarSet& v) const {
00098     assert(v.vsetSize==vsetSize);
00099     return vset<v.vset;
00100   }
00101
00102   bool operator==(const VarSet& v) const {
00103     assert(v.vsetSize==vsetSize);
00104     return vset==v.vset;
00105   }
00106
00107   bool operator>(const VarSet& v) const {
00108     assert(v.vsetSize==vsetSize);
00109     return vset>v.vset;
00110   }
00111
00112   size_t size() const {
00113     return vsetSize;
00114   }
00115 };
00116
00117
00118 class VarSetEnumerator {
00119   VarSet varset;
00120   bool firstCall;
00121
00122
00123  public:
00124   VarSetEnumerator(size_t numVars) : varset(numVars), firstCall(true) {
00125
00126   }
00127
00128   const VarSet& operator*() const {
00129     return varset;
00130   }
00131
00132   bool operator++(int)  {
00133     if(firstCall && varset.size()!=0 ) {
00134       firstCall=false;
00135       return true;
00136     }
00137     else
00138       return varset++;
00139   }
00140
00141   // if a parameter val!=0 is provided, it returns the number of valid subsets
00142   // that can be formed from a set having size val. Otherwise it returns the same
00143   // information, but using the current VarSet size in evaluating it.
00144   size_t enumerationSize(size_t val=0) const {
00145     if( val==0 )
00146       val = varset.size();
00147
00148     return (size_t)(1«val);
00149   }
00150 };
00151
00152
00153 // streaming operators
00154 inline std::ostream& operator«(std::ostream& os, const VarSet& vset) {
00155   for(size_t i=0; i<vset.size(); i++) {
00156     if(vset.getVar(i))
00157       os « "1";
00158     else
00159       os « "0";
00160   }
00161   return os;
00162 }
00163
00164
00165 } // namespace minerule
00166
00167
00168 #endif
```

7.131 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtable.hpp File Reference

Data Structures

- class minerule::TranslatedTable

Namespaces

- namespace minerule

7.132 translatedtable.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __TRANSLATEDTABLE_H__
00017 #define __TRANSLATEDTABLE_H__
00018
00019
00020
00021
00022 namespace minerule {
00023
00024 class TranslationManager;
00025
00031 class TranslatedTable {
00032 public:
00037   virtualstd::string getTranslatedName() const = 0;
00038
00042 virtual std::string getOriginalName() const = 0;
00043
00051 virtual std::string getTranslatedColumnName( const std::string& columnName) const = 0;
00052
00053   /* Cannot see any use for this method... if needed I will add it in future
00054    * @param translatedColumnName a column name of the translated table
00055    * @return the name, in the original table, of the column named
00056    * translatedColumnName in the translated table.
00057    */
00058 // virtual std::string getOriginalColumnName( const std::string& translatedColumnName ) const = 0;
00059
00065 virtual std::string getTranslatedValue( const std::string& columnName,
00066                                         const std::string& value) const = 0;
00067
00073 virtual std::string getOriginalValue( const std::string& columnName,
00074                                       const std::string& translatedValue) const = 0;
00075 };
00076
00077 }
00078
00079 #endif
```

7.133 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translatedtablestandardsql.hpp File Reference

```
#include "minerule/Utils/common.hpp"
#include "translatedtable.hpp"
#include <ext/hash_map>
```

Data Structures

- class minerule::Scmp
- class minerule::TranslatedTableStandardSQL

Namespaces

- namespace minerule

Macros

- #define __TRANSLATEDTABLESTANDARDSQL_H_

7.133.1 Macro Definition Documentation

7.133.1.1 __TRANSLATEDTABLESTANDARDSQL_H_

```
#define __TRANSLATEDTABLESTANDARDSQL_H_
```

Definition at line 17 of file translatedtablestandardsql.hpp.

7.134 translatedtablestandardsql.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __TRANSLATEDTABLESTANDARDSQL_H__
00017 #define __TRANSLATEDTABLESTANDARDSQL_H_
00018
00019
```

```
00025 #include "minerule/Utils/common.hpp"
00026 #include "translatedtable.hpp"
00027 #include <ext/hash_map>
00028
00029
00030
00031 using namespace __gnu_cxx;
00032
00033 namespace minerule {
00034
00035 class TranslationManagerStandardSQL;
00036
00037 // This class is needed by the hash_map used in TranslatedTableStandardSQL
00038 class Scmp {
00039 public:
00040   int operator()(const char* s1, const char* s2) const {
00041     return std::strcmp(s1,s2)==0;
00042   }
00043 };
00044
00045 class TranslatedTableStandardSQL : public TranslatedTable {
00046   friend class TranslationManagerStandardSQL;
00047
00048  private:
00049   const TranslationManagerStandardSQL* tmanager;
00050  std::string originalTable;
00051  std::string translatedTable;
00052   bool translateEverything;
00053   typedef hash_map<
00054     const char*,
00055     bool,
00056     hash<const char*>,
00057     Scmp> ColumnDictionary;
00058
00059   ColumnDictionary columnsToTranslate ;
00060
00061  protected:
00062   void
00063     setOriginalTableName(const std::string& table) {
00064     originalTable = table;
00065   }
00066
00067   void
00068     setTranslatedTableName(const std::string& table) {
00069     translatedTable = table;
00070   }
00071
00079   virtual void updateColumnsToTranslate();
00080
00081  public:
00082   TranslatedTableStandardSQL(const TranslationManagerStandardSQL* manager,
00083                              const std::string& parOriginalTable,
00084                              const std::string& parTranslatedTable,
00085                              bool parTranslateEverything) :
00086     tmanager(manager), originalTable(parOriginalTable), translatedTable(parTranslatedTable) {
00087     translateEverything = parTranslateEverything;
00088
00089     updateColumnsToTranslate();
00090   }
00091
00092
00093
00098   virtualstd::string getTranslatedName() const  ;
00099
00103   virtualstd::string getOriginalName() const  ;
00104
00112   virtualstd::string getTranslatedColumnName( const std::string& columnName) const  ;
00113
00114   /* I sincerely cannot see any applicatio for this method... if needed
00115    * I will add it to the interface.
00116    *
00117    * @param translatedColumnName a column name of the translated table
00118    * @return the name, in the original table, of the column named
00119    * translatedColumnName in the translated table.
00120    */
00121   //std::string getOriginalColumnName( const std::string& translatedColumnName ) const = 0;
00122
00128   virtualstd::string getTranslatedValue( const std::string& columnName,
00129                              const std::string& value) const  ;
00130
00136 virtual std::string getOriginalValue( const std::string& columnName,
00137                            const std::string& translatedValue) const  ;
00138
00139
00140  virtual ~TranslatedTableStandardSQL();
00141 };
00142 }
```

```
00143
00144 #endif
```

7.135 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanager.hpp File Reference

```
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/mrdb/Types.hpp"
#include "translatedtable.hpp"
```

Data Structures

- class minerule::TranslationManager

Namespaces

- namespace minerule

Macros

- #define BBLOG(a) do { std::cerr << a << std::endl; } while(0)

7.135.1 Macro Definition Documentation

7.135.1.1 BBLOG

```
#define BBLOG(
              a ) do { std::cerr << a << std::endl; } while(0)
```

Definition at line 27 of file translationmanager.hpp.

7.136 translationmanager.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __TABLETRANSLATIONMANAGER__H__
00017 #define __TABLETRANSLATIONMANAGER__H__
00018
00019 #include"minerule/mrdb/Connection.hpp"
00020 #include"minerule/mrdb/ResultSet.hpp"
00021 #include"minerule/mrdb/Statement.hpp"
00022 #include"minerule/mrdb/ResultSetMetaData.hpp>
00023 #include"minerule/mrdb/Types.hpp"
00024
00025 using namespace::std;
00026
00027 #define BBLOG(a) do { std::cerr « a « std::endl; } while(0)
00028
00029 #include "translatedtable.hpp"
00030
00031
00052 namespace minerule {
00053
00054 class TranslationManager {
00055  protected:
00066  std::string
00067     getTranslationTableNameForColumn(const std::string& tableName,
00068                                      const std::string& columnName) const;
00069
00078   string
00079     getTranslatedColumnNameForColumn(const std::string& tableName,
00080                                      const std::string& columnName) const;
00089   string
00090     getOriginalColumnNameForColumn(const std::string& tableName,
00091                                    const std::string& columnName) const;
00102   string
00103     getTranslationTableNameForTable( const std::string& tableName ) const;
00104  public:
00105
00123   virtual TranslatedTable*
00124     translateTable(const std::string& tableName) const = 0;
00125
00140   virtual pair<bool,string>
00141     alreadyTranslated(const std::string& tableName) const
00142     = 0;
00143 };
00144
00145 }
00146 #endif
```

7.137 /Users/esposito/Software/minerule/include/minerule/PreProcessor/translationmanagerstandardsql.hpp File Reference

```
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/mrdb/Types.hpp"
#include "translationmanager.hpp"
```

Data Structures

- class minerule::TranslationManagerStandardSQL

Namespaces

- namespace minerule

7.138 translationmanagerstandardsql.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __TABLETRANSLATIONMANAGERSTANDARDSQL__H__
00017 #define __TABLETRANSLATIONMANAGERSTANDARDSQL__H__
00018
00019 #include"minerule/mrdb/Connection.hpp"
00020 #include"minerule/mrdb/ResultSet.hpp"
00021 #include"minerule/mrdb/Statement.hpp"
00022 #include"minerule/mrdb/ResultSetMetaData.hpp>
00023 #include"minerule/mrdb/Types.hpp"
00024
00025 #include "translationmanager.hpp"
00026
00034 namespace minerule {
00035
00036 class TranslationManagerStandardSQL : public TranslationManager {
00037   friend class TranslatedTableStandardSQL;
00038  private:
00039   // data members
00043   mrdb::Connection* connection;
00044
00050   bool translateEverything;
00051
00052   // static constants ============================================
00053
00058   static conststd::string TM_PREFIX;
00063   static conststd::string TABLEDICTIONARY_NAME;
00064
00069   static conststd::string TABLEDICTIONARY_ORIGINAL_TABLE;
00073   static const int TABLEDICTIONARY_ORIGINAL_TABLE_LEN;
00078   static conststd::string TABLEDICTIONARY_TRANSLATED_TABLE;
00082   static const int TABLEDICTIONARY_TRANSLATED_TABLE_LEN;
00083
00084   // private methods =============================================
00085
00090   void ensureTableDictionaryExist() const ;
00091
00096   bool tableDictionaryExists() const  ;
00097
00107   void doTranslateTable(const std::string& tableName)
00108     const ;
00109
00110
00118   bool shouldTranslateColumnsOfType(int colType) const;
00119
00120
00125   virtualstd::string quote(const std::string&) const;
00126
00134   virtualstd::string quoteAttribute(const std::string&) const;
00135
00141   virtualstd::string getIntTypeName() const {
00142     return std::string("INTEGER");
```

```
00143   }
00144
00145   void updateTManagerDictionary(const std::string& table) const ;
00146
00153   virtual void createTranslationTableForColumn(
00154                 const std::string& table,
00155                 int colPos,
00156                 mrdb::ResultSetMetaData& ) const  ;
00157
00163   virtual void doTranslateColumn(
00164                     const std::string& tableName,
00165                     int colPos,
00166                     mrdb::ResultSetMetaData&
00167                     ) const ;
00168
00172   virtual void dropTranslationForColumn(
00173                      const std::string& tableName,
00174                     int colPos,
00175                     mrdb::ResultSetMetaData&
00176                     ) const ;
00177
00191   virtual void doSetResultSetParameter(
00192                       mrdb::PreparedStatement* pstm,
00193                       mrdb::ResultSet* rs,
00194                       mrdb::ResultSetMetaData* rsmd) const;
00195
00196 virtual void buildLists(
00197             const std::string& table,
00198             mrdb::ResultSetMetaData* rsmd,
00199             std::string& newTable,
00200             std::string& selectList,
00201             std::string& tableList,
00202             std::string& conditionList ) const  ;
00203
00204
00205 void finalizeTranslationOfTable(
00206                                 const std::string& table,
00207                                 mrdb::ResultSetMetaData* rsmd
00208                                 ) const  ;
00209
00210
00211 TranslatedTable*
00212   newAccessPointForTable(const std::string& tableName) const  ;
00213   // public methods ============================================================
00214  public:
00223   TranslationManagerStandardSQL(mrdb::Connection* aConnection,
00224                                 bool translateEverythingOption=true);
00225
00226
00231   TranslatedTable*
00232     translateTable(const std::string& tableName) const ;
00233
00238   pair<bool,string>
00239     alreadyTranslated(const std::string& tableName) const ;
00240 };
00241
00242 } // namespace minerule
00243
00244 #endif
```

7.139 /Users/esposito/Software/minerule/include/minerule/Result/↩ QueryResult-header.hpp File Reference

```
#include "minerule/Database/ItemType.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
#include "minerule/Result/Rule.hpp"
#include "minerule/mrdb/Statement.hpp"
```

Data Structures

- class minerule::QueryResult

- class minerule::QueryResult::Iterator
- class minerule::QueryResult::FastSorter
- class minerule::QueryResult::SortBodyHeadSuppConf
- class minerule::QueryResult::SortBodyHead
- class minerule::QueryResult::SortHeadBodySuppConf
- class minerule::QueryResult::SortSuppConfBodyHead
- class minerule::QueryResult::SortConfSuppBodyHead
- class minerule::QueryResult::SortConfBodyHeadSupp
- class minerule::QueryResult::SortConfBodySuppHead
- class minerule::QueryResult::ResultSet< Sorter >

Namespaces

- namespace minerule

7.140 QueryResult-header.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __QUERY_RESULT_HEADER_H__
00017 #define __QUERY_RESULT_HEADER_H__
00018
00019 #include "minerule/Database/ItemType.hpp"
00020 #include "minerule/Parsers/ParsedMinerule.hpp"
00021 #include "minerule/mrdb/PreparedStatement.hpp"
00022 #include "minerule/Result/Rule.hpp"
00023 #include "minerule/mrdb/Statement.hpp"
00024
00025 namespace minerule {
00026
00027         class QueryResult {
00028         public:
00029
00030         // ------------------------------------------------------------------
00031         // Rule
00032         // ------------------------------------------------------------------
00033
00034                 // ------------------------------------------------------------------------------
00035                 // Iterator
00036                 // ------------------------------------------------------------------------------
00037
00038
00039                 class Iterator {
00040                 private:
00041                         mrdb::Statement* state;
00042                         mrdb::ResultSet* rs_rules;
00043                         mrdb::PreparedStatement* body_elems;
00044                         mrdb::PreparedStatement* head_elems;
00045
00046                         void readElems(int id, ItemSet& elems, mrdb::PreparedStatement* elems_rs);
00047                 public:
00048                         Iterator() : state(NULL), rs_rules(NULL), body_elems(NULL), head_elems(NULL)
    {}
00049
00050                         ~Iterator() {
00051                                 if(rs_rules!=NULL)
00052                                         delete rs_rules;
00053
```

```
00054                                 if(state!=NULL)
00055                                         delete state;
00056
00057                                 if( body_elems!=NULL )
00058                                         delete body_elems;
00059
00060                                 if( head_elems!=NULL )
00061                                         delete head_elems;
00062                         }
00063
00064
00065                 //
00066                     // initialize the result set
00067                 //
00068                         void init( const std::string& rulesTable, double support, double confidence )
    ;
00069
00070                     // it moves the iterator on the following rule. The first time it is called
00071                     // it position on the first element. It return false when it cannot
00072                     // succeed in positioning on the next rule.
00073                         bool next();
00074
00075                 //
00076                     // It fills the Rule given as input with the current
00077                     // values.
00078                 //
00079
00080                         void getRule( Rule& r ) ;
00081                   };
00082
00083                   // ----------------------------------------------------------------------------
00084                   // FastSorter
00085                   // ----------------------------------------------------------------------------
00086                   // Implements the fastest possible sorting mechanisms.
00087                   class FastSorter {
00088                   public:
00089                         bool operator()(const Rule& r1, const Rule& r2) {
00090                                 return &r1 < &r2;
00091                         }
00092                   };
00093
00094
00095                   // ----------------------------------------------------------------------------
00096                   // SortBodyHeadSuppConf
00097                   // ----------------------------------------------------------------------------
00098
00099                   class SortBodyHeadSuppConf {
00100                   public:
00101                         bool operator()(const Rule& r1, const Rule& r2) const {
00102                                 if( r1.getBody() > r2.getBody() )
00103                                         return true;
00104
00105                                 if( r1.getBody() == r2.getBody() ) {
00106                                         if(r1.getHead() > r2.getHead() )
00107                                                 return true;
00108
00109                                         if(r1.getHead() == r2.getHead()) {
00110                                                 return r1.getSupport() > r2.getSupport() ||
00111                                                         (r1.getSupport()==r2.getSupport() &&
    r1.getConfidence() > r2.getConfidence());
00112                                         }
00113                                 }
00114
00115                                 return false;
00116                         }
00117                   };
00118
00119
00120                   // ----------------------------------------------------------------------------
00121                   // SortBodyHead
00122                   // ----------------------------------------------------------------------------
00123
00124                   class SortBodyHead {
00125                   public:
00126                         bool operator()(const Rule& r1, const Rule& r2) const {
00127                                 if( r1.getBody() > r2.getBody() )
00128                                         return true;
00129
00130                                 if( r1.getBody() == r2.getBody() ) {
00131                                         if(r1.getHead() > r2.getHead() )
00132                                                 return true;
00133                                 }
00134
00135                                 return false;
00136                         }
00137                   };
00138
```

```
00139
00140                 // ------------------------------------------------------------------------------
00141                 // SortHeadBodySuppConf
00142                 // ------------------------------------------------------------------------------
00143
00144                 class SortHeadBodySuppConf {
00145                 public:
00146                         bool operator()(const Rule& r1, const Rule& r2) const {
00147                                 if( r1.getHead() > r2.getHead() )
00148                                         return true;
00149
00150                                 if( r1.getHead() == r2.getHead() ) {
00151                                         if(r1.getBody() > r2.getBody() )
00152                                                 return true;
00153
00154                                         if(r1.getBody() == r2.getBody()) {
00155                                                 return r1.getSupport() > r2.getSupport() ||
00156                                                         (r1.getSupport()==r2.getSupport() &&
    r1.getConfidence() > r2.getConfidence());
00157                                         }
00158                                 }
00159
00160                                 return false;
00161                         }
00162                 };
00163
00164
00165                 // ------------------------------------------------------------------------------
00166                 // SortSuppConf
00167                 // ------------------------------------------------------------------------------
00168
00169                 class SortSuppConfBodyHead {
00170                 public:
00171                         bool operator()(const Rule& r1, const Rule& r2) const {
00172                                 if(r1.getSupport() > r2.getSupport())
00173                                         return true;
00174                                 if(r1.getSupport() < r2.getSupport())
00175                                         return false;
00176
00177         // here r1.getSupport() == r2.getSupport()
00178                                 if(r1.getConfidence() > r2.getConfidence())
00179                                         return true;
00180                                 if(r1.getConfidence() < r2.getConfidence())
00181                                         return false;
00182
00183         // here r1.sup=r2.sup && r1.conf=r2.conf
00184                                 if(r1.getBody() > r2.getBody())
00185                                         return true;
00186                                 if(r1.getBody() < r2.getBody())
00187                                         return false;
00188
00189                                 if(r1.getHead() > r2.getHead())
00190                                         return true;
00191                                 if(r1.getHead() < r2.getHead())
00192                                         return false;
00193
00194         // here all fields are equals
00195                                 return false;
00196                         }
00197                 };
00198
00199                 // ------------------------------------------------------------------------------
00200                 // SortConfSuppBodyHead
00201                 // ------------------------------------------------------------------------------
00202
00203
00204                 class SortConfSuppBodyHead {
00205                 public:
00206                         bool operator()(const Rule& r1, const Rule& r2) const {
00207                                 if(r1.getConfidence() > r2.getConfidence())
00208                                         return true;
00209                                 if(r1.getConfidence() < r2.getConfidence())
00210                                         return false;
00211
00212                                 if(r1.getSupport() > r2.getSupport())
00213                                         return true;
00214                                 if(r1.getSupport() < r2.getSupport())
00215                                         return false;
00216
00217                                 if(r1.getBody() > r2.getBody())
00218                                         return true;
00219                                 if(r1.getBody() < r2.getBody())
00220                                         return false;
00221
00222                                 if(r1.getHead() > r2.getHead())
00223                                         return true;
00224                                 if(r1.getHead() < r2.getHead())
```

```
00225                                   return false;
00226
00227         // here all fields are equals
00228                                   return false;
00229                           }
00230                   };
00231
00232                   // ------------------------------------------------------------------------------
00233                   // SortConfBodyHeadSupp
00234                   // ------------------------------------------------------------------------------
00235
00236
00237                   class SortConfBodyHeadSupp {
00238                   public:
00239                           bool operator()(const Rule& r1, const Rule& r2) const {
00240                                   if(r1.getConfidence() > r2.getConfidence())
00241                                           return true;
00242                                   if(r1.getConfidence() < r2.getConfidence())
00243                                           return false;
00244
00245                                   if(r1.getBody() > r2.getBody())
00246                                           return true;
00247                                   if(r1.getBody() < r2.getBody())
00248                                           return false;
00249
00250                                   if(r1.getHead() > r2.getHead())
00251                                           return true;
00252                                   if(r1.getHead() < r2.getHead())
00253                                           return false;
00254
00255                                   if(r1.getSupport() > r2.getSupport())
00256                                           return true;
00257                                   if(r1.getSupport() < r2.getSupport())
00258                                           return false;
00259
00260                                   return false;
00261                           }
00262                   };
00263
00264                   // ------------------------------------------------------------------------------
00265                   // SortConfBodySuppHead
00266                   // ------------------------------------------------------------------------------
00267
00268
00269                   class SortConfBodySuppHead {
00270                   public:
00271                           bool operator()(const Rule& r1, const Rule& r2) const {
00272                                   if(r1.getConfidence() > r2.getConfidence())
00273                                           return true;
00274                                   if(r1.getConfidence() < r2.getConfidence())
00275                                           return false;
00276
00277                                   if(r1.getBody() > r2.getBody())
00278                                           return true;
00279                                   if(r1.getBody() < r2.getBody())
00280                                           return false;
00281
00282                                   if(r1.getSupport() > r2.getSupport())
00283                                           return true;
00284                                   if(r1.getSupport() < r2.getSupport())
00285                                           return false;
00286
00287                                   if(r1.getHead() > r2.getHead())
00288                                           return true;
00289                                   if(r1.getHead() < r2.getHead())
00290                                           return false;
00291
00292         // here all fields are equals
00293                                   return false;
00294                           }
00295                   };
00296
00297
00298
00299         // ------------------------------------------------------------------------------
00300         // ResultSet fields and methods (at last ;-))
00301         // ------------------------------------------------------------------------------
00302
00303         // This class allows one to load in memory a whole result set.
00304         // It also allows to perform intersections and unions of different
00305         // query results, as well as to sort the result using different
00306         // criteria.
00307
00308     template <class Sorter>
00309                   class ResultSet : public  std::set<Rule,Sorter> {
00310
00311                   public:
```

```
00312                         ResultSet() {}
00313                         ResultSet(const ResultSet<Sorter>& rhs) {}
00314
00315
00316                 // Load a result set into the object. NOTICE: this function (this class
00317                 // at large) should be used when you need to load a full result set
00318                 // into memory. Do not use it as a short-hand for iterating through
00319                 // a result set. Use the ResultSetIterator class instead (you can
00320                 // initialize it using the OptimizerCatalogue. Another other reason
00321                 // that justify the use of this class is when you need to iterate thru
00322                 // the result set, but you require it to be sorted).
00323                 //
00324                 // @param qryName the name of the query to load into the class
00325                 // @param sup Support threshold used to filter out uninteresting rules.
00326                 // The default is -1 and it means that no filter should occur.
00327                 // @param conf Confidence threshold used to filter out uninteresting rules.
00328                 // The default is -1 and it means that no filter should occur.
00329
00330                         void load(const std::string& qryName, double sup=-1, double con=-1) ;
00331
00332                 //
00333                 // Save the current result set. The tables being created are associated
00334                 // to the minerule mr. Notice that this function do not update the catalogue.
00335                 //
00336
00337                         void save(const ParsedMinerule& mr) const;
00338
00339                         ResultSet& inplace_intersect(const ResultSet&);
00340                         ResultSet& inplace_union(const ResultSet&);
00341
00342                 //
00343                 // It builds the intersection of rhs and *this. The result is stored
00344                 // into *this.
00345                 // Notice that whether two rules has to be considered equal depends
00346                 // on the Sorter you choose. For instance if you use the SortSuppConf
00347                 // sorter, then two queries are equal iff
00348                 //
00349                 // @param rhs the query that has to be intersected with *this
00350                 //
00351                         ResultSet& operator&=(const ResultSet&);
00352
00353                 //
00354                 // It builds the union of rhs and *this. The result is stored
00355                 // into *this.
00356                 //
00357                 // @param rhs the query that has to be merged with *this
00358                 //
00359                         ResultSet& operator|=(const ResultSet&);
00360
00361                 }; // ResultSet
00362
00363         }; // QueryResult
00364 } // namespace
00365
00366 #endif
```

7.141 /Users/esposito/Software/minerule/include/minerule/Result/↩ QueryResult-impl.hpp File Reference

```
#include <algorithm>
#include "minerule/Result/QueryResult.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
```

Namespaces

- namespace minerule

7.142 QueryResult-impl.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __QUERY_RESULT_IMPL_H__
00017 #define __QUERY_RESULT_IMPL_H__
00018
00019 #include <algorithm>
00020 #include "minerule/Result/QueryResult.hpp"
00021 #include "minerule/Database/Connection.hpp"
00022 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00023
00024 namespace minerule {
00025
00026   template <class Sorter>
00027         void
00028         QueryResult::ResultSet<Sorter>::load(const std::string& qryName, double sup, double con) {
00029                 QueryResult::Iterator qit;
00030                 OptimizerCatalogue::getMRQueryResultIterator(qryName,qit,sup,con);
00031
00032                 while( qit.next() ) {
00033                         Rule r;
00034                         qit.getRule(r);
00035
00036                         std::set<Rule,Sorter>::insert(r);
00037                 }
00038         }
00039
00040
00041  //
00042  // Complexity note: let this->size()=n, rhs.size()=m, and let o denote
00043  // the size of the intersection.
00044  // This implementation requires:
00045  //  m+n steps to iterate thru the two lists
00046  //  o*log(o) steps to inser the results into the tmp variable
00047  //  o*log(o) steps to copy the result into *this
00048  //
00049  // The implementation that deletes the nodes which are not found in
00050  // *this, would require:
00051  // m steps to iterate thru rhs
00052  // m*log(n) steps to search the objects in this
00053  // (n-o)log(n) (approximated) steps to delete the object that do *
00054  // not belong to the intersection from *this.
00055  //
00056  // The analysis about when one is better than the other is a little
00057  // bit complicated, however, it roughly speaking, it seems
00058  // reasonable to say that if o is very small then this
00059  // implementation should be a good one, since it is ~= m+n while the
00060  // second one is ~= m+mlog(n)+nlog(n). On the other side, if o~=n
00061  // then this implementation is m+n+2nlog(n), while the other one is
00062  // m+mlog(n)+nlog(n).
00063  //
00064
00065   template <class Sorter>
00066         QueryResult::ResultSet<Sorter>& QueryResult::ResultSet<Sorter>::inplace_intersect(const
       ResultSet<Sorter>& rhs) {
00067                 QueryResult::ResultSet<Sorter> tmp;
00068                 set_intersection( this->begin(), this->end(), rhs.begin(),rhs.end(),
00069                         std::insert_iterator<ResultSet<Sorter> >( tmp, tmp.begin() ), Sorter());
00070
00071                 *this = tmp;
00072                 return *this;
00073         }
00074
00075   template <class Sorter>
00076         QueryResult::ResultSet<Sorter>&   QueryResult::ResultSet<Sorter>::inplace_union(const
       ResultSet<Sorter>& rhs) {
00077
00078                 copy( rhs.begin(), rhs.end(),
00079                         std::insert_iterator<ResultSet<Sorter> >(*this, this->begin()));
00080                 return *this;
```

```
00081         }
00082
00083   template <class Sorter>
00084         QueryResult::ResultSet<Sorter>& QueryResult::ResultSet<Sorter>::operator&=(const
      ResultSet<Sorter>& rhs) {
00085                 return inplace_intersect(rhs);
00086         }
00087    template <class Sorter>
00088         QueryResult::ResultSet<Sorter>& QueryResult::ResultSet<Sorter>::operator|=(const
      ResultSet<Sorter>& rhs) {
00089                 return inplace_union(rhs);
00090         }
00091
00092
00093   template <class Sorter>
00094         void QueryResult::ResultSet<Sorter>::save(const ParsedMinerule& mr) const {
00095                 Connection connection;
00096                 connection.setOutTableName(mr.tab_result);
00097                 connection.setBodyCardinalities(mr.bodyCardinalities);
00098                 connection.setHeadCardinalities(mr.headCardinalities);
00099                 connection.useMRDBConnection(
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00100
00101                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(), mr));
00102
00103                 typename ResultSet<Sorter>::const_iterator it;
00104                 for(it=this->begin();it!=this->end();it++) {
00105                         connection.insert( it->getBody(), it->getHead(), it->getSupport(),
      it->getConfidence() );
00106                 }
00107         }
00108 }
00109
00110 #endif
```

7.143 /Users/esposito/Software/minerule/include/minerule/Result/↩ QueryResult.hpp File Reference

```
#include "minerule/Result/QueryResult-header.hpp"
#include "minerule/Result/QueryResult-impl.hpp"
```

7.144 QueryResult.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __QUERY_RESULT_H__
00017 #define __QUERY_RESULT_H__
00018
00019 #include "minerule/Result/QueryResult-header.hpp"
00020 #include "minerule/Result/QueryResult-impl.hpp"
00021
00022 #endif
```

7.145 /Users/esposito/Software/minerule/include/minerule/Result/Rule.hpp File Reference

```
#include "minerule/Database/ItemType.hpp"
```

Data Structures

- class minerule::Rule

Namespaces

- namespace minerule

7.146 Rule.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef RULE_H_SP35S3OZ
00017 #define RULE_H_SP35S3OZ
00018
00019 #include "minerule/Database/ItemType.hpp"
00020
00021 namespace minerule {
00022
00023         class Rule {
00024         private:
00025                 unsigned int bodyId;
00026                 unsigned int headId;
00027                 ItemSet* body;
00028                 ItemSet* head;
00029                 double support;
00030                 double confidence;
00031         protected:
00032                 void clear() {
00033                         if(body!=NULL)
00034                                 delete body;
00035                         if(head!=NULL)
00036                                 delete head;
00037
00038                         body=NULL;
00039                         head=NULL;
00040                 }
00041
00042         public:
00043                 Rule() : body(NULL), head(NULL), support(0), confidence(0) {}
00044
00045                 Rule(const Rule& rule) : support(rule.support), confidence(rule.confidence),
       bodyId(rule.bodyId), headId(rule.headId) {
00046                         if(rule.body!=NULL) {
00047                                 body= new ItemSet(*rule.body);
00048                         }
00049
00050                         if(rule.head!=NULL) {
00051                                 head= new ItemSet(*rule.head);
```

```
00052                   }
00053           }
00054
00055           ~Rule() {
00056                   clear();
00057           }
00058
00059           bool operator<(const Rule& rhs) const {
00060                   return *this->body < *rhs.body || ( *this->body == *rhs.body && *this->head <
        *rhs.head );
00061           }
00062
00063           bool operator<=(const Rule& rhs) const {
00064                   return *this->body <= *rhs.body || (*this->body == *rhs.body && *this->head <=
        *rhs.head);
00065           }
00066
00067           bool operator==(const Rule& rhs) const {
00068                   return *this->body == *rhs.body && *this->head == *rhs.head;
00069           }
00070
00071           bool operator>(const Rule& rhs) const {
00072                   return ! (*this <= rhs);
00073           }
00074
00075           bool operator>=(const Rule& rhs) const {
00076                   return ! (*this < rhs);
00077           }
00078
00079           bool operator!=(const Rule& rhs) const {
00080                   return ! (*this == rhs);
00081           }
00082
00083           Rule& operator=(const Rule& rule) {
00084                   clear();
00085                   support=rule.support;
00086                   confidence=rule.confidence;
00087
00088                   if(rule.body!=NULL) {
00089                           body= new ItemSet(*rule.body);
00090                   }
00091
00092                   if(rule.head!=NULL) {
00093                           head= new ItemSet(*rule.head);
00094                   }
00095
00096                   return *this;
00097           }
00098
00099           void setBody(ItemSet* b) {
00100                   body = b;
00101           }
00102
00103           const ItemSet& getBody() const {
00104                   assert(body!=NULL);
00105                   return *body;
00106           }
00107
00108           ItemSet& getBody() {
00109                   assert(body!=NULL);
00110                   return *body;
00111           }
00112
00113
00114
00115           void setHead(ItemSet* h) {
00116                   head = h;
00117           }
00118
00119           const ItemSet& getHead() const {
00120                   assert(head!=NULL);
00121                   return *head;
00122           }
00123
00124           ItemSet& getHead() {
00125                   assert(head!=NULL);
00126                   return *head;
00127           }
00128
00129
00130           void setSupport(double s)  {
00131                   support = s;
00132           }
00133
00134           double getSupport() const {
00135                   return support;
00136           }
```

```
00137
00138                 void setConfidence(double c) {
00139                         confidence = c;
00140                 }
00141
00142                 double getConfidence() const {
00143                         return confidence;
00144                 }
00145
00146                 // Rules id getters and setters
00147                 void setBodyId(unsigned int bid) { bodyId = bid; }
00148                 void setHeadId(unsigned int hid) { headId = hid; }
00149                 unsigned int getBodyId() const { return bodyId; }
00150                 unsigned int getHeadId() const { return headId; }
00151
00152         };
00153
00154 } // namespace
00155
00156 #endif /* end of include guard: RULE_H_SP35S3OZ */
00157
00158
```

7.147 /Users/esposito/Software/minerule/include/minerule/Result/RuleFormatter.hpp File Reference

```
#include <vector>
#include <iostream>
#include <set>
#include "minerule/Result/QueryResult.hpp"
```

Data Structures

- class minerule::RuleFormatter
- class minerule::RuleFormatter::FieldWidths
- class minerule::SimpleRuleFormatter
- class minerule::SortedRuleFormatter< Sorter >

Namespaces

- namespace minerule

7.148 RuleFormatter.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __RULEFORMATTER_H__
```

```
00017 #define __RULEFORMATTER_H__
00018
00019 #include <vector>
00020 #include <iostream>
00021 #include <set>
00022 #include "minerule/Result/QueryResult.hpp"
00023
00024
00025 namespace minerule {
00029         class RuleFormatter {
00030         public:
00031                 class FieldWidths {
00032                 public:
00033                         size_t body;  // width of the body column
00034                         size_t head;  // width of the head column
00035                         size_t supp;
00036                         size_t conf;
00037
00038                         FieldWidths(size_t b,size_t h,size_t s,size_t c) : body(b), head(h), supp(s),
      conf(c) {}
00039                 };
00040
00041         protected:
00042                 bool _suppressLog;
00043
00044                 std::string _bhSep;   // separates body from head
00045                 FieldWidths _fieldWidths;
00046
00047
00048                 static std::string quote(const std::string& elem);
00049                 static std::string quoteElems(const ItemSet& elems);
00050         public:
00051                 RuleFormatter() : _suppressLog(false), _bhSep("=>"), _fieldWidths(30,30,9,9) {
00052                 }
00053
00054                 virtual ~RuleFormatter() {}
00055
00056                 virtual std::string formatRule(const Rule&, bool includeSuppConf=true) = 0;
00057                 virtual void printRule(const Rule&) = 0;
00058
00059                 void setSuppressLog(bool newVal) {  _suppressLog = newVal; }
00060                 bool suppressLog() const { return _suppressLog; }
00061
00062                 void setFieldWidths( const FieldWidths& f ) { _fieldWidths = f; }
00063                 FieldWidths& fieldWidths() { return _fieldWidths; }
00064                 const FieldWidths& fieldWidths() const { return _fieldWidths; }
00065
00066                 virtual void
00067                         postExec()=0;
00068         };
00069
00070
00071         class SimpleRuleFormatter : public RuleFormatter {
00072         public:
00073                 SimpleRuleFormatter() : RuleFormatter() {};
00074
00075                 virtual ~SimpleRuleFormatter() {};
00076
00077                 virtual std::string formatRule(const Rule&, bool=true);
00078                 virtual void printRule(const Rule&);
00079
00080                 virtual void
00081                         postExec() {};
00082         };
00083
00084
00085 template <class Sorter>
00086         class SortedRuleFormatter : public SimpleRuleFormatter {
00087                 typedef std::set<Rule,Sorter> SortedContainer;
00088                 SortedContainer sortedRules;
00089         public:
00090                 SortedRuleFormatter() : SimpleRuleFormatter() {};
00091                 virtual ~SortedRuleFormatter() {};
00092
00093                 virtual void printRule(const Rule&);
00094
00095                 virtual void postExec();
00096         };
00097
00098 template <class Sorter>
00099         void SortedRuleFormatter<Sorter>::printRule(const Rule& r) {
00100                 sortedRules.insert(r);
00101         }
00102
00103 template <class Sorter>
00104         void SortedRuleFormatter<Sorter>::postExec() {
00105                 typename SortedContainer::const_iterator it;
```

```
00106                 for(it=sortedRules.begin(); it!=sortedRules.end(); it++) {
00107                         SimpleRuleFormatter::printRule(*it);
00108                 }
00109         }
00110
00111 } // namespace
00112
00113 #endif
```

7.149 /Users/esposito/Software/minerule/include/minerule/Result/RulesMatcher.hpp File Reference

```
#include "minerule/Database/Transaction.hpp"
#include "minerule/Result/Rule.hpp"
```

Data Structures

- class minerule::RulesMatcher

Namespaces

- namespace minerule

7.150 RulesMatcher.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef RULESMATCHER_H_2OI7RAD0
00017 #define RULESMATCHER_H_2OI7RAD0
00018
00019 #include "minerule/Database/Transaction.hpp"
00020 #include "minerule/Result/Rule.hpp"
00021
00022 namespace minerule {
00023         class RulesMatcher {
00024         public:
00025                 typedef std::set<ItemType> ItemSetType;
00026                 typedef std::set< std::pair<ItemType, ItemType> > RuleSetType;
00027
00028                 static bool match( const Rule& r, const RuleTransaction<RuleSetType>& t );
00029                 static bool match( const Rule& r, const ItemTransaction<ItemSetType>& bodySet, const
       ItemTransaction<ItemSetType>& headSet );
00030         };
00031
00032 }
00033
00034 #endif /* end of include guard: RULESMATCHER_H_2OI7RAD0 */
```

7.151 /Users/esposito/Software/minerule/include/minerule/Utils/↩ AlgorithmTypes.hpp File Reference

```
#include <string>
#include "minerule/Utils/MineruleException.hpp"
```

Namespaces

- namespace minerule

Enumerations

- enum minerule::MiningTasks { minerule::MTMineRules =0 , minerule::MTMineItemsets , minerule::MTMineSequences , minerule::MTEnd }
- enum minerule::AlgorithmTypes { minerule::ATNone =0 , minerule::ATBFSWithGidsNoCross , minerule::ATBFSWithGidsAndCross , minerule::ATCare , minerule::ATConstrainedItemsets , minerule::ATEnd }

Functions

- const std::string & minerule::miningTaskToString (MiningTasks mt)
- const std::string & minerule::stringWithListOfMiningTasks ()
- const std::string & minerule::algorithmTypeToString (AlgorithmTypes t)
- AlgorithmTypes minerule::stringToAlgorithmType (const std::string &s)
- std::string minerule::stringWithListOfAlgorithmTypes ()

7.152 AlgorithmTypes.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __ALGORITHMTYPES_H__
00017 #define __ALGORITHMTYPES_H__
00018
00031 #include <string>
00032 #include "minerule/Utils/MineruleException.hpp"
00033
00034 namespace minerule {
00035
00036   typedef enum {
00037     MTMineRules=0,
00038     MTMineItemsets,
00039     MTMineSequences,
00040     MTEnd
00041   } MiningTasks;
```

```
00042
00043
00044   typedef enum {
00045     ATNone=0, // dummy algorithm type
00046     ATBFSWithGidsNoCross,
00047     ATBFSWithGidsAndCross,
00048     ATCare,
00049     ATConstrainedItemsets,
00050     ATEnd // dummy algorithm type do not add algorithms
00051           // below this element
00052   } AlgorithmTypes;
00053
00054   const std::string& miningTaskToString(MiningTasks mt);
00055   const std::string& stringWithListOfMiningTasks();
00056
00057   const std::string& algorithmTypeToString(AlgorithmTypes t);
00058
00059   AlgorithmTypes stringToAlgorithmType(const std::string& s) ;
00060
00061   std::string stringWithListOfAlgorithmTypes();
00062 }
00063
00064 #endif
```

7.153 /Users/esposito/Software/minerule/include/minerule/Utils/Bitstring.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string.h>
#include <stdio.h>
```

Data Structures

- class minerule::BitString

Namespaces

- namespace minerule

Macros

- #define CHAR_BIT 8

Typedefs

- typedef int minerule::boolean
- typedef unsigned int minerule::Bits

Variables

- const int minerule::DEBUG_LEVEL = 0
- const int minerule::DB_LEVEL_ALL = 1

7.153.1 Macro Definition Documentation

7.153.1.1 CHAR_BIT

```
#define CHAR_BIT 8
```

Definition at line 32 of file Bitstring.hpp.

7.154 Bitstring.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 /*****************************************************************
00017 Specification module : bitstring.h
00018 Code generated by Object Domain script
00019 Copyright 95
00020 *****************************************************************/
00021
00022
00023 #ifndef BITSTRING_H
00024 #define BITSTRING_H
00025
00026 #include <iostream>
00027 #include <vector>
00028 #include <string.h>
00029 #include <stdio.h>
00030
00031 #ifndef CHAR_BIT
00032 #define CHAR_BIT 8
00033 #endif
00034
00035 namespace minerule {
00036
00037         const int DEBUG_LEVEL = 0;
00038         const int DB_LEVEL_ALL = 1;
00039
00040
00041
00042         typedef int boolean;
00043         typedef unsigned int Bits;
00044
00045
00046
00047         class BitString {
00048                 std::vector<Bits> bits;
00049                 int n;
00050                 int Nw;
00051                 enum { Nb=CHAR_BIT * sizeof(Bits) };
00052                 void _Xran(int i);
00053                 void init(int nbits);
00054         public:
00055                 static int intersections;
00056
00057                 //Esistenza dell'elemento di posto  i
00058                 boolean test(int i) const {
00059                         if (i<0 || n <= i) return false;
00060                         return ((bits[i/Nb] & ((Bits)1 << i%Nb)) != 0);
00061                 }
```

```
00062
00063                 BitString();
00064                 BitString(int nbits);
00065                 BitString(const BitString& bs);
00066                 BitString&  set();
00067                 BitString&  set(int i,boolean value= true);
00068                 BitString&  reset();
00069                 BitString&  reset(int i);
00070                 BitString&  clear();
00071                 BitString&  clear(int i);
00072                 BitString&  invert();
00073                 BitString&  invert(int i);
00074                 BitString&  operator&=(const BitString& bs);
00075                 BitString&  operator|=(const BitString& bs);
00076                 BitString&  operator^=(const BitString& bs);
00077                 BitString&  operator=(const BitString& bs);
00078                 boolean operator==(const BitString& bs);
00079                 boolean operator!=(const BitString& bs);
00080
00081                 BitString  operator&(const BitString& bs1);
00082                 friend std::ostream&  operator«(std::ostream& out,const BitString& bs);
00083                 friend std::istream&  operator»(std::istream& in,BitString& bs);
00084
00085                 int count(boolean what = true) const ;
00086                 bool moreThan(double threshold) const ;
00087                 int length() const { return n; }
00088                 int ssize() const { return bits.size(); }
00089                 int size() const { return n; }
00090
00091                 void serialize(char* serialized,int* start);
00092                 void unserialize(char* serialized,int* start);
00093                 void print();
00094                 void print(std::ostream& out);
00095
00096
00097                 void setNBit(int num)            {  Nw = (num == 0) ? 0 : (num-1) / Nb;n=num; }
00098                 void* getElmA(int which)         {  return &(bits[which]); }
00099                 Bits getElm(int which)               {  return bits[which]; }
00100                 void insert(Bits element)     {  bits.push_back(element); }
00101                 std::vector <Bits>& gbits() {  return bits; }
00102
00103                 char operator[](int which) const { return (test(which) ? '1' : '0'); }
00104
00105         };
00106
00107 } //end namespace
00108
00109 #endif
```

7.155 /Users/esposito/Software/minerule/include/minerule/↩ Utils/common.hpp File Reference

```
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/mrdb/Types.hpp"
```

7.156 common.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
```

```
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __COMMONDEFINITIONS_H__
00017 #define __COMMONDEFINITIONS_H__
00018
00019 #include"minerule/mrdb/Connection.hpp"
00020 #include"minerule/mrdb/ResultSet.hpp"
00021 #include"minerule/mrdb/Statement.hpp"
00022 #include"minerule/mrdb/ResultSetMetaData.hpp>
00023 #include"minerule/mrdb/Types.hpp"
00024
00025 #endif
```

7.157 /Users/esposito/Software/minerule/include/minerule/Utils/Constraints.hpp File Reference

```
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Utils/Bitstring.hpp"
```

Data Structures

- class minerule::AggregateConstraint
- class minerule::AggregateMonoConstraint
- class minerule::AggregateAntiMonoConstraint
- class minerule::SumLessThan
- class minerule::SumLessEq
- class minerule::SumGreaterThan
- class minerule::SumGreaterEq

Namespaces

- namespace minerule

7.158 Constraints.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef _CONSTRAINTS_H
00017 #define _CONSTRAINTS_H
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019 #include "minerule/Database/MRResultSet.hpp"
```

```
00020 #include "minerule/Utils/Bitstring.hpp"
00021
00022 namespace minerule {
00023
00024 class BodyMap;
00025
00026 class AggregateConstraint {
00027   public:
00028         int attributeIndex;
00029         AggregateConstraint (int ai) : attributeIndex(ai) {}
00030         AggregateConstraint () : attributeIndex(0) {}
00031         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items) =0;
00032         virtual bool check (Transaction& items) =0;
00033         friend std::istream& operator»(std::istream& in, AggregateConstraint& i1) {
00034                 in » i1.attributeIndex;
00035                 return in;
00036         }
00037         friend std::ostream& operator«(std::ostream& out, const AggregateConstraint& i1) {
00038                 out « i1.attributeIndex;
00039                 return out;
00040         }
00041 };
00042
00043
00044 class AggregateMonoConstraint : public AggregateConstraint {
00045   public:
00046 //      AggregateMonoConstraint (int ai) : AggregateConstraints(ai) {}
00047 //      AggregateMonoConstraint () : AggregateConstraints(0) {}
00048         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items) =0;
00049         virtual bool check (Transaction& items) =0;
00050 };
00051
00052 class AggregateAntiMonoConstraint : public AggregateConstraint {
00053   public:
00054 //      AggregateAntiMonoConstraint (int ai) : AggregateConstraints(ai) {}
00055 //      AggregateAntiMonoConstraint () : AggregateConstraints(0) {}
00056         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items) =0;
00057         virtual bool check (Transaction& items) =0;
00058 };
00059
00060 class SumLessThan : public AggregateAntiMonoConstraint {
00061         int value;
00062   public:
00063         SumLessThan (int ai, int v) : value(v) { attributeIndex = ai;}
00064         SumLessThan (int v) : value(v)  {}
00065         SumLessThan () : value(0) {}
00066         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items);
00067         virtual bool check (Transaction& items);
00068 };
00069
00070 class SumLessEq : public AggregateAntiMonoConstraint {
00071         int value;
00072   public:
00073         SumLessEq (int ai, int v) : value(v) { attributeIndex = ai;}
00074         SumLessEq (int v) : value(v)  {}
00075         SumLessEq () : value(0) {}
00076         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items);
00077         virtual bool check (Transaction& items);
00078 };
00079
00080 class SumGreaterThan : public AggregateMonoConstraint {
00081         int value;
00082   public:
00083         SumGreaterThan (int ai, int v) : value(v) { attributeIndex = ai;}
00084         SumGreaterThan (int v) : value(v)  {}
00085         SumGreaterThan () : value(0) {}
00086         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items);
00087         virtual bool check (Transaction& items);
00088 };
00089
00090 class SumGreaterEq : public AggregateMonoConstraint {
00091         int value;
00092   public:
00093         SumGreaterEq (int ai, int v) : value(v) { attributeIndex = ai;}
00094         SumGreaterEq (int v) : value(v)  {}
00095         SumGreaterEq () : value(0) {}
00096         virtual bool check (BodyMap& itemMap, std::vector<ItemType>& items);
00097         virtual bool check (Transaction& items);
00098 };
00099
00100 } // end namespace
00101
00102 #endif
```

7.159 /Users/esposito/Software/minerule/include/minerule/Utils/Converter.hpp File Reference

```
#include <sstream>
#include <string>
#include <cerrno>
#include <string.h>
#include <stdlib.h>
#include "minerule/Utils/MineruleException.hpp"
```

Data Structures

- class minerule::Converter

Namespaces

- namespace minerule

7.160 Converter.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __CONVERSIONS_H__
00017 #define __CONVERSIONS_H__
00018
00019 #include <sstream>
00020 #include <string>
00021 #include <cerrno>
00022 #include <string.h>
00023 #include <stdlib.h>
00024 #include "minerule/Utils/MineruleException.hpp"
00025
00026 namespace minerule {
00027
00028 class Converter {
00029   typedef enum { NO_FLAG = 0, ALLOW_QUOTES_IN_NUMBERS = 1 } ConversionFlags;
00030   std::string value;
00031
00032 public:
00033   explicit Converter(const char *val) : value(val){};
00034   explicit Converter(std::string val) : value(val){};
00035   explicit Converter(double val) {
00036     std::stringstream ss;
00037     ss « val; // « std::ends;
00038     value = ss.str();
00039   }
00040
00041   explicit Converter(int val) {
00042     std::stringstream ss;
00043     ss « val;
00044     value = ss.str();
00045   }
```

```
00046
00047   explicit Converter(long val) {
00048     std::stringstream ss;
00049     ss « val;
00050     value = ss.str();
00051   }
00052
00053   explicit Converter(unsigned int val) {
00054     std::stringstream ss;
00055     ss « val; // « std::ends;
00056     value = ss.str();
00057   }
00058
00059   explicit Converter(unsigned long val) {
00060     std::stringstream ss;
00061     ss « val; // « std::ends;
00062     value = ss.str();
00063   }
00064
00065   explicit Converter(bool val) {
00066     if (val)
00067       value = "True";
00068     else
00069       value = "False";
00070   }
00071
00072   std::string toString() const { return value; }
00073
00074   double toDouble(unsigned int flags = ALLOW_QUOTES_IN_NUMBERS) const
00075       {
00076     char *endPtr = NULL;
00077     double result;
00078     const char *startChar;
00079     char endChar;
00080     errno = 0;
00081
00082     if ((flags & ALLOW_QUOTES_IN_NUMBERS) && value.length() > 1 &&
00083         value[0] == '\"' && value[value.length() - 1] == '\") {
00084       startChar = &value.c_str()[1];
00085       endChar = '\";
00086     } else {
00087       startChar = value.c_str();
00088       endChar = '\0';
00089     }
00090
00091     result = strtod(startChar, &endPtr);
00092     if (*endPtr != endChar || errno == ERANGE || errno == EINVAL)
00093       throw MineruleException(MR_ERROR_INTERNAL,
00094                               "Convertion error while converting string:" +
00095                                   value + " to a Double value");
00096     return result;
00097   }
00098
00099   long toLong(unsigned int flags = ALLOW_QUOTES_IN_NUMBERS) const {
00100     char *endPtr = NULL;
00101     errno = 0;
00102     long result;
00103     const char *startChar;
00104     char endChar;
00105
00106     if ((flags & ALLOW_QUOTES_IN_NUMBERS) && value.length() > 1 &&
00107         value[0] == '\"' && value[value.length() - 1] == '\") {
00108       startChar = &value.c_str()[1];
00109       endChar = '\";
00110     } else {
00111       startChar = value.c_str();
00112       endChar = '\0';
00113     }
00114
00115     result = strtol(startChar, &endPtr, 10);
00116     if (*endPtr != endChar || errno == ERANGE || errno == EINVAL) {
00117       std::stringstream ss;
00118       ss « "Convertion error while converting string:" « value
00119          « " to a long value. lenght:" « value.length()
00120          « " strlen:" « strlen(value.c_str());
00121       if (errno != 0)
00122         ss « " errno string:" « strerror(errno);
00123
00124       throw MineruleException(MR_ERROR_INTERNAL, ss.str());
00125     }
00126
00127     return result;
00128   }
00129
00130   bool toBool() const {
00131     if (value == "True")
00132       return true;
```

```
00133     else if (value == "False")
00134       return false;
00135     else if (isNumber()) {
00136       return toDouble() > 0;
00137     } else
00138       throw MineruleException(MR_ERROR_INTERNAL,
00139                               "Converting error while converting string:"
00140                               "'" +
00141                                   value + "' to a bool value");
00142   }
00143
00144   bool isNumber() const {
00145     try {
00146       toDouble();
00147     } catch (MineruleException &e) {
00148       return false;
00149     }
00150
00151     return true;
00152   }
00153 };
00154 }
00155
00156 #endif
```

7.161 /Users/esposito/Software/minerule/include/minerule/Utils/FileUtils.hpp File Reference

```
#include <string>
```

Data Structures

- class minerule::FileUtils

Namespaces

- namespace minerule

7.162 FileUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef FILEUTILS_H_MLEUV1SZ
00017 #define FILEUTILS_H_MLEUV1SZ
00018
00019 #include <string>
00020
00021 namespace minerule {
00022         class FileUtils {
00023         public:
00024                 static bool fileExists(const std::string& filename);
00025         };
00026 }
00027
00028 #endif /* end of include guard: FILEUTILS_H_MLEUV1SZ */
```

7.163 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleErrors.hpp File Reference

Namespaces

- namespace minerule

Enumerations

- enum minerule::MineruleErrors {
  minerule::MR_ERROR_NO_ERROR =0 , minerule::MR_ERROR_UNKNOWN , minerule::MR_ERROR_INTERNAL , minerule::MR_ERROR_INPUT_FILE_NOT_FOUND , minerule::MR_ERROR_OUTPUT_FILE_PROBLEM , minerule::MR_ERROR_NO_MINERULE_SPECIFIED , minerule::MR_ERROR_NO_OPTIONFILE_SPECIFIED , minerule::MR_ERROR_OPTION_CONFIGURATION , minerule::MR_ERROR_MINERULE_ALREADY_EXISTS , minerule::MR_ERROR_DATABASE_ERROR , minerule::MR_ERROR_OPTION_PARSING , minerule::MR_ERROR_MINERULETEXT_PARSING , minerule::MR_ERROR_CATALOGUE_ERROR , minerule::MR_ERROR_OPTIMIZER_ERROR , minerule::MR_ERROR_INSTA , minerule::MR_ERROR_SAFETY_PROBLEM }

Functions

- const char * minerule::me_error_name (MineruleErrors me)
- int minerule::me_error_begin ()
- int minerule::me_error_end ()

7.164 MineruleErrors.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MINERULEERRORS_H__
00017 #define __MINERULEERRORS_H__
00018
00019 namespace minerule {
00020   typedef enum {
00021     MR_ERROR_NO_ERROR=0,
00022     MR_ERROR_UNKNOWN,
00023     MR_ERROR_INTERNAL,
00024     MR_ERROR_INPUT_FILE_NOT_FOUND,
00025     MR_ERROR_OUTPUT_FILE_PROBLEM,
00026     MR_ERROR_NO_MINERULE_SPECIFIED,
00027     MR_ERROR_NO_OPTIONFILE_SPECIFIED,
00028     MR_ERROR_OPTION_CONFIGURATION,
00029     MR_ERROR_MINERULE_ALREADY_EXISTS,
00030     MR_ERROR_DATABASE_ERROR,
00031     MR_ERROR_OPTION_PARSING,
00032     MR_ERROR_MINERULETEXT_PARSING,
00033     MR_ERROR_CATALOGUE_ERROR,
00034     MR_ERROR_OPTIMIZER_ERROR,
```

```
00035     MR_ERROR_INSTALLATION_PROBLEM,
00036     MR_ERROR_SAFETY_PROBLEM,
00037   } MineruleErrors;
00038
00039   const char* me_error_name(MineruleErrors me);
00040   int me_error_begin();
00041   int me_error_end();
00042 }
00043
00044
00045
00046
00047
00048 #endif
00049
```

7.165 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleException.hpp File Reference

```
#include <exception>
#include <sstream>
#include <string>
#include "minerule/Utils/MineruleErrors.hpp"
```

Data Structures

- class minerule::MineruleException

Namespaces

- namespace minerule

Macros

- #define _NOEXCEPT throw()
- #define MineruleException(a, b) MineruleException(__FILE__, __LINE__, a,b)

7.165.1 Macro Definition Documentation

7.165.1.1 _NOEXCEPT

```
#define _NOEXCEPT throw()
```

Definition at line 25 of file MineruleException.hpp.

7.165.1.2 MineruleException

```
#define MineruleException(
              a,
              b ) MineruleException( __FILE__, __LINE__, a,b )
```

Definition at line 55 of file MineruleException.hpp.

7.166 MineruleException.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MINERULE_EXCEPTION_H__
00017 #define __MINERULE_EXCEPTION_H__
00018
00019 #include <exception>
00020 #include <sstream>
00021 #include <string>
00022 #include "minerule/Utils/MineruleErrors.hpp"
00023
00024 #ifndef _NOEXCEPT
00025 #define _NOEXCEPT throw()
00026 #endif
00027
00028 namespace minerule {
00029
00030 class MineruleException : public std::exception {
00031   std::string message;
00032         std::string formattedMessage;
00033
00034         std::string file;
00035         int line;
00036   size_t errorCode;
00037
00038         void formatMessage() _NOEXCEPT;
00039  public:
00040   MineruleException(std::string sourceFile, int sourceLine,  size_t errCode, std::string msg);
00041
00042   virtual ~MineruleException() _NOEXCEPT {
00043   }
00044
00045   virtual const char* what() const _NOEXCEPT ;
00046         virtual const char* unformattedMessage() const _NOEXCEPT ;
00047
00048   virtual size_t getErrorCode() const _NOEXCEPT {
00049     return errorCode;
00050   }
00051 };
00052
00053 }
00054
00055 #define MineruleException(a,b) MineruleException( __FILE__, __LINE__, a,b )
00056
00057 #endif
```

7.167 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleLogs.hpp File Reference

```
#include <string>
#include <iostream>
```

Data Structures

- class minerule::MRLogPusher
- class minerule::MRErrPusher
- class minerule::MRWarnPusher
- class minerule::MRDebugPusher

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::MRLog ()
- std::ostream & minerule::MRErr ()
- std::ostream & minerule::MRWarn ()
- std::ostream & minerule::MRDebug ()
- void minerule::MRLog (const std::string &msg)
- void minerule::MRErr (const std::string &msg)
- void minerule::MRWarn (const std::string &msg)
- void minerule::MRDebug (const std::string &msg)
- void minerule::MRLogPush (const std::string &descr)
- void minerule::MRLogPop ()
- void minerule::MRErrPush (const std::string &descr)
- void minerule::MRErrPop ()
- void minerule::MRWarnPush (const std::string &descr)
- void minerule::MRWarnPop ()
- void minerule::MRDebugPush (const std::string &descr)
- void minerule::MRDebugPop ()
- void minerule::MRLogStartMeasuring (const std::string &description)
- void minerule::MRLogStopMeasuring (const std::string &description)
- void minerule::MRLogShowMeasurements ()
- int minerule::MRLogGetNewID ()

7.168 MineruleLogs.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MINERULE_LOGS_H__
00017 #define __MINERULE_LOGS_H__
00018
00019 #include <string>
00020 #include <iostream>
00021
00022 namespace minerule {
```

```
00023   /* LOGGING FACILITIES */
00024   /* These functions returns the shared streams contained in the
00025    * object returned by sharedOptions() (see classes below)
00026    * They are the preferred way of accessing such streams since
00027    * they provide time and memory logging (they output the informations
00028    * on the required stream before returning it).
00029    */
00030
00031   std::ostream& MRLog();
00032   std::ostream& MRErr();
00033   std::ostream& MRWarn();
00034   std::ostream& MRDebug();
00035
00036   // The following methods print msg on the log and ensure that
00037   // the whole thing stays in 80 characters.
00038   void MRLog(const std::string& msg);
00039   void MRErr(const std::string& msg);
00040   void MRWarn(const std::string& msg);
00041   void MRDebug(const std::string& msg);
00042
00043   void MRLogPush(const std::string& descr);
00044   void MRLogPop();
00045   void MRErrPush(const std::string& descr);
00046   void MRErrPop();
00047   void MRWarnPush(const std::string& descr);
00048   void MRWarnPop();
00049   void MRDebugPush(const std::string& descr);
00050   void MRDebugPop();
00051
00052
00053   void MRLogStartMeasuring(const std::string& description);
00054   void MRLogStopMeasuring(const std::string& description);
00055   void MRLogShowMeasurements();
00056
00063   class MRLogPusher {
00064   public:
00065     MRLogPusher(const std::string& str) {
00066       MRLogPush(str);
00067     }
00068
00069     ~MRLogPusher() {
00070       MRLogPop();
00071     }
00072   };
00073
00081   class MRErrPusher {
00082   public:
00083     MRErrPusher(const std::string& str) {
00084       MRErrPush(str);
00085     }
00086
00087     ~MRErrPusher() {
00088       MRErrPop();
00089     }
00090   };
00091
00097   class MRWarnPusher {
00098   public:
00099     MRWarnPusher(const std::string& str) {
00100       MRWarnPush(str);
00101     }
00102
00103     ~MRWarnPusher() {
00104       MRWarnPop();
00105     }
00106   };
00107
00108
00114   class MRDebugPusher {
00115   public:
00116     MRDebugPusher(const std::string& str) {
00117       MRDebugPush(str);
00118     }
00119
00120     ~MRDebugPusher() {
00121       MRDebugPop();
00122     }
00123   };
00124
00125   int MRLogGetNewID();
00126 }
00127
00128
00129 #endif
```

7.169 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions.hpp File Reference

```
#include <string>
#include <string.h>
#include "minerule/mrdb/Connection.hpp"
#include <map>
#include <iostream>
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/MinMaxPair.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Utils/MRLogger.hpp"
#include "minerule/Utils/AlgorithmTypes.hpp"
#include "minerule/Utils/StringUtils.hpp"
#include "minerule/Utils/MineruleOptions_implementations/root.hpp"
```

Data Structures

- class minerule::OptionBase

Namespaces

- namespace minerule

7.170 MineruleOptions.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017 #ifndef __MINERULE_OPTIONS_H__
00018 #define __MINERULE_OPTIONS_H__
00019
00020 #include <string>
00021 #include <string.h>
00022 #include "minerule/mrdb/Connection.hpp"
00023 #include <map>
00024 #include <iostream>
00025
00026 #include "minerule/Utils/MineruleLogs.hpp"
00027
00028
00029 #include "minerule/Utils/MineruleException.hpp"
00030 #include "minerule/Utils/MinMaxPair.hpp"
00031 #include "minerule/Utils/Converter.hpp"
00032 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00033 #include "minerule/Utils/MRLogger.hpp"
```

```
00034 #include "minerule/Utils/AlgorithmTypes.hpp"
00035 #include "minerule/Utils/StringUtils.hpp"
00036
00037 namespace minerule {
00038
00039   class OptionBase {
00040   public:
00041     virtual ~OptionBase() {}
00042
00043     virtual void setOption(const std::string& name,const std::string& value)
00044       =0;
00045
00046     virtual OptionBase& subclassForName(const std::string& subclassName)
00047       {
00048       throw MineruleException(MR_ERROR_OPTION_PARSING,
00049                               className()+" does not support sub class named:"+
00050                               subclassName);
00051     }
00052
00053     virtual std::string className() const=0;
00054
00055   };
00056
00057 // Options subclasses
00058 #include "minerule/Utils/MineruleOptions_implementations/root.hpp"
00059
00060 }// end namespace minerule
00061
00062 #endif
```

7.171 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/miningalgorithms.hpp File Reference

```
#include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.↩
hpp"
#include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/itemsetmining.
hpp"
#include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/sequencemining
hpp"
```

Data Structures

- class MiningAlgorithms

7.172 miningalgorithms.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class MiningAlgorithms : public OptionBase {
00017 public:
```

```
00018   /* ========= RULE MINING ALGORITHMS ======== */
00019         #include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/rulemining.hpp"
00020   /* ========= ITEMSET MINING ALGORITHMS ======== */
00021         #include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/itemsetmining.hpp"
00022   /* ========= SEQUENCE MINING ALGORITHMS ======= */
00023         #include "minerule/Utils/MineruleOptions_implementations/miningalgorithms/sequencemining.hpp"
00024
00025 private:
00026   RulesMiningAlgorithms rulesMiningAlgorithms;
00027   ItemsetsMiningAlgorithms itemsetsMiningAlgorithms;
00028   SequencesMiningAlgorithms sequencesMiningAlgorithms;
00029
00030 public:
00031
00032   virtual ~MiningAlgorithms() { };
00033
00034   virtual std::string className() const {
00035         return "MiningAlgorithm";
00036   }
00037
00038   virtual void setOption(const std::string& name,const std::string& value) {
00039         throw MineruleException( MR_ERROR_OPTION_PARSING, "MiningAlgorithms option class does not
      support option:"+value );
00040   }
00041
00042   virtual OptionBase& subclassForName(const std::string& subclassName) {
00043         if( subclassName=="rulesmining" ) {
00044           return rulesMiningAlgorithms;
00045         }  else if(subclassName=="itemsetsmining") {
00046           return itemsetsMiningAlgorithms;
00047         }  else if(subclassName=="sequencesmining") {
00048           return sequencesMiningAlgorithms;
00049           } else {
00050           return OptionBase::subclassForName(subclassName);
00051         }
00052   }
00053
00054
00055   RulesMiningAlgorithms& getRulesMiningAlgorithms() { return rulesMiningAlgorithms; }
00056   const RulesMiningAlgorithms& getRulesMiningAlgorithms() const { return rulesMiningAlgorithms; }
00057   const ItemsetsMiningAlgorithms& getItemsetsMiningAlgorithms() const { return
      itemsetsMiningAlgorithms; }
00058   const   SequencesMiningAlgorithms& getSequencesMiningAlgorithms() const { return
      sequencesMiningAlgorithms; }
00059
00060 };
```

7.173 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_↩ implementations/miningalgorithms/itemsetmining.hpp File Reference

Data Structures

- class ItemsetsMiningAlgorithms

7.174 itemsetmining.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class ItemsetsMiningAlgorithms : public OptionBase {
00017 public:
00018
00019         ItemsetsMiningAlgorithms() : preferredAlgorithm(ATNone) {  }
00020
00021         ItemsetsMiningAlgorithms(const ItemsetsMiningAlgorithms& rhs) :
00022           preferredAlgorithm(rhs.preferredAlgorithm) {};
00023
00024         virtual ~ItemsetsMiningAlgorithms() {}
00025
00026         void setPreferredAlgorithm(AlgorithmTypes type) {
00027           preferredAlgorithm = type;
00028         }
00029
00030         AlgorithmTypes getPreferredAlgorithm() const {
00031           return preferredAlgorithm;
00032         }
00033
00034         virtual std::string className() const {
00035           return "itemsetsmining";
00036         }
00037
00038
00039         virtual OptionBase& subclassForName(const std::string& subclassName)
00040           {
00041             return OptionBase::subclassForName(subclassName);
00042         }
00043
00044         virtual void setOption(const std::string& name,const std::string& value)
00045           ;
00046
00047
00048 private:
00049         AlgorithmTypes preferredAlgorithm;
00050 }; // class ItemSetMiningAlgorithms
```

7.175 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_↩ implementations/miningalgorithms/rulemining.hpp File Reference

Data Structures

- class RulesMiningAlgorithms
- class RulesMiningAlgorithms::PartitionBase
- class RulesMiningAlgorithms::PartitionWithClusters
- class RulesMiningAlgorithms::FPGrowth

7.176 rulemining.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class RulesMiningAlgorithms : public OptionBase {
```

```
00017 public:
00018         class PartitionBase : public OptionBase {
00019           unsigned int rowsPerPartition;
00020         public:
00021           virtual ~PartitionBase() {};
00022
00023           virtual std::string className() const {
00024             return "partitionbase";
00025           }
00026
00027           virtual void setOption(const std::string& name,const std::string& value)
00028             ;
00029           unsigned int getRowsPerPartition() const {
00030             return rowsPerPartition;
00031           }
00032           void setRowsPerPartition(unsigned int num) {
00033             rowsPerPartition = num;
00034           }
00035         };
00036
00037         class PartitionWithClusters : public OptionBase {
00038           unsigned int rowsPerPartition;
00039         public:
00040           virtual ~PartitionWithClusters() {};
00041
00042           virtual std::string className() const {
00043             return "partitionwithclusters";
00044           }
00045
00046           virtual void setOption(const std::string& name,const std::string& value)
00047             ;
00048           unsigned int getRowsPerPartition() const {
00049             return rowsPerPartition;
00050           }
00051           void setRowsPerPartition(unsigned int num) {
00052             rowsPerPartition = num;
00053           }
00054
00055         };
00056
00057         class FPGrowth : public OptionBase {
00058         public:
00059           typedef enum {
00060             Original,
00061             SingleReorder
00062           } FPAlgoType;
00063         private:
00064
00065           FPAlgoType algoType;
00066         public:
00067           virtual ~FPGrowth() {};
00068           virtual std::string className() const {
00069             return "fpgrowth";
00070           }
00071
00072           virtual void setOption(const std::string& name,const std::string& value)
00073             ;
00074
00075           FPAlgoType getAlgoType() const {
00076             return algoType;
00077           }
00078           void setAlgoType(FPAlgoType type) {
00079             algoType = type;
00080           }
00081         };
00082
00083         // --- MiningAlgorithms methods and variables
00084
00085         PartitionBase partitionbase;
00086         PartitionWithClusters partitiongeneralized;
00087         FPGrowth fpgrowth;
00088
00089         AlgorithmTypes preferredAlgorithm;
00090
00091         PartitionBase&
00092           getPartitionBase()   {
00093           return partitionbase;
00094         }
00095
00096         const PartitionBase&
00097           getPartitionBase() const  {
00098           return partitionbase;
00099         }
00100
00101         PartitionWithClusters&
00102           getPartitionWithClusters()   {
00103           return partitiongeneralized;
```

```
00104         }
00105
00106         const PartitionWithClusters&
00107           getPartitionWithClusters() const  {
00108           return partitiongeneralized;
00109         }
00110
00111         FPGrowth&
00112           getFPGrowth()  {
00113           return fpgrowth;
00114         }
00115
00116         const FPGrowth&
00117           getFPGrowth() const {
00118           return fpgrowth;
00119         }
00120
00121         virtual void setOption(const std::string& name,const std::string& value)
00122           ;
00123
00124         virtual std::string className() const {
00125           return "rulesmining";
00126         }
00127
00128         virtual OptionBase& subclassForName(const std::string& subclassName)
00129           {
00130           if( subclassName=="partitionbase" ) {
00131             return partitionbase;
00132           } else if(subclassName=="partitionwithclusters") {
00133             return partitiongeneralized;
00134           } else if(subclassName=="fpgrowth") {
00135             return fpgrowth;
00136           } else {
00137             return OptionBase::subclassForName(subclassName);
00138           }
00139         }
00140
00141
00142
00143         RulesMiningAlgorithms() : preferredAlgorithm(ATNone) {  }
00144
00145         RulesMiningAlgorithms(const RulesMiningAlgorithms& rhs) :
00146           partitionbase(rhs.partitionbase),
00147           partitiongeneralized(rhs.partitiongeneralized),
00148           fpgrowth(rhs.fpgrowth),
00149           preferredAlgorithm(rhs.preferredAlgorithm) {};
00150
00151         virtual ~RulesMiningAlgorithms() {}
00152
00153         void setPreferredAlgorithm(AlgorithmTypes type) {
00154           preferredAlgorithm = type;
00155         }
00156
00157         AlgorithmTypes getPreferredAlgorithm() const {
00158           return preferredAlgorithm;
00159         }
00160 }; /* End of RuleMiningAlgorithms options */
00161
```

7.177 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_↩ implementations/miningalgorithms/sequencemining.hpp File Reference

Data Structures

- class SequencesMiningAlgorithms

7.178 sequencemining.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class SequencesMiningAlgorithms : public OptionBase {
00017 public:
00018
00019         SequencesMiningAlgorithms() : preferredAlgorithm(ATNone) {  }
00020
00021         SequencesMiningAlgorithms(const SequencesMiningAlgorithms& rhs) :
00022           preferredAlgorithm(rhs.preferredAlgorithm) {}
00023
00024         virtual ~SequencesMiningAlgorithms() {}
00025
00026         void setPreferredAlgorithm(AlgorithmTypes type) {
00027           preferredAlgorithm = type;
00028         }
00029
00030         AlgorithmTypes getPreferredAlgorithm() const {
00031           return preferredAlgorithm;
00032         }
00033
00034         virtual std::string className() const {
00035           return "sequencesmining";
00036         }
00037
00038
00039         virtual OptionBase& subclassForName(const std::string& subclassName)
00040           {
00041             return OptionBase::subclassForName(subclassName);
00042         }
00043
00044         virtual void setOption(const std::string& name,const std::string& value)
00045           ;
00046
00047
00048 private:
00049         AlgorithmTypes preferredAlgorithm;
00050 }; // class SequenceMiningAlgorithms
```

7.179 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/mrdb.hpp File Reference

Data Structures

- class Mrdb

7.180 mrdb.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
```

```
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017 class Mrdb : public OptionBase {
00018   std::string name;
00019   std::string username;
00020   std::string password;
00021   mrdb::Connection *connection;
00022   bool cacheWrites;
00023   std::string dbms;
00024
00025 public:
00026   Mrdb() : connection(NULL), cacheWrites(false), dbms("postgres"){};
00027
00028   virtual ~Mrdb() {}
00029
00030   virtual std::string className() const { return "mrdb"; }
00031
00032   virtual void setOption(const std::string &name,
00033                          const std::string &value) ;
00034
00035   std::string getName() const { return name; }
00036
00037   std::string getUsername() const { return username; }
00038   std::string getPassword() const { return password; }
00039
00040   std::string getDBMS() const { return dbms; }
00041
00042   bool getCacheWrites() const { return cacheWrites; }
00043
00044   // at least for the sharedOptions, this
00045   // object is set by init()
00046   mrdb::Connection *getMRDBConnection() const {
00047     if (connection == NULL) {
00048       throw MineruleException(
00049           MR_ERROR_DATABASE_ERROR,
00050           (std::string) "Check out the parameters, it looks like that"
00051                         " you did not specified any mrdb data source");
00052     }
00053
00054     return connection;
00055   }
00056
00057   void setName(const std::string &str) { name = str; }
00058
00059   void setUsername(const std::string &str) { username = str; }
00060
00061   void setPassword(const std::string &str) { password = str; }
00062
00063   void setCacheWrites(bool v) { cacheWrites = v; }
00064
00065   void setDBMS(std::string v) { dbms = v; }
00066
00067   void setConnection(mrdb::Connection *con) { connection = con; }
00068
00069   void clearConnection() {
00070     if (connection != NULL) {
00071       delete connection;
00072       connection = NULL;
00073     }
00074   }
00075
00076   std::string getLibName(const std::string& dbms)  {
00077     if(dbms=="postgres") {
00078       return "libmrdb_pg.so";
00079     } else {
00080       throw MineruleException(MR_ERROR_DATABASE_ERROR,
00081       "Database " + dbms + " not supported yet.");
00082     }
00083   }
00084
00085   // if necessary, free the current connection (using delete) and
00086   // create a new one using the currente parameters
00087   void resetConnection() {
00088     clearConnection();
00089
00090     std::string libName(getLibName(getDBMS()));
00091     static void *handle = dlopen( libName.c_str(), RTLD_LAZY);
00092
00093     if (handle == NULL) {
00094       throw MineruleException(MR_ERROR_DATABASE_ERROR, dlerror());
00095     }
00096
00097     MRDBConnectFun connect = (MRDBConnectFun)dlsym(handle, "connect");
00098
00099     if (connect == NULL) {
```

```
00100       throw MineruleException(MR_ERROR_DATABASE_ERROR, dlerror());
00101     }
00102
00103     setConnection(connect(getName().c_str(), getUsername().c_str(),
00104                           getPassword().c_str()));
00105   }
00106 };
```

7.181 /Users/esposito/Software/minerule/include/minerule/Utils/MineruleOptions_implementations/optimizations.hpp File Reference

Data Structures

- class Optimizations
- class Optimizations::Combinator

7.182 optimizations.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class Optimizations: public OptionBase {
00017 public:
00018   typedef enum  {
00019         ConstructiveAlgo,
00020         DestructiveAlgo,
00021         AutochooseIncrAlgo
00022   } PreferredIncrementalAlgorithm;
00023
00024   class Combinator: public OptionBase {
00025   private:
00026         float  timeOut;
00027         size_t maxDisjuncts;
00028         size_t maxQueries;
00029         size_t maxDistinctPredicates;
00030   public:
00031         Combinator() :
00032           timeOut(4.0),
00033           maxDisjuncts(3),
00034           maxQueries(5),
00035           maxDistinctPredicates(10) {}
00036         virtual ~Combinator() {};
00037
00038
00039         virtual std::string className() const {
00040           return "combinator";
00041         }
00042
00043         virtual void setOption(const std::string& name, const std::string& value)
00044           ;
00045
00046         float getTimeOutThreshold() const { return timeOut; }
00047         size_t getMaxDisjuncts() const {return maxDisjuncts;}
00048         size_t getMaxQueries() const {return maxQueries;}
00049         size_t getMaxDistinctPredicates() const {return maxDistinctPredicates;}
00050
00051         void setMaxDisjuncts(size_t m) { maxDisjuncts=m; }
```

```
00052         void setMaxQueries(size_t m) { maxQueries=m; }
00053         void setMaxDistinctPredicates(size_t m) { maxDistinctPredicates=m; }
00054         void setTimeOutThreshold(float to) { timeOut=to; }
00055   };
00056
00057 private:
00058   bool tryOptimizations;
00059   OptimizerCatalogue* catalogue;
00060   PreferredIncrementalAlgorithm preferredIncrementalAlgorithm;
00061   bool avoidDominanceDetection;
00062   bool avoidEquivalenceDetection;
00063   bool avoidCombinationDetection;
00064   Combinator combinator;
00065 public:
00066   Optimizations() :
00067         catalogue(NULL),
00068         preferredIncrementalAlgorithm(AutochooseIncrAlgo),
00069         avoidDominanceDetection(false),
00070         avoidEquivalenceDetection(false),
00071         avoidCombinationDetection(false) {};
00072
00073   virtual ~Optimizations() {
00074         if( catalogue!=NULL )
00075           delete catalogue;
00076   };
00077
00078   virtual std::string className() const {
00079         return "optimizations";
00080   }
00081
00082   virtual void setOption(const std::string& name,const std::string& value)
00083         ;
00084
00085   virtual OptionBase& subclassForName(const std::string& subclassName)
00086   {
00087         if( subclassName=="combinator" ) {
00088           return combinator;
00089         } else {
00090           return OptionBase::subclassForName(subclassName);
00091         }
00092   }
00093
00094   Combinator& getCombinator() { return combinator; }
00095   const Combinator& getCombinator() const { return combinator; }
00096
00097   bool getTryOptimizations() const {
00098         return tryOptimizations;
00099   }
00100
00101   void setTryOptimizations(bool val) {
00102         tryOptimizations=val;
00103   }
00104
00105   bool getAvoidEquivalenceDetection() const {
00106         return avoidEquivalenceDetection;
00107   }
00108
00109   void setAvoidEquivalenceDetection(bool val) {
00110         avoidEquivalenceDetection=val;
00111   }
00112
00113   bool getAvoidCombinationDetection() const {
00114         return avoidCombinationDetection;
00115   }
00116
00117   void setAvoidCombinationDetection(bool val) {
00118         avoidCombinationDetection=val;
00119   }
00120
00121   bool getAvoidDominanceDetection() const {
00122         return avoidDominanceDetection;
00123   }
00124
00125   void setAvoidDominanceDetection(bool val) {
00126         avoidDominanceDetection=val;
00127   }
00128
00129   PreferredIncrementalAlgorithm
00130   getIncrementalAlgorithm() const {
00131         return preferredIncrementalAlgorithm;
00132   }
00133
00134   void setIncrementalAlgorithm(PreferredIncrementalAlgorithm p) {
00135         preferredIncrementalAlgorithm=p;
00136   }
00137
00138   OptimizerCatalogue& getCatalogue() {
```

```
00139         if( catalogue==NULL ) {
00140           catalogue = new OptimizerCatalogue();
00141         }
00142 
00143         return *catalogue;
00144   }
00145 };
```

7.183 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/outstream.hpp File Reference

Data Structures

- class OutStream

7.184 outstream.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class OutStream : public OptionBase {
00017 private:
00018   MRLogger* logger;
00019   size_t logLevel;
00020   // reference to the enclosing option object
00021   MineruleOptions* options;
00022   std::string streamName;
00023 public:
00024   OutStream(MineruleOptions* opt, const std::string& name)
00025         : logger(NULL), options(opt), streamName(name) {};
00026   virtual ~OutStream() {};
00027 
00028   virtual std::string className() const {
00029         return streamName;
00030   }
00031 
00032   virtual void setOption(const std::string& name,const std::string& value)
00033         ;
00034 
00035   void setLogger(MRLogger& log) {
00036         logger=&log;
00037         logLevel = log.getLogLevel();
00038   }
00039 
00040   MRLogger& getLogger() {
00041         return *logger;
00042   }
00043   std::ostream& getStream() const {
00044         assert(logger!=NULL);
00045         return logger->log();
00046   }
00047 
00048   void setLogLevel(size_t level) {
00049         assert(logger!=NULL);
00050         logger->setLogLevel(level);
00051         logLevel=level;
00052   }
00053 
00054   void disable() {
00055         logger->setLogLevel(0);
```

```
00056   }
00057
00058   void enable() {
00059         logger->setLogLevel(logLevel);
00060   }
00061 };
```

7.185 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/parsers.hpp File Reference

Data Structures

- class Parsers

7.186 parsers.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class Parsers : public OptionBase {
00017   FILE* logfile;
00018   MinMaxPair bodyCardinalities;
00019   MinMaxPair headCardinalities;
00020
00021   void clearStream() {
00022       if(logfile!=NULL && logfile!=stdout && logfile!=stderr) {
00023         fclose(logfile);
00024       }
00025       logfile=NULL;
00026   }
00027  public:
00028   Parsers() : logfile(NULL),
00029       bodyCardinalities(MinMaxPair(1,1000)),
00030       headCardinalities(MinMaxPair(1,1000)) {};
00031
00032   virtual ~Parsers() {
00033       clearStream();
00034   }
00035
00036   virtual std::string className() const {
00037       return "parsers";
00038   }
00039
00040   virtual void setOption(const std::string& name, const std::string& value)
00041       ;
00042
00043   void setLogFILE(const std::string& fname) ;
00044   void setLogOnStdout();
00045   void setLogOnStderr();
00046
00047   const FILE* getLogFILE() const {
00048       assert( logfile!=NULL );
00049       return logfile;
00050   }
00051
00052   void setMinBodyElems(int m) {
00053       bodyCardinalities.setMin(m);
00054   }
00055
00056   void setMaxBodyElems(int M) {
```

```
00057         bodyCardinalities.setMax(M);
00058     }
00059
00060     void setMinHeadElems(int m) {
00061         headCardinalities.setMin(m);
00062     }
00063
00064     void setMaxHeadElems(int M) {
00065         headCardinalities.setMax(M);
00066     }
00067
00068     const MinMaxPair& getBodyCardinalities() const {
00069         return bodyCardinalities;
00070     }
00071
00072     const MinMaxPair& getHeadCardinalities() const {
00073         return headCardinalities;
00074     }
00075 };
```

7.187 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/root.hpp File Reference

```
#include <dlfcn.h>
#include "minerule/Utils/MineruleOptions_implementations/mrdb.hpp"
#include "minerule/Utils/MineruleOptions_implementations/safety.hpp"
#include "minerule/Utils/MineruleOptions_implementations/miningalgorithms.↩
hpp"
#include "minerule/Utils/MineruleOptions_implementations/optimizations.hpp"
#include "minerule/Utils/MineruleOptions_implementations/parsers.hpp"
#include "minerule/Utils/MineruleOptions_implementations/outstream.hpp"
```

Data Structures

- class MineruleOptions

7.188 root.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017 #include <dlfcn.h>
00018
00019
00020 class MineruleOptions : public OptionBase {
00021 public:
00022         static const std::string DEFAULT_FILE_NAME;
00023
00024   /* ============================
00025    * Inner classes
00026    * ============================ */
```

```
00027
00028         /* ========== MRDB_DB ========== */
00029
00030         #include "minerule/Utils/MineruleOptions_implementations/mrdb.hpp"
00031   /* ========== SAFETY ========== */
00032
00033         #include "minerule/Utils/MineruleOptions_implementations/safety.hpp"
00034   /* ========== MININGALGORITHMS ========== */
00035
00036
00037         #include "minerule/Utils/MineruleOptions_implementations/miningalgorithms.hpp"
00038
00039   /* ========== OPTIMIZATIONS ========== */
00040         #include "minerule/Utils/MineruleOptions_implementations/optimizations.hpp"
00041
00042
00043   /* ========== PARSERS ========== */
00044
00045         #include "minerule/Utils/MineruleOptions_implementations/parsers.hpp"
00046
00047   /* ========== OUTSTREAM ========== */
00048
00049         #include "minerule/Utils/MineruleOptions_implementations/outstream.hpp"
00050
00051
00052   /* ============================
00053    * Private members
00054    * ============================ */
00055
00056 private:
00057   static MineruleOptions sharedOptions;
00058   Mrdb mrdb_db;
00059   Safety safety;
00060   Optimizations optimizations;
00061   MiningAlgorithms miningAlgorithms;
00062   OutStream logStream;
00063   OutStream errStream;
00064   OutStream warnStream;
00065   OutStream debugStream;
00066   Parsers parsers;
00067   std::map<std::string,std::string> userOptions;
00068   std::string mineruleSourceName; // file name of the file which contained the
00069                                   // minerule text
00070   std::string mineruleName; // name of the minerule as it appear in the
00071                             // minerule statement
00072
00073   // the following mapping is used to allow streams to share a common
00074   // std::ostream. If, for instance, the user says that both log and err
00075   // stream should be redirected onto file "foo", then we will create
00076   // an std::ostream on "foo" once and use for both the streams. Obviously
00077   // this means that we need to keep track of already "known" streams.
00078   std::map<std::string, MRLogger* > knownStreams;
00079
00080
00081   // This boolean is set when an option object
00082   // is ready to provide the informations for
00083   // which it was created
00084   bool ready;
00085
00086   bool isReady() {
00087     return ready;
00088   }
00089
00094   void init();
00095
00096   static long stringToLong(const std::string& value, const std::string& name)
00097     {
00098     Converter c(value);
00099     long result;
00100     try {
00101         result = c.toLong();
00102     } catch(MineruleException&) {
00103         throw MineruleException(MR_ERROR_OPTION_PARSING,
00104                                 "Error while parsing option '"+name+"':"
00105                                 + " an integer was expected, but " + value
00106                                 + " found");
00107     }
00108
00109     return result;
00110   }
00111
00112 protected:
00113   virtual ~MineruleOptions();
00114    // till now there is no need to declare any
00115   // new MineruleOptions, nor to create copy of
00116   // them...
00117   MineruleOptions() :
```

```
00118     logStream(this,"logstream"),
00119     errStream(this,"errstream"),
00120     warnStream(this,"warnstream"),
00121     debugStream(this,"debugstream") {  }
00122
00123   MineruleOptions(const MineruleOptions& rhs) :
00124     mrdb_db(rhs.mrdb_db),
00125     miningAlgorithms(rhs.miningAlgorithms),
00126     logStream(this,"logstream"),
00127     errStream(this,"errstream"),
00128     warnStream(this,"warnstream"),
00129     debugStream(this,"debugstream")  {  }
00130
00131
00132 public:
00133   /* ============================
00134    * Public Members
00135    * ============================ */
00136
00137   virtual std::string className() const {
00138     return "root";
00139   }
00140
00141   virtual void setOption(const std::string& name, const std::string& value)
00142     {
00143     throw MineruleException(MR_ERROR_OPTION_PARSING,
00144                             "Attempting to set an option in the root class");
00145   }
00146
00147   virtual OptionBase& subclassForName(const std::string& oclass)
00148     ;
00149
00150   static
00151     MineruleOptions& getSharedOptions() {
00152     if( !sharedOptions.isReady() )
00153         sharedOptions.init();
00154
00155     return sharedOptions;
00156   }
00157
00158   void
00159     readFromFile(std::string filename) ;
00160
00161   void
00162     readFromString(const std::string&) ;
00163
00164
00165   // set the name of the file from which the current
00166   // minerule was read
00167   void setMineruleSourceName(const std::string& name) {
00168     mineruleSourceName=name;
00169   }
00170
00171   // see set method
00172   const std::string&
00173     getMineruleSourceName() const {
00174     return mineruleSourceName;
00175   }
00176
00177   // set the name of the minerule as it appears in
00178   // the minerule statement
00179   void setMineruleName(const std::string& name) {
00180     mineruleName = name;
00181   }
00182
00183   const std::map< std::string, MRLogger*>&
00184     getKnownStreams() const {
00185     return knownStreams;
00186   }
00187
00188   // see set method
00189   const std::string&
00190     getMineruleName() const {
00191     return mineruleName;
00192   }
00193
00194   Mrdb&
00195     getMRDB() {
00196     return mrdb_db;
00197   }
00198
00199   const Mrdb&
00200     getMRDB() const {
00201     return mrdb_db;
00202   }
00203
00204   const Safety&
```

```
00205   getSafety() const {
00206     return safety;
00207   }
00208
00209   Safety&
00210   getSafety() {
00211     return safety;
00212   }
00213
00214   Optimizations&
00215     getOptimizations()  {
00216     return optimizations;
00217   }
00218
00219   const Optimizations&
00220     getOptimizations() const {
00221     return optimizations;
00222   }
00223
00224   MiningAlgorithms&
00225     getMiningAlgorithms()  {
00226     return miningAlgorithms;
00227   }
00228
00229   const MiningAlgorithms&
00230     getMiningAlgorithms() const {
00231     return miningAlgorithms;
00232   }
00233
00234   std::ostream&
00235     getLogStream() const {
00236                 return logStream.getStream() << StringUtils::toGreen("Log:");
00237   }
00238
00239   std::ostream&
00240     getErrStream() const {
00241     return errStream.getStream() << StringUtils::toRed("ERROR:");
00242   }
00243
00244   std::ostream&
00245     getWarnStream() const {
00246     return warnStream.getStream() << StringUtils::toYellow("WARNING:");
00247   }
00248
00249   std::ostream&
00250     getDebugStream() const {
00251     return debugStream.getStream() << StringUtils::toOrange("DEBUG:");
00252   }
00253
00254   OutStream&
00255     getLogStreamObj()  {
00256     return logStream;
00257   }
00258
00259   const OutStream&
00260     getLogStreamObj() const {
00261     return logStream;
00262   }
00263
00264   OutStream&
00265     getErrStreamObj()  {
00266     return errStream;
00267   }
00268
00269   const OutStream&
00270     getErrStreamObj() const {
00271     return errStream;
00272   }
00273
00274   OutStream&
00275     getWarnStreamObj()  {
00276     return warnStream;
00277   }
00278
00279   const OutStream&
00280     getWarnStreamObj() const {
00281     return warnStream;
00282   }
00283
00284   OutStream&
00285     getDebugStreamObj()  {
00286     return debugStream;
00287   }
00288
00289   const OutStream&
00290     getDebugStreamObj() const {
00291     return debugStream;
```

```
00292   }
00293
00294   Parsers&
00295     getParsers() {
00296     return parsers;
00297   }
00298
00299   const Parsers&
00300     getParsers() const {
00301     return parsers;
00302   }
00303
00304   std::ostream& saveOptions(std::ostream& os) const;
00305
00306   // this could be used to store particular user options
00307   std::map<std::string,std::string>& getUserOptions() {
00308     return userOptions;
00309   }
00310 };
```

7.189 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MineruleOptions_implementations/safety.hpp File Reference

Data Structures

- class Safety

7.190 safety.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 class Safety : public OptionBase {
00017   bool overwriteHomonymMinerules;
00018   bool allowCascadeDeletes;
00019 public:
00020   Safety() : overwriteHomonymMinerules(false),
00021              allowCascadeDeletes(false) {}
00022   virtual ~Safety() {}
00023
00024   virtual std::string className() const {
00025         return "safety";
00026   }
00027
00028   virtual void setOption(const std::string& name, const std::string& value)
00029         ;
00030
00031   bool getOverwriteHomonymMinerules() const { return overwriteHomonymMinerules; };
00032   void setOverwriteHomonymMinerules(bool rhs) { overwriteHomonymMinerules = rhs; };
00033
00034   bool getAllowCascadeDeletes() const { return allowCascadeDeletes; }
00035   void setAllowCascadeDeletes(bool rhs) { allowCascadeDeletes = rhs; }
00036 };
```

7.191 /Users/esposito/Software/minerule/include/minerule/Utils/MinMaxPair.hpp File Reference

```
#include <iostream>
#include <assert.h>
```

Data Structures

- class minerule::MinMaxPair

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const MinMaxPair &p)

7.192 MinMaxPair.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __MINMAXPAIR_H__
00017 #define __MINMAXPAIR_H__
00018
00019 #include <iostream>
00020 #include <assert.h>
00021
00022 namespace minerule {
00023
00024 class MinMaxPair {
00025  private:
00026   int _min;
00027   int _max;
00028
00029   static int _MIN(int m1, int m2) {
00030     if(m1<m2)
00031       return m1;
00032     else
00033       return m2;
00034   }
00035
00036   static int _MAX(int m1, int m2) {
00037     if(m1>m2)
00038       return m1;
00039     else
00040       return m2;
00041   }
00042
00043  public:
```

```
00044   MinMaxPair(int m,int M) : _min(m), _max(M) {
00045     assert( _min <= _max );
00046   };
00047
00048   ~MinMaxPair() {};
00049
00050   void applyConstraints(const MinMaxPair& constr) {
00051     assert( _min<=_max );
00052     assert( constr._min <= constr._max );
00053
00054     // the minimal el, cannot be lower than constr.minimal so we set
00055     // it to the maximum of the two
00056     _min = _MAX( _min, constr._min );
00057     // at the same time it cannot be higher than constr.maximal so we
00058     // set it to the minimum of the two
00059     _min = _MIN( _min, constr._max );
00060
00061     // An analogous reasoning lead to the following two lines of code
00062     _max = _MIN( _max, constr._max );
00063     _max = _MAX( _max, constr._min );
00064
00065     assert( _min<=_max );
00066   }
00067
00068   void setMin(int m) {
00069     _min=m;
00070   }
00071
00072   void setMax(int M) {
00073     _max=M;
00074   }
00075
00076   int getMin() const {
00077     return _min;
00078   }
00079
00080   int getMax() const {
00081     return _max;
00082   }
00083
00084   int getDefaultMax() const {
00085     return 1000;
00086   }
00087
00088   void setDefaultMax() {
00089     setMax(getDefaultMax());
00090   }
00091
00092   bool validate( int value ) const {
00093     if( _min <= value && value <= _max )
00094       return true;
00095     else
00096       return false;
00097   }
00098
00099   bool contains(const MinMaxPair& mm) const {
00100     return _min<=mm._min && mm._max<=_max;
00101   }
00102 };
00103
00104  inline std::ostream& operator«(std::ostream& os, const MinMaxPair& p) {
00105    os « "MinMax(" « p.getMin() « "," « p.getMax() « ")";
00106    return os;
00107  }
00108 } // namespace
00109
00110
00111
00112 #endif
```

7.193 /Users/esposito/Software/minerule/include/minerule/Utils/↩ MRLogger.hpp File Reference

```
#include <vector>
#include <iostream>
#include <fstream>
#include <sys/time.h>
#include <sstream>
```

```
#include <map>
```

Data Structures

- class minerule::MRLogger

Namespaces

- namespace minerule

7.194 MRLogger.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <vector>
00017 #include <iostream>
00018 #include <fstream>
00019 #include <sys/time.h>
00020 #include <sstream>
00021 #include <map>
00022
00023
00024 namespace minerule {
00025
00026         class MRLogger {
00027         private:
00028                 class LogInfo {
00029                 public:
00030                         std::string indent;
00031                         clock_t cpu;
00032                         struct timeval time;
00033
00034                         LogInfo(const LogInfo& li) : indent(li.indent), cpu(li.cpu), time(li.time) { }
00035
00036                         LogInfo(const std::string& ind) :
00037                         indent(ind), cpu(clock()) {
00038                                 gettimeofday(&time,NULL);
00039                         };
00040
00041                         LogInfo& operator=(const LogInfo& li) {
00042                                 indent=li.indent;
00043                                 cpu=li.cpu;
00044                                 time=li.time;
00045                                 return *this;
00046                         }
00047                 }; // LogInfo
00048
00049                 class MeasurementInfo {
00050                 public:
00051                         LogInfo start;
00052                         double totCpu;
00053                         double totTime;
00054                         bool startIsValid;
00055
00056                         MeasurementInfo() : start(""), totCpu(0.0), totTime(0.0), startIsValid(false)
      {};
00057                 };
00058
```

```
00059
00060                 static const std::string START_SEPARATOR;
00061                 static const std::string CONT_SEPARATOR;
00062                 static const std::string END_SEPARATOR;
00063                 static std::ofstream nullLog;
00064
00065                 typedef std::vector< LogInfo > LogStack;
00066                 typedef std::map< std::string, MeasurementInfo > Measurements;
00067
00068                 LogStack logStack;
00069                 Measurements measurements;
00070
00071                 std::string indentString;
00072                 std::string indentInset;
00073                 std::ostream* os;
00074                 size_t logLevel;    // the desired logLevel
00075                 size_t curLogLevel; // the current logLevel
00076
00077                 void indent() {
00078                         *os<<indentString;
00079                 }
00080
00081                 double getTimeSecs(const LogInfo& last, const LogInfo& first) const {
00082                         return double(last.time.tv_sec-first.time.tv_sec)
00083                                 +double((last.time.tv_usec-first.time.tv_usec)/1e6);
00084                 }
00085
00086                 double getCpuSecs(const LogInfo& last, const LogInfo& first) const {
00087                         return double(last.cpu-first.cpu)/double(CLOCKS_PER_SEC);
00088                 }
00089
00090                 std::string evalTimeMemInfo(const LogInfo& li) const;
00091
00092                 void logMeasurement(const std::string& description, const MeasurementInfo& data);
00093         public:
00094                 MRLogger(std::ostream& ostr);
00095                 MRLogger();
00096                 ~MRLogger(void);
00097
00098                 void setStream(std::ostream& ostr);
00099                 std::ostream* getStream() const {
00100                         return os;
00101                 }
00102
00103                 void updateIndentString();
00104                 void push(const std::string& descr);
00105                 void pop();
00106
00107                 double getCurrentCpuSecs() const;
00108                 double getCurrentTimeSecs() const;
00109                 double getCurrentCpuDelta() const;
00110                 double getCurrentTimeDelta() const;
00111
00112
00113                 std::ostream& log() {
00114                         if(curLogLevel>logLevel)
00115                                 return nullLog;
00116                         indent();
00117                         *os<<" ";
00118                         return *os;
00119                 }
00120
00121                 size_t getLogLevel() {
00122                         return logLevel;
00123                 }
00124
00125                 void setLogLevel(size_t newLevel) {
00126                         logLevel=newLevel;
00127                 }
00128
00129                 size_t getIndentLen() { return indentString.size(); }
00130
00131                 void startMeasuring(std::string description) {
00132                         MeasurementInfo& m = measurements[description];
00133                         assert(!m.startIsValid);
00134
00135                         m.start = LogInfo(""); // saving start time
00136                         m.startIsValid = true;
00137                 }
00138
00139                 void stopMeasuring(std::string description) {
00140                         MeasurementInfo& m = measurements[description];
00141                         LogInfo stop("");
00142
00143                         assert(m.startIsValid);
00144
00145                         m.totCpu += getCpuSecs(stop, m.start);
```

```
00146                         m.totTime += getTimeSecs(stop, m.start);
00147                         m.startIsValid = false;
00148                 }
00149
00150                 void logMeasurements();
00151
00152         }; // MRLogger
00153
00154
00155 } // namespace
```

7.195 /Users/esposito/Software/minerule/include/minerule/Utils/OptionParserLib.hpp File Reference

```
#include "MineruleOptions.hpp"
```

Namespaces

- namespace minerule

Functions

- void minerule::initializeOptionsFromFile (MineruleOptions &mrOpts, FILE *file)
- void minerule::initializeOptionsFromString (MineruleOptions &mrOpts, std::string str)
- void minerule::pushOptionClassIntoContext (const std::string &oclass)
- void minerule::popOptionClassFromContext ()
- void minerule::setOption (const std::string &name, const std::string &value)

7.196 OptionParserLib.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef __OPTION_PARSER_LIB_H__
00017 #define __OPTION_PARSER_LIB_H__
00018
00019 #include "MineruleOptions.hpp"
00020
00021
00022 namespace minerule {
00023   void initializeOptionsFromFile(MineruleOptions& mr, FILE* file);
00024   void initializeOptionsFromString(MineruleOptions& mr, std::string);
00025
00026   /* the following functions are used by the option parser */
00027   void pushOptionClassIntoContext(const std::string& oclass) ;
00028   void popOptionClassFromContext();
00029   void setOption(const std::string& name, const std::string& value);
00030 } // namespace
00031
00032 #endif
```

7.197 /Users/esposito/Software/minerule/include/minerule/Utils/↩ Progress.hpp File Reference

```
#include <iostream>
```

Data Structures

- class minerule::Progress

Namespaces

- namespace minerule

Functions

- void minerule::clockHandsUpdateHandler (int n, int count, int num)
- void minerule::tickNumberUpdateHandler (int n, int count, int num)
- void minerule::barSpeenWheelUpdateHandler (int n, int count, int num)
- void minerule::angledSpeenWheelUpdateHandler (int n, int count, int num)

7.198 Progress.hpp

Go to the documentation of this file.

```
00001 #ifndef MR_PROGRESS_HPP_K3MX32K3U2L__
00002 #define MR_PROGRESS_HPP_K3MX32K3U2L__
00003
00004 #include <iostream>
00005
00006 namespace minerule {
00007
00008 // Implements an indeterminate progress indicator
00009 class Progress {
00010 public:
00011   typedef void (*UpdateHandler)(int n, int count, int num);
00012
00013 private:
00014   static UpdateHandler defaultHandler_;
00015
00016         UpdateHandler handler_;
00017   int count_;
00018   int num_;
00019
00020 public:
00021   Progress(int num) : count_(0), num_(100), handler_(defaultHandler_) {}
00022   ~Progress() {}
00023
00024   void start() {
00025         count_=0;
00026         handler_(-1, 0, 0);
00027   }
00028
00029   void end() {
00030     handler_(-2, 0, 0);
00031   }
00032
00033
00034   void tick(int n=1) {
00035     handler_(n, count_, num_);
00036     count_+=n;;
00037   }
00038
00039         // helper function for default behavior on start/stop call
```

```
00040         static bool handleStartStop(std::ostream&, int n);
00041
00042   static UpdateHandler setDefaultHandler(UpdateHandler handler) {
00043     UpdateHandler tmp = defaultHandler_;
00044     defaultHandler_ = handler;
00045     return tmp;
00046   }
00047
00048         void setHandler(UpdateHandler handler) {
00049                 handler_ = handler;
00050         }
00051 };
00052
00053
00054 extern void clockHandsUpdateHandler(int,int,int);
00055 extern void tickNumberUpdateHandler(int,int,int);
00056 extern void barSpeenWheelUpdateHandler(int,int,int);
00057 extern void angledSpeenWheelUpdateHandler(int,int,int);
00058
00059 } /* minerule */
00060
00061 #endif /* end of include guard: MR_PROGRESS_HPP_K3MX32K3U2L__ */
```

7.199 /Users/esposito/Software/minerule/include/minerule/Utils/SQLUtils.hpp File Reference

```
#include <string>
#include <string.h>
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/Types.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/Converter.hpp"
```

Data Structures

- class minerule::SQLUtils

Namespaces

- namespace minerule

7.200 SQLUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
00016 #ifndef __SQLUTILS_H__
00017 #define __SQLUTILS_H__
00018
00019 #include <string>
00020 #include <string.h>
00021 #include "minerule/mrdb/Connection.hpp"
00022 #include "minerule/mrdb/Types.hpp"
00023 #include "minerule/mrdb/ResultSet.hpp"
00024 #include "minerule/mrdb/ResultSetMetaData.hpp"
00025 #include "minerule/Utils/MineruleException.hpp"
00026 #include "minerule/Utils/Converter.hpp"
00027
00028
00029 namespace minerule {
00030
00031   class SQLUtils {
00032   public:
00033     typedef enum {
00034       Numeric,
00035       String,
00036       Binary,
00037       DateTime,
00038       Bit
00039     } Type;
00040
00041     static void
00042     removeHeadBodyFromAttrName( std::string& str );
00043
00044     static Type
00045       getType( mrdb::Types::SQLType ) ;
00046
00047     static Type
00048       getType( mrdb::ResultSet* rs, int colNum ) {
00049       return getType( (mrdb::Types::SQLType)
00050                       rs->getMetaData()->getColumnType(colNum) );
00051     }
00052
00053     static Type
00054       getType( mrdb::Connection* connection,
00055                const std::string& tabName,
00056                std::string colName );
00057
00058     static bool
00059       isNumericType(mrdb::Types::SQLType type) {
00060       return getType(type)==Numeric;
00061     }
00062
00063     static bool
00064       isStringType(mrdb::Types::SQLType type) {
00065       return getType(type)==String;
00066     }
00067
00068     static bool
00069       isBinaryType(mrdb::Types::SQLType type) {
00070       return getType(type)==Binary;
00071     }
00072
00073     static bool
00074       isDateTimeType(mrdb::Types::SQLType type) {
00075       return getType(type)==DateTime;
00076     }
00077
00078     static bool
00079       isBitType(mrdb::Types::SQLType type) {
00080       return getType(type)==Bit;
00081     }
00082
00083     static std::string quote(const std::string& str);
00084
00085     static bool isAttribute(const std::string& str) {
00086     if(str.length()==0)
00087       return false;
00088
00089     if( isdigit(str[0]) )
00090       return false;
00091
00092     if( str.find_first_of("\"'")!=str.npos )
00093       return false;
00094
00095     if( Converter(str).isNumber() )
00096       return false;
00097
00098     return true;
00099     }
00100
00101   };
00102
```

```
00103 } // namespace minerule
00104
00105
00106 #endif
```

7.201 /Users/esposito/Software/minerule/include/minerule/Utils/StringUtils.hpp File Reference

```
#include <vector>
#include <string>
```

Data Structures

- class minerule::StringUtils

Namespaces

- namespace minerule

7.202 StringUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef STRINGUTILS_H_7KCWVFQO
00017 #define STRINGUTILS_H_7KCWVFQO
00018
00019 #include <vector>
00020 #include <string>
00021
00022 namespace minerule {
00023         class StringUtils {
00024         private:
00025                 static bool enableColors;
00026                 static std::string colorize(const std::string& str, const std::string& color) { return
       enableColors ? color + str + CLOSE : str; }
00027
00028         public:
00029                 static std::vector<std::string>* splitToLength(const std::string& str, size_t len);
00030                 static std::vector<std::string> split(const std::string& str, const std::string& sep);
00031                 static std::string join(const std::vector<std::string>&, const std::string& sep );
00032
00033                 static const std::string RED;
00034                 static const std::string BOLD;
00035                 static const std::string GREEN;
00036                 static const std::string YELLOW;
00037                 static const std::string BLUE;
00038                 static const std::string WHITE;
00039                 static const std::string CLOSE;
00040
00041                 static void setColorsEnabled(bool newVal) { enableColors = newVal; }
```

```
00042                 static std::string toBold(const std::string& str){ return colorize(str, BOLD); }
00043                 static std::string toRed(const std::string& str)   { return colorize(str,RED);   }
00044                 static std::string toBoldRed(const std::string& str) { return colorize(colorize(str,
      RED), BOLD); }
00045                 static std::string toGreen(const std::string& str) { return colorize(str,GREEN); }
00046                 static std::string toWhite(const std::string& str) { return colorize(str,WHITE); }
00047                 static std::string toBlue(const std::string& str)         { return colorize(str,BLUE);
      }
00048                 static std::string toYellow(const std::string& str){ return colorize(str,YELLOW); }
00049                 static std::string toOrange(const std::string& str){ return
      colorize(colorize(str,YELLOW), BOLD); }
00050
00051         };
00052 }
00053
00054 #endif /* end of include guard: STRINGUTILS_H_7KCWVFQO */
```

7.203 /Users/esposito/Software/minerule/src/Algorithms/Algorithms.cpp File Reference

```
#include "minerule/Algorithms/Algorithms.hpp"
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
#include "minerule/Algorithms/ConstrTree.hpp"
#include "minerule/Algorithms/DestrTree.hpp"
#include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
#include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
#include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
#include "minerule/Algorithms/FSMiner.hpp"
#include "minerule/Algorithms/CCSMiner.hpp"
#include "minerule/Algorithms/STSMiner.hpp"
#include "minerule/mrdb/SQLException.hpp"
```

Namespaces

- namespace minerule

7.204 Algorithms.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/Algorithms.hpp"
00017 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00018 #include "minerule/Parsers/ParsedMinerule.hpp"
00019 #include "minerule/Utils/MineruleOptions.hpp"
00020 #include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
```

```
00021 #include "minerule/Algorithms/ConstrTree.hpp"
00022 #include "minerule/Algorithms/DestrTree.hpp"
00023 #include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
00024 #include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
00025 #include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
00026 #include "minerule/Algorithms/FSMiner.hpp"
00027 #include "minerule/Algorithms/CCSMiner.hpp"
00028 #include "minerule/Algorithms/STSMiner.hpp"
00029 #include "minerule/mrdb/SQLException.hpp"
00030
00031 namespace minerule {
00032
00033         MiningAlgorithmBase* Algorithms::getBestRulesMiningAlgorithm(const OptimizedMinerule& mr) {
00034                 MRDebugPusher pusher("Choosing the best algorithm for the given MR");
00035
00036                 AlgorithmTypes userChoiceOfAT =
      MineruleOptions::getSharedOptions().getMiningAlgorithms().getRulesMiningAlgorithms().getPreferredAlgorithm();
00037
00038                 MiningAlgorithmBase* userChoice= MiningAlgorithm::algorithmForType(userChoiceOfAT,
      mr);
00039                 if( userChoice->canHandleMinerule() ) {
00040                         MRDebug() << "Selected the algorithm given by the user preference" << std::endl;
00041                         return userChoice;
00042                 }
00043                 else {
00044                         MRDebug() << "User preference cannot be fulfilled" << std::endl;
00045                         delete userChoice;
00046                         userChoice=NULL;
00047                 }
00048
00049  // we failed to satisfy user preference, it is up to us
00050  // to find the best algorithm.
00051
00052                 if( BFSWithGidsNoCross(mr).canHandleMinerule() ) {
00053                         MRDebug() << "Selected BFSWithGidsNoCross" << std::endl;
00054                         return new BFSWithGidsNoCross(mr);
00055                 }
00056
00057                 MRDebug() << "BFSWithGidsNoCross cannot handle it." << std::endl;
00058
00059                 if( BFSWithGidsAndCross(mr).canHandleMinerule() ) {
00060                         MRDebug() << "Selected BFSWithGidsAndCross" << std::endl;
00061                         return new BFSWithGidsAndCross(mr);
00062                 }
00063
00064                 MRDebug() << "BFSWithGidsAndCross cannot handle it." << std::endl;
00065
00066
00067                 MRDebug() << "Panic! No known algorithm can handle it." << std::endl;
00068                 throw MineruleException( MR_ERROR_INTERNAL, "No known algorithm can handle the given
      minerule!" );
00069         }
00070
00071
00072
00073         MiningAlgorithmBase*
00074         Algorithms::getBestItemsetsMiningAlgorithm(const OptimizedMinerule& mr) {
00075                 MRDebugPusher pusher("Choosing the best algorithm for the given MR");
00076
00077                 AlgorithmTypes userChoiceOfAT =
00078
      MineruleOptions::getSharedOptions().getMiningAlgorithms().getRulesMiningAlgorithms().getPreferredAlgorithm();
00079
00080                 MiningAlgorithmBase* userChoice= MiningAlgorithm::algorithmForType(userChoiceOfAT,
      mr);
00081                 if( userChoice->canHandleMinerule() ) {
00082                         MRDebug() << "Selected the algorithm given by the user preference" << std::endl;
00083                         return userChoice;
00084                 }
00085                 else {
00086                         MRDebug() << "User preference cannot be fulfilled" << std::endl;
00087                         delete userChoice;
00088                         userChoice=NULL;
00089                 }
00090
00091  // we failed to satisfy user preference, it is up to us
00092  // to find the best algorithm.
00093
00094                 if( ConstrItemSetsExtraction(mr).canHandleMinerule() ) {
00095                         MRDebug() << "Selected ConstrItemSetsExtraction" << std::endl;
00096                         return new ConstrItemSetsExtraction(mr);
00097                 }
00098
00099                 MRDebug() << "Panic! No known algorithm can handle it." << std::endl;
00100                 throw MineruleException( MR_ERROR_INTERNAL, "No known algorithm can handle the given
      minerule!" );
00101         }
```

```
00102
00103
00104
00105         MiningAlgorithmBase*
00106         Algorithms::getBestSequencesMiningAlgorithm( const OptimizedMinerule& mr ) {
00107                 AlgorithmsOptions opts;
00108
00109                 opts.setSupport( mr.getParsedMinerule().sup );
00110                 opts.setBodyCardinalities( mr.getParsedMinerule().bodyCardinalities );
00111
00112                 //CCSMiner* miner = new CCSMiner(mr,opts);
00113                 STSMiner* miner= new STSMiner(mr,opts);
00114                 if(miner->canHandleMinerule())
00115                         return miner;
00116                 else {
00117                         delete miner;
00118                         throw MineruleException( MR_ERROR_INTERNAL, "Cannot handle specified mine
      request." );
00119                 }
00120         }
00121
00122 //
00123 //      This procedure crate a new algorithm and returns its reference.
00124 //      As more algorithms become available it will choose among them
00125 //      using some (hopefully) good heuristic... In modifying it please
00126 //      remember that any future criteria must ensure that the chosen
00127 //      algorithm is able to deal with SourceRowColumnIds that
00128 //      will be passed to it.
00129 //      Remember that the informations stored in SourceRowColumnIds
00130 //      depends upon mr.mineruleRequiresClusters().
00131 //
00132         MiningAlgorithmBase* Algorithms::newAlgorithm(const OptimizedMinerule& mr) {
00133                 switch( mr.getParsedMinerule().miningTask ) {
00134                         case MTMineRules:            return getBestRulesMiningAlgorithm(mr);
00135                         case MTMineItemsets:      return getBestItemsetsMiningAlgorithm(mr);
00136                         case MTMineSequences:    return getBestSequencesMiningAlgorithm(mr);
00137                         default: throw MineruleException( MR_ERROR_INTERNAL, "Cannot handle
      "+miningTaskToString(mr.getParsedMinerule().miningTask)+" mining task");
00138                 }
00139         }
00140
00141         bool Algorithms::executeIncrementalAlgorithm(OptimizedMinerule& mr) {
00142
00143                 IncrementalAlgorithm* incrAlgo = IncrementalAlgorithm::newIncrementalAlgorithm(mr);
00144
00145                 if( incrAlgo!=NULL ) {
00146                         incrAlgo->execute();
00147
00148                         delete incrAlgo;
00149                         return true;
00150                 } else {
00151                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00152                                         « " to the system yet." « std::endl;
00153                         return false;
00154                 }
00155         }
00156
00157
00158
00159         void Algorithms::executeExtractionAlgorithm(OptimizedMinerule& mr) {
00160                 MiningAlgorithmBase* algo =  Algorithms::newAlgorithm(mr);
00161                 algo->execute();
00162                 delete algo;
00163         }
00164
00165         void Algorithms::checkAndHandleHomonymMinerules(OptimizedMinerule& mr) {
00166                 if( OptimizerCatalogue::existsMinerule(mr.getParsedMinerule().tab_result) ) {
00167                         if(
      MineruleOptions::getSharedOptions().getSafety().getOverwriteHomonymMinerules() ) {
00168                                 MRLog() « "The optimizer Catalogue reports that a minerule " «
      std::endl;
00169                                 MRLog() « "having the same name as the one you gave already" «
      std::endl;
00170                                 MRLog() « "exists. I'm now going to delete the previous result as" «
      std::endl;
00171                                 MRLog() « "it has been specified in the option settings." « std::endl;
00172
00173                                 OptimizerCatalogue::deleteMinerule(mr.getParsedMinerule().tab_result);
00174
00175                         } else {
00176                                 throw MineruleException(MR_ERROR_MINERULE_ALREADY_EXISTS,
00177                                         "The Optimizer Catalogue reports that a minerule "
00178                                         "having the same name as the one you gave already"
00179                                         "exists. Please change the name of your current "
00180                                         "minerule or set the option "
00181                                         "safety::overwriteHomonymMinerules "
```

```
00182                                           "to true in your configuration file");
00183                           }
00184                   }
00185           }
00186
00187           void Algorithms::showDebugInfo(const std::string& msg, OptimizedMinerule& mr) {
00188                   MRDebugPush("Unoptimized Minerule info");
00189                   MRDebug("Optimized Minerule:[" + mr.getParsedMinerule().getText() + "]");
00190                   MRDebug() « "Body attribute list size:" « mr.getParsedMinerule().ba.size() «
      std::endl;
00191                   MRDebug() « "Head attribute list size:" « mr.getParsedMinerule().ha.size() «
      std::endl;
00192                   MRDebug() « "Rule attribute list size:" « mr.getParsedMinerule().ra.size() «
      std::endl;
00193                   MRDebugPop();
00194           }
00195
00196           void Algorithms::executeMinerule(OptimizedMinerule& mr) {
00197                   checkAndHandleHomonymMinerules(mr);
00198
00199                   showDebugInfo("Unoptimized Minerule info", mr);
00200                   mr.optimize(); // internally it will check if the optimization option is set.
00201                   showDebugInfo("Optimized Minerule info", mr);
00202
00203
00204                   std::string unsupportedRelation = "";
00205                   OptimizerCatalogue::MineruleResultInfo result(mr.getParsedMinerule());
00206
00207                   switch(mr.getOptimizationInfo().relationship) {
00208
00209                           // EQUIVALENCE
00210                           case OptimizedMinerule::Equivalence:
00211                                   {
00212                                           MRLog("Using equivalence relationship.");
00213
00214                                           CatalogueInfo catInfo;
00215                                           OptimizerCatalogue::getMRQueryInfo(
      mr.getOptimizationInfo().minerule.tab_result, catInfo );
00216                                           result.resultset = catInfo.resName;
00217
00218                                           MRDebug() « "Inclusion found with respect minerule:" «
      mr.getOptimizationInfo().minerule.getText() « std::endl;
00219                                           MRDebug() « "Current Minerule:" « result.getText() «
      std::endl;
00220                                           OptimizerCatalogue::addMineruleResult(result);
00221                                   }
00222                                   break;
00223
00224                           // DOMINANCE
00225                           case OptimizedMinerule::Dominance:
00226                                   unsupportedRelation = "Dominance";
00227                                   MRLog("Using dominance relationship.");
00228
00229                           // COMBINATION
00230                           case OptimizedMinerule::Combination:
00231                                   if(unsupportedRelation=="") {
00232                                           MRLog("Using combination relationship.");
00233
00234                                           unsupportedRelation="Combination";
00235                           }
00236
00237                                   if(executeIncrementalAlgorithm(mr)) {
00238                                           OptimizerCatalogue::addMineruleResult(result);
00239                                           break;
00240                                   } else {
00241                                           MRLog() « "The support for the found dominance relationship is
      not yet implemented"
00242                                                   « " switching back to the non-incremental mining
      algorithm;" « std::endl;
00243                                   }
00244
00245                           // INCLUSION
00246                           case OptimizedMinerule::Inclusion:
00247                                   if( unsupportedRelation=="" ) {
00248                                           unsupportedRelation = "Inclusion";
00249                                           MRLog("Using inclusion relationship.");
00250                                   }
00251
00252                           // NO RELATION FOUND
00253                           case OptimizedMinerule::None:
00254                                   if( unsupportedRelation!="" ) {
00255                                           MRWarn() « "The optimizer found that there exists a minerule
      in the catalogue" « std::endl
00256                                                   « "which is in `" « unsupportedRelation «"'
      relationship with the" « std::endl
00257                                                   « "current one. Unfortunately such kind of
      relationship is still" « std::endl
```

```
00258                                                 « "not supported and hence I will switch to the
      default algorithm" « std::endl;
00259                                 }
00260
00261                                 executeExtractionAlgorithm(mr);
00262
00263                                 if( mr.getParsedMinerule().miningTask==MTMineRules ||
      mr.getParsedMinerule().miningTask==MTMineItemsets )
00264                                                 OptimizerCatalogue::addMineruleResult(result);
00265                                 break;
00266
00267                         // ERROR
00268                         default:
00269                                 throw MineruleException(MR_ERROR_INTERNAL, "Unexpected Relationship!
      This is a BUG! Please report it!");
00270                 }
00271         }
00272
00273
00274 } // namespace minerule
```

7.205 /Users/esposito/Software/minerule/src/Algorithms/Algorithms↩ Options.cpp File Reference

```
#include "minerule/Algorithms/AlgorithmsOptions.hpp"
```

7.206 AlgorithmsOptions.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/AlgorithmsOptions.hpp"
00017
00018 using namespace minerule;
00019
00020 void AlgorithmsOptions::setSupport(double sup) { support = sup; }
00021
00022 void AlgorithmsOptions::setConfidence(double conf) { confidence = conf; }
00023
00024 void AlgorithmsOptions::setSourceRowDescription(
00025     const SourceRowColumnIds &srdes) {
00026   sourceRowDescription = srdes;
00027 }
00028
00029 double AlgorithmsOptions::getSupport() const { return support; }
00030
00031 double AlgorithmsOptions::getConfidence() const { return confidence; }
00032
00033 /*
00034 unsigned int
00035 AlgorithmsOptions::getRowsPerPartition() const {
00036   return rowsPerPartition;
00037   }*/
00038
00039
00040 const SourceRowColumnIds &AlgorithmsOptions::getSourceRowDescription() const {
00041   return sourceRowDescription;
00042 }
```

7.207 /Users/esposito/Software/minerule/src/Algorithms/BFSWithGids↩ AndCross.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
#include <iterator>
#include <algorithm>
#include "minerule/Algorithms/AlgorithmsOptions.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
```

Namespaces

- namespace minerule

7.208 BFSWithGidsAndCross.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 // CORE.CPP
00017
00018 #include<stdio.h>
00019 #include<string.h>
00020 #include<stdlib.h>
00021 #include <iostream>
00022 #include <iterator>
00023 #include <algorithm>
00024
00025
00026 #include "minerule/Algorithms/AlgorithmsOptions.hpp"
00027 #include "minerule/Utils/MineruleOptions.hpp"
00028 #include "minerule/Database/MRResultSet.hpp"
00029 #include "minerule/Database/ItemType.hpp"
00030 #include "minerule/Utils/Converter.hpp"
00031 #include "minerule/Database/Connection.hpp"
00032 #include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
00033 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00034 #include "minerule/Database/PrepareDataUtils.hpp"
00035
00036 namespace minerule {
00037
00038         bool BFSWithGidsAndCross::mineruleHasSameBodyHead = false;
00039
```

```
00040         void BFSWithGidsAndCross::BodyMapElement::insert(const ItemType& item, MapElement& gidList,
      bool secondPass) {
00041                 std::map<ItemType, MapElement>::iterator found = heads.find(item);
00042                 if (found == heads.end()) {
00043                         if (!secondPass) heads[item] = gidList;
00044                         } else found->second.insert(gidList.begin(),gidList.end());
00045                 }
00046
00047                 bool BFSWithGidsAndCross::BodyMapElement::pruneMap (float threshold) {
00048                         std::map<ItemType, MapElement> newMap;
00049                         for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i !=
      heads.end(); i++)
00050                                 if (i->second.count >= threshold) {i->second.count = 0;
      newMap[i->first] = i->second;}
00051                         heads = newMap;
00052                         return heads.size() > 0;
00053                 }
00054
00055                 bool BFSWithGidsAndCross::BodyMapElement::updateCount () {
00056                         for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i !=
      heads.end(); i++)
00057                                 i->second.count += i->second.size();
00058                         return heads.size() > 0;
00059                 }
00060
00061
00062 //  int BFSWithGidsAndCross::BodyMap::add(ItemType& gid, Transaction& t1, Transaction& t2, bool
      secondPass) {
00063                 int BFSWithGidsAndCross::BodyMap::add(int gid, Transaction& t1, bool secondPass) {
00064                         int howMany = 0;
00065                         MapElement me;
00066                         me.insert(gid);
00067                         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00068                                 std::map<ItemType, BodyMapElement>::iterator found = find(i->first);
00069                                 if (found == end()) {
00070                                         if (secondPass) continue;
00071                                         BodyMapElement bme;
00072                                         (*this)[i->first] = bme;
00073                                         found = find(i->first);
00074                                 }
00075                                 found->second.insert(gid);
00076                                 found->second.insert(i->second,me,secondPass);
00077                                 howMany++;
00078                         }
00079                         return howMany;
00080                 }
00081
00082                 int BFSWithGidsAndCross::BodyMap::add(int gid, mrdb::ResultSet* rs, SourceRow& hbsr,
      bool secondPass) {
00083                         int howMany = 0;
00084                         MapElement me;
00085                         me.insert(gid);
00086                         ItemType body = hbsr.getBody();
00087                         std::map<ItemType, BodyMapElement>::iterator found = find(body);
00088                         if (found == end()) {
00089                                 if (secondPass) return 0;
00090                                 BodyMapElement bme;
00091                                 (*this)[body] = bme;
00092                                 found = find(body);
00093                         }
00094                         found->second.insert(gid);
00095                         ItemType head = hbsr.getHead();
00096                         found->second.insert(head,me,secondPass);
00097                         howMany++;
00098                         return howMany;
00099                 }
00100
00101  /*void BodyMap::saveMap (std::ostream& out, bool withGids) {
00102                 for (iterator i = begin(); i != end(); i++) {
00103                         out « i->first;
00104                         i->second.saveMap(out, withGids);
00105                 }
00106         }
00107
00108         void BodyMap::loadMap (istream& in, bool withGids) {
00109                 ItemType item;
00110                 while (!in.eof()) {
00111                         in » item;
00112                         if (in.eof()) break;
00113                         (*this)[item].loadMap(in, withGids);
00114                 }
00115         }
00116  */
00117         void BFSWithGidsAndCross::BodyMap::pruneMap (float threshold) {
00118                 BodyMap newMap(*connection, *progress);
00119                 for (iterator i = begin(); i != end(); i++)
00120                         if (i->second.pruneMap(threshold)) newMap[i->first] = i->second;
```

```
00121                 (*this) = newMap;
00122         }
00123
00124         void BFSWithGidsAndCross::BodyMap::updateCount () {
00125                 for (iterator i = begin(); i != end(); i++)
00126                         i->second.updateCount();
00127         }
00128
00129
00130         // void BFSWithGidsAndCross::BodyMap::createBodies (NewRuleSet& rs, float threshold, size_t
      maxBody) {
00131         //      for (iterator i = begin(); i != end(); i++) {
00132         //              NewRule r(i,i->second);
00133         //              rs.insert(rs.end(),r);
00134         //      }
00135         //      for (unsigned int n = 0; n < rs.size(); n++) {
00136         //              NewRuleSet::iterator i = rs.begin()+n;
00137         //              NewRule& rc = *i;
00138         //              if (rc.body.size() < maxBody) {
00139         //                      BodyMap::iterator lb = rc.lastBody;
00140         //                      for (BodyMap::iterator j = ++lb; j != end(); j++) {
00141         //                              GidList newGidList;
00142         //                              NewRule& rc2 = *(rs.begin()+n);
00143         //                              set_intersection(rc2.gids.begin(),rc2.gids.end(),
00144         //                                      j->second.begin(), j->second.end(),
00145         //                                              inserter(newGidList,newGidList.begin()));
00146         //                              if (newGidList.size() >= threshold) {
00147         //                                      NewRule r(rc2,j,newGidList);
00148         //                                      rs.insert(rs.end(),r);
00149         //                              }
00150         //                      }
00151         //              }
00152         //      }
00153         // }
00154
00155
00156         // void BFSWithGidsAndCross::BodyMap::createHeads (NewRuleSet& rs, NewRuleSet& rs1, float
      threshold, size_t maxHead) {
00157         //              for (size_t n = 0; n < rs.size(); n++) {
00158         //                      NewRule& rc = rs[n];
00159         //                      if (rc.head.size() < maxHead) {
00160         //                              map<ItemType, MapElement>::iterator lh = rc.lastHead;
00161         //                              map<ItemType, MapElement>::iterator eh = rc.headTable.end();
00162         //                              double supportBody = rc.gids.size();
00163         //
00164         //                              for (map<ItemType, MapElement>::iterator j = lh; j != eh; j++)
      {
00165         //                                      GidList newGidList;
00166         //                                      if (std::find(rc.body.begin(), rc.body.end(),
      j->first) == rc.body.end()/* &&
00167         //                                      rc.head.find(j->first) == rc.head.end()*/) {
00168         //
      set_intersection(rc.gids.begin(),rc.gids.end(),
00169         //                                                      j->second.begin(), j->second.end(),
00170         //
      inserter(newGidList,newGidList.begin()));
00171         //                                              if (newGidList.size() >= threshold) {
00172         //                                                      NewRule r(rc,j,newGidList);
00173         //                                                      r.lastHead++;
00174         //                                                      r.conf = newGidList.size() /
      supportBody;
00175         //                                                      rs1.insert(rs1.end(),r);
00176         //                                              }
00177         //                                      }
00178         //                              }
00179         //                      }
00180         //              }
00181         //      }
00182
00183         bool BFSWithGidsAndCross::BodyMap::checkHeads (NewRule& r, std::map<ItemType,
      BodyMapElement>::iterator b, GidList& g, float threshold) {
00184                 bool ok = true;
00185                 for (std::vector<ItemType>::iterator i = r.head.begin(); i != r.head.end() && ok; i++)
      {
00186                         std::map<ItemType, MapElement >::iterator found = b->second.heads.find(*i);
00187                         if (found == b->second.heads.end()) return false;
00188                         else {
00189                                 GidList newGidList;
00190                                 set_intersection((*found).second.begin(),(*found).second.end(),
00191                                         g.begin(), g.end(),
00192                                                 inserter(newGidList,newGidList.begin()));
00193                                 g = newGidList;
00194                                 ok = newGidList.size() >= threshold;
00195                         }
00196                 }
00197                 return ok;
00198         }
```

```
00199
00200         int BFSWithGidsAndCross::BodyMap::generateRules (float support, int totGroups, int maxBody,
      int maxHead) {
00201                 NewRuleSet rs1;
00202                 int howManyRules = 0;
00203                 float threshold = support*totGroups;
00204                 for (iterator i = begin(); i != end(); i++) {
00205                         progress->tick();
00206
00207                         NewRule r(i,i->second);
00208                         rs1.insert(rs1.end(),r);
00209                 }
00210                 for (size_t n = 0; n < rs1.size(); n++) {
00211                         NewRule& rc = *(rs1.begin()+n);
00212                         NewRuleSet rs;
00213                         addHead(rs, threshold, maxHead, rc.gids.size(), rc);
00214                         howManyRules += rs.size();
00215                         insertRules(rs, totGroups);
00216                         if (rc.body.size() < (size_t)maxBody) {
00217                                 BodyMap::iterator lb = rc.lastBody;
00218                                 for (BodyMap::iterator j = ++lb; j != end(); j++) {
00219                                         progress->tick();
00220
00221                                         NewRule& rc2 = *(rs1.begin()+n);
00222                                         GidList newGidList;
00223                                         set_intersection(rc2.gids.begin(),rc2.gids.end(),
00224                                                 j->second.begin(), j->second.end(),
00225                                                         inserter(newGidList,newGidList.begin()));
00226                                         if (newGidList.size() >= threshold) {
00227                                                 NewRule r(rc2,j,newGidList);
00228                                                 rs1.insert(rs1.end(),r);
00229                                         }
00230                                 }
00231                         }
00232                 }
00233                 return howManyRules;
00234         }
00235
00236         void BFSWithGidsAndCross::BodyMap::addHead (NewRuleSet& rs, float threshold, int maxHead, int
      suppBody, NewRule& rc) {
00237                 if (rc.head.size() < (size_t)maxHead) {
00238                         std::map<ItemType, MapElement>::iterator lh = rc.lastHead;
00239                         std::map<ItemType, MapElement>::iterator eh = rc.headTable.end();
00240                         for (map<ItemType, MapElement>::iterator j = lh; j != eh; j++) {
00241                                 GidList newGidList;
00242                                 if (!BFSWithGidsAndCross::getMineruleHasSameBodyHead() ||
      std::find(rc.body.begin(), rc.body.end(), j->first) == rc.body.end()) {
00243
00244                                         set_intersection(       rc.gids.begin(),rc.gids.end(),
00245                                                                                         j->second.begin(),
      j->second.end(),
00246
      inserter(newGidList,newGidList.begin()));
00247
00248                                         if (newGidList.size() >= threshold) {
00249                                                 NewRule r(rc,j,newGidList);
00250                                                 r.lastHead++;
00251                                                 r.conf = newGidList.size() / (double)suppBody;
00252                                                 rs.insert(rs.end(),r);
00253                                                 addHead(rs,threshold,maxHead,suppBody,r);
00254                                         }
00255                                 }
00256                         }
00257                 }
00258         }
00259
00260
00261          std::map<ItemType, BFSWithGidsAndCross::MapElement>
      BFSWithGidsAndCross::NewRule::hash_intersection( std::map<ItemType, MapElement>& lhs, const
      std::map<ItemType, MapElement>& rhs ) {
00262                 std::map<ItemType, MapElement> result;
00263
00264                 std::map<ItemType, MapElement>::iterator it;
00265                 for( it=lhs.begin(); it!=lhs.end(); ++it ) {
00266                         std::map<ItemType, MapElement>::const_iterator found = rhs.find(it->first);
00267                         if( found != rhs.end() ) {
00268                                 // GIDS INTERSECTION
00269                                 GidList newGidList;
00270                                 set_intersection(it->second.begin(),it->second.end(),
00271                                         found->second.begin(), found->second.end(),
00272                                                 inserter(newGidList,newGidList.begin()));
00273                                 result[it->first] = newGidList;
00274                         }
00275                 }
00276
00277                 return result;
00278         }
```

```
00279
00280
00281         void BFSWithGidsAndCross::BodyMap::insertRules( const NewRuleSet& rs, double totGroups ) {
00282                 NewRuleSet::const_iterator it;
00283                 for(it=rs.begin(); it!=rs.end(); it++) {
00284                         connection->insert( it->body, it->head, it->gids.size()/totGroups, it->conf );
00285                 }
00286         }
00287
00288
00289         void BFSWithGidsAndCross::prepareData() {
00290                 sourceTable = new SourceTable(*this);
00291                 ruleIterator = sourceTable->newIterator(SourceTable::FullIterator);
00292
00293                 // options.setTotGroups(sourceTable->getTotGroups());
00294         }
00295
00296
00297         void BFSWithGidsAndCross::mineRules() {
00298                 MRLogPush("Starting BFSWithGidsAndCross mining algorithm...");
00299
00300                 MRLog() « "Preparing data sources..." « std::endl;
00301                 prepareData();
00302                 Progress progress(200);
00303
00304                 float support = options.getSupport();
00305                 int maxBody = options.getBodyCardinalities().getMax();
00306                 int maxHead = options.getHeadCardinalities().getMax();
00307
00308                 ItemType gid1;
00309                 BodyMap bodyMap(connection, progress);
00310
00311                 int totalGroups = sourceTable->getTotGroups();
00312                 int howManyRows = 0;
00313                 int howManyGroups = 0;
00314
00315
00316                 MRLogPush("Reading data...");
00317
00318                 while (!ruleIterator.isAfterLast()) {
00319                         gid1 = ruleIterator->getGroup();
00320                         howManyGroups++;
00321
00322                         Transaction t1;
00323
00324                         t1.load(gid1,ruleIterator);
00325                         howManyRows += bodyMap.add(howManyGroups,t1);
00326                 }
00327
00328                 MRLog() « "Total groups: " « totalGroups « std::endl;
00329                 MRLogPop();
00330
00331                 MRLogPush("Starting rule extraction...");
00332
00333                 bodyMap.updateCount();
00334                 MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00335                 bodyMap.pruneMap(support*totalGroups);
00336                 MRLog() « "Total bodies after pruning: " « bodyMap.size() « std::endl;
00337                 NewRuleSet rs;
00338
00339                 progress.start();
00340                 int nrules = bodyMap.generateRules(support,totalGroups,maxBody,maxHead);
00341                 progress.end();
00342                 MRLog() « "After extracting rules, rules: " « nrules « std::endl;
00343
00344                 MRLogPop();
00345                 connection.finalize();
00346
00347                 MRLogPop();
00348         }
00349
00350
00351 } // namespace
```

7.209 /Users/esposito/Software/minerule/src/Algorithms/BFSWithGids↩ NoCross.cpp File Reference

```
#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <iostream>
#include <iterator>
#include <algorithm>
#include "minerule/Algorithms/AlgorithmsOptions.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/MRResultSet.hpp"
#include "minerule/Database/ItemType.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include "minerule/Database/Transaction.hpp"
```

Namespaces

- namespace minerule

7.210 BFSWithGidsNoCross.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 // CORE.CPP
00017
00018 #include<stdio.h>
00019 #include<string.h>
00020 #include<stdlib.h>
00021 #include <iostream>
00022 #include <iterator>
00023 #include <algorithm>
00024
00025
00026 #include "minerule/Algorithms/AlgorithmsOptions.hpp"
00027 #include "minerule/Utils/MineruleOptions.hpp"
00028 #include "minerule/Database/MRResultSet.hpp"
00029 #include "minerule/Database/ItemType.hpp"
00030 #include "minerule/Utils/Converter.hpp"
00031 #include "minerule/Database/Connection.hpp"
00032 #include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
00033 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00034 #include "minerule/Database/PrepareDataUtils.hpp"
00035 #include "minerule/Database/Transaction.hpp"
00036
00037 namespace minerule {
00038   bool BFSWithGidsNoCross::mineruleHasSameBodyHead = false;
00039
00040   void BFSWithGidsNoCross::BodyMapElement::insert(const ItemType& item, MapElement& gidList, bool
      secondPass) {
00041     std::map<ItemType, MapElement>::iterator found = heads.find(item);
00042     if (found == heads.end()) {
00043       if (!secondPass) heads[item] = gidList;
00044     } else found->second.insert(gidList.begin(),gidList.end());
00045   }
00046
```

```
00047   bool BFSWithGidsNoCross::BodyMapElement::pruneMap (float threshold) {
00048     std::map<ItemType, MapElement> newMap;
00049     for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++) {
00050       if (i->second.count >= threshold) {
00051                 i->second.count = 0;
00052                 newMap[i->first] = i->second;
00053         }
00054       }
00055     heads = newMap;
00056     return heads.size() > 0;
00057   }
00058
00059   bool BFSWithGidsNoCross::BodyMapElement::updateCount () {
00060     for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++) {
00061       i->second.count += i->second.size();
00062       }
00063     return heads.size() > 0;
00064   }
00065
00066
00067   int BFSWithGidsNoCross::BodyMap::add(int gid, Transaction& t1, Transaction& t2, bool secondPass) {
00068           int howMany = 0;
00069           MapElement me;
00070           me.insert(gid);
00071           for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00072                   std::map<ItemType, BodyMapElement>::iterator found = find(*i);
00073                   if (found == end()) {
00074                           if (secondPass) continue;
00075                           BodyMapElement bme;
00076                           (*this)[*i] = bme;
00077                           found = find(*i);
00078                   }
00079                   found->second.insert(gid);
00080                   for (Transaction::iterator j = t2.begin(); j != t2.end(); j++)
00081                   if (!BFSWithGidsNoCross::getMineruleHasSameBodyHead() || *i != *j /*&& i->price <
      j->price*/) {
00082                           howMany++;
00083                           found->second.insert(*j,me,secondPass);
00084                   }
00085           }
00086           return howMany;
00087   }
00088
00089   void BFSWithGidsNoCross::BodyMap::pruneMap (float threshold) {
00090           BodyMap newMap(*connection, *progress);
00091           for (iterator i = begin(); i != end(); i++) {
00092                   if (i->second.pruneMap(threshold))
00093                           newMap[i->first] = i->second;
00094           }
00095
00096           (*this) = newMap;
00097   }
00098
00099   void BFSWithGidsNoCross::BodyMap::updateCount () {
00100     for (iterator i = begin(); i != end(); i++)
00101       i->second.updateCount();
00102   }
00103
00104   void BFSWithGidsNoCross::BodyMap::createBodies (NewRuleSet& rs, float threshold, size_t maxBody) {
00105           for (iterator i = begin(); i != end(); i++) {
00106                   NewRule r(i,i->second);
00107                   rs.insert(rs.end(),r);
00108           }
00109
00110           for (unsigned int n = 0; n < rs.size(); n++) {
00111                   NewRuleSet::iterator i = rs.begin()+n;
00112                   NewRule& rc = *i;
00113
00114                   if (rc.body.size() < maxBody) {
00115
00116                           BodyMap::iterator lb = rc.lastBody;
00117                           for (BodyMap::iterator j = ++lb; j != end(); j++) {
00118                                   GidList newGidList;
00119                                   NewRule& rc2 = *(rs.begin()+n);
00120
00121                                   set_intersection(rc2.gids.begin(),rc2.gids.end(),
00122                                           j->second.begin(),
      j->second.end(),inserter(newGidList,newGidList.begin()));
00123
00124                                   if (newGidList.size() >= threshold) {
00125                                           NewRule r(rc2,j,newGidList);
00126                                           rs.insert(rs.end(),r);
00127                                   }
00128                           }
00129
00130                   } // if(rc.body.size())
00131
```

```
00132           } // for
00133   }
00134
00135
00136   void BFSWithGidsNoCross::BodyMap::createHeads (NewRuleSet& rs, NewRuleSet& rs1, float threshold,
        size_t maxHead) {
00137           for (size_t n = 0; n < rs.size(); n++) {
00138                   NewRule& rc = rs[n];
00139                   if (rc.head.size() < maxHead) {
00140                           map<ItemType, MapElement>::iterator lh = rc.lastHead;
00141                           map<ItemType, MapElement>::iterator eh = rc.lastBody->second.heads.end();
00142                           double supportBody = rc.gids.size();
00143
00144                           for (map<ItemType, MapElement>::iterator j = lh; j != eh; j++) {
00145                                   GidList newGidList;
00146                                   if (!BFSWithGidsNoCross::getMineruleHasSameBodyHead() ||
        std::find(rc.body.begin(), rc.body.end(), j->first) == rc.body.end()) {
00147
00148                                           set_intersection(rc.gids.begin(),rc.gids.end(),
00149                                                   j->second.begin(), j->second.end(),
        inserter(newGidList,newGidList.begin()));
00150
00151                                           if (newGidList.size() >= threshold) {
00152                                                   NewRule r(rc,j,newGidList);
00153                                                   r.lastHead++;
00154                                                   r.conf = newGidList.size() / supportBody;
00155                                                   rs1.insert(rs1.end(),r);
00156                                           }
00157                                   }
00158                           } // for
00159                   } // if
00160           } // for
00161   }
00162
00163   int BFSWithGidsNoCross::BodyMap::generateRules (float support, int totGroups, int maxBody, int
        maxHead) {
00164           NewRuleSet rs1;
00165           int howManyRules = 0;
00166           float threshold = support*totGroups;
00167           for (iterator i = begin(); i != end(); i++) {
00168       progress->tick();
00169
00170                   NewRule r(i,i->second);
00171                   rs1.insert(rs1.end(),r);
00172           }
00173           for (size_t n = 0; n < rs1.size(); n++) {
00174                   NewRule& rc = *(rs1.begin()+n);
00175                   NewRuleSet rs;
00176                   addHead(rs, threshold, maxHead, rc.gids.size(), rc);
00177                   howManyRules += rs.size();
00178                   insertRules(rs, totGroups);
00179                   if (rc.body.size() < (size_t)maxBody) {
00180                           BodyMap::iterator lb = rc.lastBody;
00181                           for (BodyMap::iterator j = ++lb; j != end(); j++) {
00182           progress->tick();
00183
00184                                   NewRule& rc2 = *(rs1.begin()+n);
00185                                   GidList newGidList;
00186                                   set_intersection(rc2.gids.begin(),rc2.gids.end(),
00187                                           j->second.begin(), j->second.end(),
00188                                                   inserter(newGidList,newGidList.begin()));
00189                                   if (newGidList.size() >= threshold) {
00190                                           NewRule r(rc2,j,newGidList);
00191                                           rs1.insert(rs1.end(),r);
00192                                   }
00193                           }
00194                   }
00195           }
00196           return howManyRules;
00197   }
00198
00199   void BFSWithGidsNoCross::BodyMap::addHead (NewRuleSet& rs, float threshold, int maxHead, int
        suppBody, NewRule& rc) {
00200     if (rc.head.size() < (size_t)maxHead) {
00201       std::map<ItemType, MapElement>::iterator lh = rc.lastHead;
00202       std::map<ItemType, MapElement>::iterator eh = rc.lastBody->second.heads.end();
00203       for (map<ItemType, MapElement>::iterator j = lh; j != eh; j++) {
00204                 GidList newGidList;
00205
00206                 if (!BFSWithGidsNoCross::getMineruleHasSameBodyHead() ||
00207                         std::find(rc.body.begin(), rc.body.end(), j->first) == rc.body.end()
00208                         /* && rc.head.find(j->first) == rc.head.end()*/) {
00209                   set_intersection(rc.gids.begin(),rc.gids.end(),
00210                                    j->second.begin(), j->second.end(),
00211                                    inserter(newGidList,newGidList.begin()));
00212
00213                   if (newGidList.size() >= threshold) {
```

```
00214                   NewRule r(rc,j,newGidList);
00215                   r.lastHead++;
00216                   r.conf = newGidList.size() / (double)suppBody;
00217                   rs.insert(rs.end(),r);
00218                   addHead(rs,threshold,maxHead,suppBody,r);
00219                 }
00220               } // if (std::find
00221     } // for( map...
00222   } // if (rc.head.size...
00223 }
00224
00225 void BFSWithGidsNoCross::BodyMap::insertRules( const NewRuleSet& rs,
00226                         double totGroups ) {
00227   NewRuleSet::const_iterator it;
00228   for(it=rs.begin(); it!=rs.end(); it++) {
00229     connection->insert( it->body, it->head, it->gids.size()/totGroups, it->conf );
00230   }
00231 }
00232
00233
00234 void BFSWithGidsNoCross::prepareData() {
00235         sourceTable = new SourceTable(*this);
00236         bodyIterator = sourceTable->newIterator(SourceTable::BodyIterator);
00237         headIterator = sourceTable->newIterator(SourceTable::HeadIterator);
00238
00239         // options.setTotGroups(sourceTable->getTotGroups());
00240 }
00241
00242
00243 void BFSWithGidsNoCross::mineRules() {
00244   MRLogPush("Starting BFSWithGidsNoCross mining algorithm...");
00245
00246   MRLog() « "Preparing data sources..." « std::endl;
00247   prepareData();
00248               Progress progress(200);
00249
00250   float support = options.getSupport();
00251   int maxBody = options.getBodyCardinalities().getMax();
00252   int maxHead = options.getHeadCardinalities().getMax();
00253
00254   ItemType gid1;
00255
00256   if(bodyIterator.isAfterLast())
00257               throw new MineruleException(MR_ERROR_INTERNAL,"Cannot find initial GID for body
      elements");
00258   if(headIterator.isAfterLast())
00259               throw new MineruleException(MR_ERROR_INTERNAL,"Cannot find initial GID for head
      elements");
00260
00261       MRLogPush("Reading data");
00262   BodyMap bodyMap(connection, progress);
00263
00264   int totalGroups = sourceTable->getTotGroups();
00265   int howManyRows = 0;
00266   int howManyGroups = 0;
00267
00268               Progress readProgress(10000);
00269               readProgress.start();
00270   while (!bodyIterator.isAfterLast()) {
00271     ItemType gid = bodyIterator->getGroup();
00272
00273         Transaction t1, t2;
00274     t1.loadBody(gid,bodyIterator);
00275
00276     bool found2 = t2.findGid(gid,headIterator);
00277     if (found2) {
00278               t2.loadHead(gid,headIterator);
00279     }
00280
00281     howManyRows += bodyMap.add(howManyGroups,t1,t2);
00282     howManyGroups++;
00283
00284                     readProgress.tick();
00285   }
00286               readProgress.end();
00287
00288   MRLog() « "Total groups: " « totalGroups « std::endl;
00289       MRLogPop();
00290
00291   MRLogPush("Starting rule extraction...");
00292
00293   bodyMap.updateCount();
00294
00295   MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00296   bodyMap.pruneMap(support*totalGroups);
00297   MRLog() « "Total bodies after pruning: " « bodyMap.size() « std::endl;
00298   NewRuleSet rs;
```

```
00299
00300                 progress.start();
00301     int nrules = bodyMap.generateRules(support,totalGroups,maxBody,maxHead);
00302                 progress.end();
00303
00304     MRLog() « "After extracting rules, rules: " « nrules « std::endl;
00305
00306     MRLogPop();
00307
00308     MRLogPop();
00309
00310     connection.finalize();
00311   }
00312
00313
00314 } // namespace
```

7.211 /Users/esposito/Software/minerule/src/Algorithms/Bodymap.cpp File Reference

```
#include "minerule/Algorithms/Bodymap.hpp"
#include "minerule/Utils/Constraints.hpp"
#include <algorithm>
```

Namespaces

- namespace minerule

7.212 Bodymap.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 // CORE.CPP
00017
00018 #include "minerule/Algorithms/Bodymap.hpp"
00019 #include "minerule/Utils/Constraints.hpp"
00020 #include <algorithm>
00021
00022 namespace minerule {
00023
00024 int BodyMap::nextid = 0;
00025
00026 void BodyMapElement::insert(const ItemType& item, const int gid, bool secondPass) {
00027         std::map<ItemType, MapElement>::iterator found = heads.find(item);
00028         if (found == heads.end()) {
00029                 if (!secondPass) heads[item].insert(gid);
00030         } else found->second.insert(gid);
00031 }
00032
00033 bool BodyMapElement::pruneMap (double threshold, bool onlyBody) {
00034         if (!moreThan(threshold)) return false;
00035         else if (onlyBody) return true;
00036         std::map<ItemType, MapElement> newMap;
```

```
00037         for (std::map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++)
00038                 if (i->second.counter >= threshold) {i->second.counter = 0; newMap[i->first] =
       i->second;}
00039         heads = newMap;
00040         return heads.size() > 0;
00041 }
00042
00043 bool BodyMapElement::updateCount () {
00044 //      if (size() < threshold) return false;
00045 //      map<ItemType, MapElement> newMap;
00046 //      for (map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++)
00047 //{
00048 //              i->second.counter += i->second.count(true);
00049 //}
00050 //              if (i->second.size() >= threshold) newMap[i->first] = i->second;
00051 //      heads = newMap;
00052         counter += count(true);
00053         return heads.size() > 0;
00054 }
00055
00056 /*
00057 void BodyMapElement::saveMap (std::ostream& out, bool withGids) {
00058         out « ' ' « heads.size();
00059         for (map<ItemType, MapElement>::iterator i = heads.begin(); i != heads.end(); i++) {
00060                 out « ' ' « i->first;
00061                 out « ' ' « i->second.size();
00062         if (withGids) {
00063                 out  « ' ';
00064                 copy(i->second.begin(), i->second.end(), std::ostream_iterator<ItemType>(out, " "));
00065         }
00066                 i->second.erase(i->second.begin(), i->second.end());
00067         }
00068         out « std::endl;
00069 }
00070 */
00071
00072 /*
00073 void BodyMapElement::loadMap (istream& in, bool withGids) {
00074         int n,m; in » n;
00075         ItemType item, gid;
00076         for (int i=0; i<n; i++) {
00077                 in » item;
00078                 in » m;
00079                 heads[item].counter += m;
00080                 if (withGids) {
00081                 in » m;
00082                 for (int j=0; j<m; j++) { in » gid; heads[item].insert(gid); }
00083                 } //else { me = heads[item]; }
00084         }
00085 }
00086 */
00087
00088 void BodyMapElement::setMinMax (int which, int v) {
00089         MinMax mm;
00090         for (int i=attribute.size(); i<=which; i++) {
00091                 attribute.insert(attribute.end(),mm);
00092         }
00093         if (attribute[which].minValue() > v) attribute[which].min = v;
00094         if (attribute[which].maxValue() < v) attribute[which].max = v;
00095 }
00096
00097 int BodyMap::add(Transaction& t1, const int gid) {
00098         int howMany = 0;
00099         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00100                 map<ItemType, BodyMapElement>::iterator found = find(*i);
00101                 if (found == end()) {
00102                         BodyMapElement bme;
00103 //                      bme.insert(gid);
00104                         (*this)[*i] = bme;
00105                         found = find(*i);
00106                 }
00107                 found->second.insert(gid);
00108                 int n = t1.values.size()/t1.size();
00109                 for (int k=0; k < n; k++) {
00110                         found->second.setMinMax(k,t1.values[k+howMany*n]);
00111                 }
00112                 howMany++;
00113         }
00114         return howMany;
00115 }
00116
00117 int BodyMap::add(const int gid, Transaction& t1, Transaction& t2, bool secondPass) {
00118         int howMany = 0;
00119         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00120                 map<ItemType, BodyMapElement>::iterator found = find(*i);
00121                 if (found == end()) {
00122                         if (secondPass) continue;
```

```
00123                         BodyMapElement bme;
00124 //                      bme.insert(gid);
00125                         (*this)[*i] = bme;
00126                         found = find(*i);
00127                 }
00128                 found->second.insert(gid);
00129                 for (Transaction::iterator j = t2.begin(); j != t2.end(); j++)
00130                         if (*i != *j /*&& i->price < j->price*/) {
00131                                 howMany++;
00132                                 found->second.insert(*j,gid,false);
00133 //                              found->second.insert(*j,me,secondPass);
00134                         }
00135         }
00136         return howMany;
00137 }
00138
00139 /*
00140 int BodyMap::add(const int gid, ItemType& t1, ItemType& t2, bool secondPass) {
00141         int howMany = 0;
00142         map<ItemType, BodyMapElement>::iterator found = find(t1);
00143         if (found == end() && !secondPass) {
00144                 BodyMapElement bme;
00145 //              bme.insert(gid);
00146                 (*this)[t1] = bme;
00147                 found = find(t1);
00148         }
00149         found->second.insert(gid);
00150         if (t1 != t2) {
00151                 howMany++;
00152                 found->second.insert(t2,gid,secondPass);
00153         }
00154         return howMany;
00155 }
00156
00157 int BodyMap::add(ItemType& gid, Transaction& t1) {
00158         int howMany = 0;
00159         BodyMapElement bme;
00160 //      bme.insert(gid);
00161         (*this)[gid] = bme;
00162         map<ItemType, BodyMapElement>::iterator found = find(gid);
00163         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00164                 found->second.insert(*i);
00165                 howMany++;
00166         }
00167         return howMany;
00168 }
00169 int BodyMap::add(const int gid, Transaction& t1, bool secondPass) {
00170         int howMany = 0;
00171         ItemType prev;
00172         map<ItemType, BodyMapElement>::iterator found;
00173         for (Transaction::iterator i = t1.begin(); i != t1.end(); i++) {
00174                 if (prev != *i) {
00175                         found = find(*i);
00176                         if (found == end()) {
00177                                 if (secondPass) continue;
00178                                 BodyMapElement bme;
00179 //                      bme.insert(gid);
00180                                 (*this)[*i] = bme;
00181                                 found = find(*i);
00182                         }
00183                         prev = *i;
00184                         found->second.insert(gid);
00185                 }
00186                 if (*i != *i) {
00187                         howMany++;
00188                         found->second.insert(*i,gid,secondPass);
00189                 }
00190         }
00191         return howMany;
00192 }
00193 */
00194
00195 /*
00196 void BodyMap::saveMap (std::ostream& out, bool withGids) {
00197         for (iterator i = begin(); i != end(); i++) {
00198                 out « i->first;
00199                 i->second.saveMap(out, withGids);
00200         }
00201 }
00202 */
00203 /*
00204 void BodyMap::loadMap (istream& in, bool withGids) {
00205         ItemType item;
00206         while (!in.eof()) {
00207                 in » item;
00208                 if (in.eof()) break;
00209                 (*this)[item].loadMap(in, withGids);
```

```
00210         }
00211 }
00212 */
00213
00214 void BodyMap::pruneMap (double threshold, bool onlyBody) {
00215         //BodyMap newMap(outfile, totGroups);
00216         for (iterator i = begin(); i != end(); i++)
00217                 if (!i->second.pruneMap(threshold,onlyBody)) {
00218                 //       newMap[i->first] = i->second;
00219                         i->second.done = true;
00220                 } else i->second.done = false;
00221         //(*this) = newMap;
00222 }
00223
00224 void BodyMap::updateCount () {
00225         for (iterator i = begin(); i != end(); i++)
00226                 i->second.updateCount();
00227 }
00228
00229 int BodyMap::buildRules (NewRuleSet& rs1, NewRule& rc, std::vector<BodyMap::iterator>& vb,
       std::vector<BodyMap::iterator>& vh, int maxBodyCard, int maxHeadCard, float threshold, int j) {
00230  std::vector<BodyMap::iterator> newVb, newVh;
00231         NewRuleSet rs;
00232         int start = rs.size();
00233         bool intersect = (rc.gids.size() > 0 );
00234         for (size_t k=j+1; k<vb.size(); k++) {
00235                 //GidList newGidList;
00236                 GidList newGidList(vb[k]->second);
00237                 if (intersect)
00238                         newGidList &= rc.gids;
00239 //              else newGidList = v[k]->second;
00240                 if (newGidList.moreThan(threshold)) {
00241                         NewRule r(rc,vb[k],newGidList,newVb.size());
00242                         rs.insert(rs.end(),r);
00243                         newVb.insert(newVb.end(),vb[k]);
00244                 }
00245         }
00246         int nrules = rs.size();
00247         int howManyRules = 0;
00248         for (int i=start; i<nrules; i++) {
00249             if (i<nrules) {
00250                 NewRuleSet rs2;
00251 //              curBodySupp = rs[i].gids.count(true);
00252                 newVh = buildHead(rs2,rs[i],vh,maxHeadCard,threshold,-1);
00253                 howManyRules += rs2.size();
00254                 saveRules(rs2, rs[i].gids.count());
00255              }
00256                 if (newVb.size() > 0 && maxBodyCard > 1) {
00257                         howManyRules += buildRules(rs1,
       rs[i],newVb,newVh,maxBodyCard-1,maxHeadCard,threshold,rs[i].bodySupp);
00258                 }
00259         }
00260         return howManyRules;
00261 }
00262  std::vector<BodyMap::iterator> BodyMap::buildHead (NewRuleSet& rs, NewRule rc,
       std::vector<BodyMap::iterator>& v, int maxCard, float threshold, int j) {
00263  std::vector<BodyMap::iterator> newV;
00264 //      NewRuleSet rs;
00265         int start = rs.size();
00266         for (size_t k=j+1; k<v.size(); k++)// {
00267                 //GidList newGidList;
00268         if (std::find(rc.body.begin(), rc.body.end(),v[k]->first) == rc.body.end()) {
00269                 GidList newGidList(v[k]->second);
00270         //      if (rc.gids.count(true) > 0 )
00271                         newGidList &= rc.gids;
00272 //              else newGidList = v[k]->second;
00273                 if (newGidList.moreThan(threshold)) {
00274                         NewRule r(rc,v[k],newV.size(),newGidList);
00275                         rs.insert(rs.end(),r);
00276                         newV.insert(newV.end(),v[k]);
00277                 }
00278         }
00279         int nrules = rs.size();
00280         if (newV.size() > 0 && 1 < maxCard) {
00281                 for (int i=start; i<nrules; i++)
00282                         buildHead(rs,rs[i],newV,maxCard-1,threshold,rs[i].bodySupp);
00283         }
00284         return newV;
00285 }
00286
00287 bool BodyMap::checkAntiMono (NewRule& rc) {
00288         bool flag = true;
00289         for (size_t i=0; i<antiMonoConstr.size() && flag; i++)
00290                 flag = flag && antiMonoConstr[i]->check(*this,rc.body);
00291         return flag;
00292 }
00293
```

```
00294 bool BodyMap::checkMono (NewRule& rc) {
00295         bool flag = true;
00296         for (size_t i=0; i<monoConstr.size() && flag; i++)
00297                 flag = flag && monoConstr[i]->check(*this,rc.body);
00298         return flag;
00299 }
00300
00301 int BodyMap::buildItemset (NewRuleSet& rs1, NewRule& rc, std::vector<BodyMap::iterator>& v, int
       maxCard, float threshold, int j) {
00302  std::vector<BodyMap::iterator> newV;
00303         NewRuleSet rs;
00304         int start = rs.size();
00305         bool intersect = rc.gids.size() > 0;
00306         int howManyRules = 0;
00307         size_t l = rc.body.size()+1;
00308         for (size_t k=j+1; k<v.size(); k++) {
00309                 //GidList newGidList;
00310                 //GidList newGidList(v[k]->second);
00311                 NewRule r(rc,v[k],v[k]->second,newV.size());
00312                 if (!checkAntiMono(r)) continue;
00313                 r.satisfy = r.satisfy || checkMono(r);
00314                 if (intersect)
00315                         r.gids &= rc.gids;
00316 //              else newGidList = v[k]->second;
00317                 if (r.gids.moreThan(threshold)) {
00318                         //NewRule r(rc,v[k],newGidList,newV.size());
00319                         if(r.satisfy) howManyRules ++;
00320                         rs.insert(rs.end(),r);
00321                         newV.insert(newV.end(),v[k]);
00322                 }
00323         }
00324         int nrules = rs.size();
00325         while (l >= itemsets.size()) itemsets.insert(itemsets.end(),0);
00326         itemsets[l] += nrules;
00327         saveItemsets(rs, -1);
00328         if (newV.size() > 0 && 1 < maxCard) {
00329                 for (int i=start; i<nrules; i++) {
00330                         //cout « rs[i];
00331                         howManyRules +=
       buildItemset(rs,rs[i],newV,maxCard-1,threshold,rs[i].bodySupp);
00332                 }
00333         }
00334         return howManyRules;
00335 }
00336
00337 int BodyMap::generateStartItemSets (NewRuleSet& rs1, NewRule& rc, std::vector<BodyMap::iterator>& v,
       int maxCard, float threshold, int j) {
00338  std::vector<BodyMap::iterator> newV;
00339         NewRuleSet rs;
00340         //      int start = rs.size();
00341         bool intersect = rc.gids.size() > 0;
00342         int howManyRules = 0;
00343         size_t l = rc.body.size()+1;
00344         while (l >= itemsets.size()) itemsets.insert(itemsets.end(),0);
00345         size_t k=j+1;
00346         if (k<v.size()) {
00347                 //GidList newGidList;
00348                 //GidList newGidList(v[k]->second);
00349                 NewRule r(rc,v[k],v[k]->second,k);
00350                 if (checkAntiMono(r)) {
00351                 r.satisfy = r.satisfy || checkMono(r);
00352                 if (intersect)
00353                         r.gids &= rc.gids;
00354 //              else newGidList = v[k]->second;
00355                 if (r.gids.moreThan(threshold)) {
00356                         //NewRule r(rc,v[k],newGidList,newV.size());
00357                         if(r.satisfy) {
00358                         //      rs.insert(rs.end(),r);
00359                                 //cout « r;
00360                                 saveItemset(r, -1);
00361                                 howManyRules += buildItemset(rs,r,v,maxCard-1,threshold,r.bodySupp) +
       1;
00362                                 itemsets[l] += 1;
00363                                 //howManyRules += generateStartItemSets(rs,rc,v,maxCard,threshold,k);
00364                         } else {
00365                                 howManyRules += generateStartItemSets(rs,r,v,maxCard-1,threshold,k);
00366                                 if (howManyRules == 0) return howManyRules;
00367                         }
00368                 }
00369                 }
00370                 howManyRules += generateStartItemSets(rs,rc,v,maxCard,threshold,k);
00371         }
00372         return howManyRules;
00373 }
00374
00375 int BodyMap::buildRules (NewRuleSet& rs2, NewRule rc, BodyMap::iterator p, float threshold, int
       maxBody, int maxHead) {
```

```
00376         int howManyRules = 0;
00377
00378         bool intersect = rc.gids.size() > 0;
00379         while (p != end()) {
00380                 GidList newGidList(p->second);
00381                 if (intersect)
00382                         newGidList &= rc.gids;
00383                 //else newGidList = p->second;
00384                 if (newGidList.moreThan(threshold)) {
00385                         NewRule r(rc,p,newGidList);
00386                         howManyRules += buildHead(rs2, threshold, maxHead, r.gids.count(true), r,
      p->second.heads.begin());
00387                         p++;
00388                 if (r.body.size() < (size_t)maxBody) {
00389                         howManyRules += buildRules(rs2, r, p, threshold, maxBody, maxHead);
00390                 }
00391                 } else p++;
00392         }
00393         return howManyRules;
00394 }
00395
00396 int BodyMap::buildHead (NewRuleSet& rs, float threshold, int maxHead, int suppBody, NewRule& rc,
      std::map<ItemType, MapElement>::iterator j) {
00397         int nrules = 0;
00398         if (rc.head.size() < (size_t)maxHead) {
00399                 map<ItemType, MapElement>::iterator eh = rc.lastBody->second.heads.end();
00400                 while (j != eh) {
00401 //cout « "Extending head of " « *i « " with " « j->first  « " size " « j->second.size() « std::endl;
00402                 if (std::find(rc.body.begin(), rc.body.end(),j->first) == rc.body.end()) {
00403                         GidList newGidList(rc.gids);
00404                         newGidList &= j->second;
00405                                 if (newGidList.moreThan(threshold)) {
00406                                         NewRule r(rc,j,newGidList);
00407                                         rs.insert(rs.end(),r);
00408                                         nrules += 1;
00409                                         nrules += buildHead(rs,threshold,maxHead,suppBody,r,++j);
00410                         } else j++;
00411                 } else j++;
00412         }
00413         }
00414         return nrules;
00415 }
00416
00417 void BodyMap::howManyItemsets () {
00418         for (size_t i = 1; i < itemsets.size(); i++) {
00419                 std::cout « "Itemset Length: " « i « " -> " « itemsets[i] « std::endl;
00420         }
00421 }
00422
00423 void BodyMap::saveItemset( const NewRule& r,
00424                            double bodySupp ) {
00425     int c = r.gids.count();
00426     connection->insert( r.body,
00427                         r.head,
00428                         c/totGroups,
00429                         1, true );
00430 }
00431
00432 void BodyMap::saveRules( const NewRuleSet& rs,
00433                            double bodySupp ) {
00434     NewRuleSet::const_iterator it = rs.begin();
00435     bool bodyID = true;
00436     int c;
00437     for(it=rs.begin(); it!=rs.end(); it++) {
00438       c = it->gids.count();
00439       connection->insert( it->body,
00440                           it->head,
00441                           c/totGroups,
00442                           c/bodySupp, bodyID );
00443 //      bodyID = false;
00444     }
00445   }
00446
00447
00448 void BodyMap::saveItemsets( const NewRuleSet& rs,
00449                            double bodySupp ) {
00450 //    static ofstream out(outfile.c_str());
00451     NewRuleSet::const_iterator it = rs.begin();
00452     int c;
00453     for(it=rs.begin(); it!=rs.end(); it++) {
00454         c = it->gids.count();
00455         connection->insert( it->body,
00456                             it->head,
00457                             c/totGroups,
00458                             1, true );
00459     }
00460   }
```

```
00461
00462 } //end namespace
```

7.213 /Users/esposito/Software/minerule/src/Algorithms/Care.cpp File Reference

```
#include "minerule/Algorithms/Care.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
```

Namespaces

- namespace minerule

Macros

- #define MROPTIONFILE "mroptions"

7.213.1 Macro Definition Documentation

7.213.1.1 MROPTIONFILE

```
#define MROPTIONFILE "mroptions"
```

Definition at line 24 of file Care.cpp.

7.214 Care.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Marco Botta (botta@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017 // care.cpp
00018
00019 #include "minerule/Algorithms/Care.hpp"
00020
00021 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00022 #include "minerule/Database/PrepareDataUtils.hpp"
```

```
00023
00024 #define MROPTIONFILE "mroptions"
00025
00026 namespace minerule {
00027
00028   // Preprocessing functions
00029
00030         size_t CARE::buildAttrStr(const ParsedMinerule::AttrVector& attr,
00031                 size_t startIndex,
00032                         std::string& attrStr,
00033         std::vector<int>& des) const {
00034                 ParsedMinerule::AttrVector::const_iterator it = attr.begin();
00035                 for( ; it!=attr.end(); it++ ) {
00036                         if(it!=attr.begin()) {
00037                                 attrStr+=",";
00038                         }
00039
00040                         attrStr+=*it;
00041                         des.push_back(++startIndex);
00042                 }
00043
00044                 return startIndex;
00045         }
00046
00047         std::string CARE::buildQry( const std::string& groupAttrStr,
00048                 const std::string& attrStr,
00049         const std::string& constraints) const {
00050
00051                 return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00052                         "FROM "+minerule.getParsedMinerule().tab_source+" "+
00053                                 (constraints.size()>0 ?
00054                                         "WHERE "+constraints+" " :
00055                 "")
00056                         +"ORDER BY "+groupAttrStr+","+attrStr);
00057         }
00058
00059         void CARE::prepareData() {
00060                 sourceTable = new SourceTable(*this);
00061 //              options.setTotGroups(sourceTable->getTotGroups());
00062
00063                 bodyIterator = sourceTable->newIterator(SourceTable::BodyIterator);
00064                 headIterator = sourceTable->newIterator(SourceTable::HeadIterator);
00065         }
00066
00067         void CARE::mineRules() {
00068                 MRLogPush("Starting CARE mining algorithm...");
00069
00070                 MRLog() << "Preparing data sources..." << std::endl;
00071                 prepareData();
00072
00073                 double support = options.getSupport();
00074                 int maxBody = options.getBodyCardinalities().getMax();
00075                 int maxHead = options.getHeadCardinalities().getMax();
00076
00077                 ItemType gid1;
00078
00079                 BodyMap bodyMap(connection,1);
00080                 BodyMap headMap(connection,1);
00081
00082                 int howManyGroups = 0;
00083                 int totalGroups = sourceTable->getTotGroups();
00084                 int howManyRows = 0;
00085
00086                 while (!bodyIterator.isAfterLast()) {
00087                         ItemType gid = bodyIterator->getGroup();
00088                         Transaction t1, t2;
00089
00090                         t1.loadBody(gid,bodyIterator);
00091                         bool found2 = t2.findGid(gid,headIterator);
00092                         if (found2) {
00093                                 t2.loadHead(gid,headIterator);
00094                         }
00095                         howManyRows += bodyMap.add(t1,howManyGroups);
00096                         howManyRows += headMap.add(t2,howManyGroups);
00097                         howManyGroups++;
00098                 }
00099
00100                 MRLog() << "Total rows: " << howManyRows << std::endl;
00101                 MRLog() << "Total groups: " << totalGroups << std::endl;
00102                 MRLogPop();
00103
00104                 MRLogPush("Starting rule extraction...");
00105
00106                 bodyMap.updateCount();
00107                 headMap.updateCount();
00108                 MRLog() << "Total bodies before pruning: " << bodyMap.size() << std::endl;
00109   //  MRLogPop();
```

```
00110 //    MRLogPush("Starting pruning ...");
00111                 bodyMap.pruneMap(support*totalGroups,true);
00112                 headMap.pruneMap(support*totalGroups,true);
00113 //    MRLogPop();
00114                 bodyMap.setTotalGroups(totalGroups);
00115 //  MRLogPush("Starting rule extraction ...");
00116                 std::vector<BodyMap::iterator> v;
00117                 std::vector<BodyMap::iterator> v1;
00118                 std::multimap<int, BodyMap::iterator> temp;
00119                 for (BodyMap::iterator i = bodyMap.begin(); i!=bodyMap.end(); i++)
00120                         if (!i->second.done) temp.insert(std::pair<int,
      BodyMap::iterator>(i->second.count(),i));
00121                 for (std::multimap<int, BodyMap::iterator>::iterator i = temp.begin(); i!=temp.end();
      i++)
00122                         v.insert(v.end(),i->second);
00123                 for (BodyMap::iterator i = headMap.begin(); i!=headMap.end(); i++)
00124                         v1.insert(v1.end(),i);
00125                 MRLog() « "Total bodies after pruning: " « v.size() « std::endl;
00126
00127                 NewRule r;
00128                 NewRuleSet rs1;
00129     //bodyMap.openOutputFiles();
00130                 int nrules = bodyMap.buildRules(rs1,r,v,v1,maxBody,maxHead,support*totalGroups,-1);
00131     //bodyMap.closeOutputFiles();
00132                 MRLog() « "After extracting rules: " « nrules « std::endl;
00133                 MRLogPop();
00134                 connection.finalize();
00135         }
00136
00137 } //end namespace
```

7.215 /Users/esposito/Software/minerule/src/Algorithms/CCSMiner.cpp File Reference

```
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>
#include <string.h>
#include <assert.h>
#include <set>
#include <time.h>
#include "minerule/Algorithms/BitVector.hpp"
#include "minerule/Algorithms/CCSMiner.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
```

Namespaces

- namespace minerule

7.216 CCSMiner.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <string>
00018 #include <fstream>
00019 #include <algorithm>
00020 #include <string.h>
00021 #include <assert.h>
00022 #include <set>
00023 #include <time.h>
00024 #include "minerule/Algorithms/BitVector.hpp"
00025 #include "minerule/Algorithms/CCSMiner.hpp"
00026 #include "minerule/Database/PrepareDataUtils.hpp"
00027
00028 namespace minerule {
00029
00030         void
00031         CCSMiner::prepareData() {
00032                 PrepareDataUtils pdu(minerule.getParsedMinerule(), this->sourceTableRequirements());
00033                 const ParsedMinerule& pm = minerule.getParsedMinerule();
00034                 std::string groupAttrList = pdu.buildAttrListDescription(pm.ga);
00035                 std::string ordAttrList = pdu.buildAttrListDescription(pm.oa);
00036
00037                 std::string sqlQuery =
00038                         "SELECT "+ groupAttrList+","
00039                         + pdu.buildAttrListDescription(pm.ba) + ","
00040                         + ordAttrList + " "
00041                         + "FROM "+ pm.tab_source + " "
00042                         + "ORDER BY "+groupAttrList+","+ordAttrList;
00043
00044                 //      std::cout « sqlQuery « std::endl;
00045
00046                 size_t last_elem;
00047
00048                 last_elem = rowDes.setGroupElems(1,pm.ga.size());
00049
00050                 rowDes.setBodyElems(last_elem+1, pm.ba.size());
00051
00052                 MRDebug() « "FSMiner query:" « sqlQuery.c_str() « std::endl;
00053
00054                 statement = connection.getMRDBConnection()->prepareStatement(sqlQuery.c_str());
00055         }
00056
00057         bool CCSMiner::find(std::vector<CCSMSequence*>* vec, CCSMSequence* elem){
00058          std::vector<CCSMSequence*>::iterator it;
00059                 bool trovato = false;
00060                 for (it=vec->begin();it!=vec->end()&&!trovato;++it){
00061                         if ((*(*it)) == *elem)
00062                                 trovato =true;
00063                 }
00064                 return trovato;
00065         }
00066
00067
00068
00069         void CCSMiner::combina(std::vector<CCSMSequence::ResultItems>& result,
      std::vector<CCSMSequence*>* k2, size_t k, int min_g, int max_g, double threshold, int check){
00070                 size_t seq_length=0;
00071                 time_t init1, end1, init, end;
00072
00073          std::vector<size_t> pos_canc;
00074          std::vector<CCSMSequence*>* rif_singleton= k2;
00075          std::vector<CCSMSequence*>* k_rif;
00076
00077
00078          std::vector<CCSMSequence*>* k_2=k2;
00079          std::vector<CCSMSequence*>* k_1=NULL;
00080
00081                 std::list<CCSMSequence*> suff;
00082                 std::list<CCSMSequence*> pre;
00083                 std::list<CCSMSequence*>::iterator pre_it;
00084                 std::list<CCSMSequence*>::iterator suf_it;
00085          std::vector<CCSMSequence*>* da_cancellare = new std::vector<CCSMSequence*>() ;
00086                 bool canContinue=true;
00087                 while (seq_length!=k&&canContinue){
00088                         time(&init1);
00089                         //utilizzato per la fase di transaction reduction, mi memorizza le seq che
      sono di livello k e da cui al passo
00090                         //successivo andro a generare le seq di livello k+1
00091
00092                         k_rif = new std::vector<CCSMSequence*>();
00093
00094
00095                         k_1 = new std::vector<CCSMSequence*>();
```

```
00096                         canContinue=false;
00097                         int cont=0;
00098                         for (size_t i=0;i<k_2->size();++i){
00099                                 //partiziono lo spazio utilizzando le sequenze che hanno (*k_2)[i]
      come suffisso e come prefisso
00100                                 //(quelle di lunghezza k-1)e le combino per ottenere le sequenze di
      lunghezza k
00101                                 suff = (*k_2)[i]->getSuffixSequence();
00102                                 pre =  (*k_2)[i]->getPrefixSequence();
00103
00104                                 for (pre_it=pre.begin();pre_it!=pre.end();++pre_it){
00105                                         for (suf_it=suff.begin();suf_it!=suff.end();++suf_it){

00106                                                 time(&init1);
00107                                                 CCSMSequence* dino =
00108
      CCSMSequence::merge((*suf_it),(*pre_it)->getLastItem(),min_g,max_g,threshold);
00109                                                 time(&end1);
00110 /*                                  //              std::cout<<"tempo di generazione
      "<<difftime(end1,init1)<<std::endl;
00111                                   //              std::cout<<(*suf_it)->toStdString()<<" + "<<std::endl<<"
      "<<(*pre_it)->toStdString();
00112                                   //              std::cout<<" = "<<std::endl;
00113                                                   std::cout<<"
      "<<dino->toStdString()<<" con conteggio "<<dino->getCount()<<std::endl;
00114 */
00115                                                   if (dino->getCount()>=threshold){
00116                                                           //inserisco i punatori alle sequenze
      di lunghezza k
00117                                                           //nelle Prefix/Suffix list degli
      elementi k-1
00118
00119                                                   k_rif->push_back(dino);//utilizzato per la
      fase di transaction reduction
00120
00121                                                   (*suf_it)->addPrefixSequence(dino);

00122                                                   (*pre_it)->addSuffixSequence(dino);
00123                                                           //cout<<"sequenza
      "<<dino->toStdString()<<std::endl;
00124                                                   result.push_back(dino->getSequenceItems());
00125 //                                                        std::cout<<"OK"<<std::endl;
00126                                                   }else
00127                                                           delete dino;
00128                                           }
00129                                   }
00130                           }
00131                         //controllo se al passo successivo e ancora possibile generare dei possibili
      candidati
00132                         //inserisco nell'insieme k-1 gli elementi che appartengono alla lista dei
      suffissi o dei prefissi
00133                         //delle sequenze degli elementi di k-2
00134
00135
00136                                 for (size_t i=0;i<k_2->size();++i){
00137
00138                                         pre =  (*k_2)[i]->getPrefixSequence();
00139
00140                                         for (pre_it=pre.begin();pre_it!=pre.end();++pre_it){
00141                                         //canGenerate testa se per quel determinato elemento c'e
      almeno un elemento nella lista dei suffissi
00142                                         //ed uno in quella dei prefissi, in modo che possa generare
      una possibile sequenza frequente di livello k
00143                                         //cout<<(*pre_it)->toStdString()<<" ";
00144                                                 if ((*pre_it)->canGenerate()){
00145                                                 //cout<<"puo generare"<<std::endl;
00146                                                         bool trovato=find(k_1,(*pre_it));
00147                                                         if (!trovato)
00148                                                                 k_1->push_back((*pre_it));
00149                                                         cont++;
00150                                                 //se ci sono sequenze che possono generare seq di
      livello k allora si puo continuare nel ciclo
00151                                                         canContinue=true;
00152                                                         }//se la sequenza non puo essere usata per
      generare ulteriori sequenze allora
00153                                         // viene inserita in un vettore di seq da cancellare
00154                                                         else{
00155                                                                 bool trovato =
      find(da_cancellare,(*pre_it));
00156                                                                 if (!trovato)
00157
      da_cancellare->push_back((*pre_it));
00158                                                 //cout<<"inserisco in da_cancellare"<<std::endl;
00159
00160                                                         }
00161                                                 }
00162                                                 suff = (*k_2)[i]->getSuffixSequence();
```

```
00163
00164                                                   for (suf_it=suff.begin();suf_it!=suff.end();++suf_it){
00165                                           //cout«(*suf_it)->toStdString()« ";
00166                                                   if ((*suf_it)->canGenerate()){
00167                                                   //se la sequenza non è gia presente allora si
      inserisce altrimenti niente
00168                                                   //cout«"puo generare"«std::endl;
00169                                                           bool trovato=find(k_1,(*suf_it));
00170                                                           if (!trovato)
00171                                                                   k_1->push_back((*suf_it));
00172                                                           cont++;
00173                                                           canContinue=true;
00174                                                   }
00175                                                   else {
00176                                                           bool trovato =
      find(da_cancellare,(*suf_it));
00177                                                           if (!trovato)
00178
      da_cancellare->push_back((*suf_it));
00179                                                   //cout«"inserisco in da_cancellare"«std::endl;
00180
00181                                                   }
00182                                           }
00183                                   }
00184
00185                                  std::vector<CCSMSequence*>::iterator del;
00186                                   if (seq_length!=0){
00187                                           for (del = k_2->begin();del!=k_2->end();++del){
00188                                                   delete (*del);
00189                                           }
00190                                           delete k_2;
00191                                   }
00192
00193                                   time(&end1);
00194                                   k_2 = k_1;
00195
00196                                   if (check>1&&canContinue){
00197                                           time(&init);
00198
00199                                           pos_canc = CCSMSequence::reduce_tr(k_rif);
00200                                           time(&end);
00201                                           //      std::cout«"tempo per scovare le transazioni
      non interessanti "«difftime(end,init)«std::endl;
00202                                           if (pos_canc.size()>1){
00203                                                   time(&init);
00204                                                   for
      (del=rif_singleton->begin();del!=rif_singleton->end();++del){
00205                                  //cout«"ridotto singleton"«std::endl;
00206                                                           (*del)->reduction(pos_canc);
00207                                                   }
00208                                                   time(&end);
00209                                                   //      std::cout«"tempo riduzione di
      "«rif_singleton->size()«" singleton "«difftime(end,init)«std::endl;
00210                                                   time(&init);
00211                                                   for
      (del=k_rif->begin();del!=k_rif->end();++del){
00212                                           //cout«"ridotta coppia"«std::endl;
00213                                                           (*del)->reduction(pos_canc);
00214                                                   }
00215                                                   time(&end);
00216                                                   //      std::cout«"tempo riduzione di
      "«k_rif->size()«" non singleton "«difftime(end,init)«std::endl;
00217                                           }
00218                                   }
00219                                   delete k_rif;
00220
00221                                   //      std::cout«"transazioni eliminate "«pos_canc.size()«"
      modalita' "«check«std::endl;
00222                                   seq_length++;
00223          //              std::cout«"da cancellare size "«da_cancellare->size()«std::endl;
00224                                   std::vector<CCSMSequence*>::iterator da_c;
00225                  /*      canContinue=false;*/
00226
00227                                   if (seq_length!=k&&canContinue){
00228                                           for ( da_c=
      da_cancellare->begin();da_c!=da_cancellare->end();++da_c){
00229          //                      //      std::cout«"cancello "«(*da_c)->toStdString()«" che hano sup
      "«(*da_c)->getCount()«std::endl;
00230                                                   delete (*da_c);
00231                                           }
00232          //                      std::cout«"finita cancellazione: "«da_cancellare->size()«std::endl;
00233                                           delete da_cancellare;
00234                                           da_cancellare = new std::vector<CCSMSequence*>();
00235                                   }
00236
00237
00238                                  }
```

```
00239                           std::vector<CCSMSequence*>::iterator del, it_kk;
00240                                   std::list<CCSMSequence*>::iterator del1;
00241                                   for (del = k_1->begin();del!=k_1->end();++del)
00242                                           delete (*del);
00243           //      std::cout<<"fuori dal ciclo"<<std::endl;
00244
00245                           std::vector<CCSMSequence*>* temp = new std::vector<CCSMSequence*>();
00246
00247                                   for (it_kk=
      da_cancellare->begin();it_kk!=da_cancellare->end();++it_kk){
00248                                           suff = (*it_kk)->getSuffixSequence();
00249                                           pre =  (*it_kk)->getPrefixSequence();
00250
00251                                           for (del1=suff.begin();del1!=suff.end();del1++)
00252                                                   temp->push_back(*del1);
00253
00254                                           for (del1=pre.begin();del1!=pre.end();++del1)
00255                                                   temp->push_back(*del1);
00256                                   }
00257                                   for (it_kk = temp->begin();it_kk!=temp->end();++it_kk){
00258
00259                                           bool trov1 =find(da_cancellare,(*it_kk));
00260                                           if (!trov1)
00261                                                   da_cancellare->push_back((*it_kk));
00262                                   }
00263                                   delete temp;
00264
00265                                   for (del= da_cancellare->begin();del!=da_cancellare->end();++del){
00266                                           delete (*del);
00267                                   }
00268           //      std::cout<<"finita cancellazione: "<<da_cancellare->size()<<std::endl;
00269                                   delete da_cancellare;
00270                                   delete k_1;
00271                           }
00272
00273
00274
00275
00276                           void CCSMiner::mineRules(){
00277                                   double time_tot=0;
00278                                   time_t init, end;
00279
00280                                   const ParsedMinerule& pm = minerule.getParsedMinerule();
00281
00282                                   int min_g=pm.sequenceAllowedGaps.getMin();
00283                                   int max_g=pm.sequenceAllowedGaps.getMax();
00284
00285                                   double threshold=pm.sup;
00286                                   if (min_g>max_g)
00287                                           assert(false);
00288
00289                                   int reduce = 1;
00290                            std::vector<CCSMSequence::ResultItems> result;
00291                            std::vector<CCSMSequence*> singleton;
00292                                   CCSMSequence input;
00293                                   CCSMSequence* single;
00294                            std::vector<CCSMSequence*>::iterator it;
00295                                   size_t number=0;
00296                                   time(&init);
00297                                   int riga=0;
00298
00299
00300                                   prepareData();
00301                                   mrdb::ResultSet* rs = statement->executeQuery();
00302
00303                                   rs->next();
00304                                   while (!rs->isAfterLast()){
00305                                           // SourceRow hbsr(rs,rowDes);
00306                                           input.read(rs, rowDes);
00307                                           // std::cout<<"LEGGO SEQUENZA: "<<input.toStdString()<<std::endl;
00308                                           for(size_t i=0;i<input.size();++i){
00309
00310                                                   single= new CCSMSequence(input,i,1);
00311
00312                                                   single->setLastItem(single);
00313                                                   bool trovato=false;
00314
00315                                                   for (it =
      singleton.begin();it!=singleton.end()&&!trovato;it++){
00316                                                           CCSMSequence* suppSingle=(*it);
00317                                                           if (*suppSingle == *single){
00318                                                                   trovato = true;
00319                                                                   (*it)->setPresent(number,1);
00320                                                                   (*it)->setEid(number,i);
00321                                                           }
00322                                                   }
00323                                                   if (!trovato){
```

```
00324                                                   single->setPresent(number,1);
00325                                                   single->setEid(number,i);
00326                                                   singleton.push_back(single);
00327                                           }
00328                                           else
00329                                                   delete single;
00330                                   }
00331                                   input.svuota();
00332                                   number++;
00333                                   riga++;
00334                           }
00335
00336                           std::cout<<std::endl;
00337                           for (size_t i=0;i<singleton.size();i++){
00338                                   singleton[i]->setPresent(number-1,0);
00339                           }
00340                           //      std::cout<<"numbe e uguale a: "<<number<<std::endl;
00341                           //                              in.close();
00342                           //      std::cout<<"size "<<singleton.size()<<std::endl;
00343                           time(&end);
00344                           // double ttt = difftime(end,init);
00345                           //                      std::cout<<"tempo finora "<<ttt<<std::endl;
00346                           //      std::cout<<"tempo medio per riga"<<ttt/number<<std::endl;
00347                           time_tot=time_tot+difftime(end,init);
00348                           time(&init);
00349
00350                           threshold=number*threshold;
00351                           if (threshold == 0)
00352                                   threshold = 0.00001;
00353
00354                   std::vector<CCSMSequence*>* suppSingleton= new std::vector<CCSMSequence*>();
00355                   std::vector<CCSMSequence*>::iterator itt;
00356
00357                           for (itt=singleton.begin();itt!=singleton.end();++itt){
00358                                   if ((*itt)->getCount()>=threshold){
00359                                           suppSingleton->push_back((*itt));
00360                                   }
00361                                   else delete (*itt);
00362                           }
00363
00364                           //      std::cout<<"tempo finora "<<difftime(end,init)<<std::endl;
00365                           time_tot=time_tot+difftime(end,init);
00366
00367                           if (reduce>0){
00368                                   time(&init);
00369                                   std::vector<size_t> temp =
    CCSMSequence::reduce_tr(suppSingleton);
00370
00371                                   if (temp.size()!=0){
00372                                           for
    (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt){
00373                                                   (*itt)->reduction(temp);
00374                                           }
00375                                   }
00376                                   time(&end);
00377                                   //      std::cout<<"transazioni eliminate sui
    singleton"<<temp.size()<<std::endl;
00378                                   //cout<<"tempo della riduzione delle transazioni
    "<<difftime(end,init)<<std::endl;
00379                                   time_tot=time_tot+difftime(end,init);
00380                           }
00381
00382                           time(&init);
00383                           time_t init1, end1;
00384                   std::vector<CCSMSequence*>* coppie= new std::vector<CCSMSequence*>();
00385                           //      std::cout<<"la threshold e uguale a: "<<threshold<<std::endl;
00386                           for (size_t i=0;i<suppSingleton->size();++i){
00387                                   for (size_t j=0;j<suppSingleton->size();++j){
00388
00389                                           time(&init1);
00390
00391                                           CCSMSequence* dino =
00392
    CCSMSequence::merge((*suppSingleton)[i],(*suppSingleton)[j]->getLastItem(),min_g,max_g,threshold);
00393                                           time(&end1);
00394                                           if (dino->getCount()>=threshold){
00395                                                   coppie->push_back(dino);
00396                                                   (*suppSingleton)[i]->addPrefixSequence(dino);
00397                                                   (*suppSingleton)[j]->addSuffixSequence(dino);
00398                                                   result.push_back(dino->getSequenceItems());
00399
00400                                                   }else
00401                                                           delete dino;
00402                                           }
00403                                   }
00404                                   time(&end);
00405                                   time_tot=time_tot+difftime(end,init);
```

```
00406                                         //cout«"tempo x generare gli insiemi F1 ed F2
      "«difftime(end,init)«std::endl;
00407
00408
00409                                         if (reduce>0){
00410                                                 time(&init);
00411                 //std::vector<size_t>
00412                                                 std::vector<size_t> temp =
      CCSMSequence::reduce_tr(coppie);
00413                                                 if (temp.size()!=0){
00414                                                         for
      (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt){
00415                                 //cout«"ridotto singleton"«std::endl;
00416                                                                 (*itt)->reduction(temp);
00417                                                         }
00418                                                         for
      (itt=coppie->begin();itt!=coppie->end();++itt){
00419                                 //cout«"ridotta coppia"«std::endl;
00420                                                                 (*itt)->reduction(temp);
00421                                                         }
00422                                                 }
00423                                                 time(&end);
00424                                                 //      std::cout«"transazioni eliminate
      "«temp.size()«std::endl;
00425                                                 //cout«"tempo della riduzione delle transazioni
      "«difftime(end,init)«std::endl;
00426                                                 time_tot=time_tot+difftime(end,init);
00427                                         }
00428                                         //
00429                                         //              std::cout«"prima di combina"«std::endl;
00430                                         //cout«"cardinalita F1 : "«suppSingleton->size()«std::endl;
00431                                         time(&init);
00432
00433                                         combina(result, suppSingleton ,10,
      min_g,max_g,threshold,reduce);
00434
00435                                         //cout«"dopo combina"«std::endl;
00436                                         time(&end);
00437
00438                                         for (size_t i=0;i<result.size();++i){
00439                                          std::vector<ItemType>::iterator it_list =
      result[i].first.begin();
00440                                                 for (;it_list!=result[i].first.end();++it_list)
00441                                                         std::cout«(*it_list)«" ";
00442                                                 std::cout«" con conteggio: "«result[i].second;
00443                                                 std::cout«std::endl;
00444                                         }
00445
00446                                         time_tot=time_tot+difftime(end,init);
00447                                         //      std::cout«"tempo x generare insiemi da F3....
      "«difftime(end,init)«std::endl;
00448                                         //cout«"tempo totale "«time_tot«std::endl;
00449                                         for
      (itt=suppSingleton->begin();itt!=suppSingleton->end();++itt)
00450                                                 delete (*itt);
00451                                         delete coppie;
00452 /*      for (itt=coppie.begin();itt!=coppie.end();++itt)
00453                                         delete (*itt);*/
00454                                                 delete suppSingleton;
00455                                         std::cout«"cardinalita insieme risultato
      "«result.size()«std::endl;
00456                                         //      std::cout«"threshold "«threshold«std::endl;
00457
00458                                 }
00459
00460                                 } // namespace
00461
```

7.217 /Users/esposito/Software/minerule/src/Algorithms/↩ CCSMSequence.cpp File Reference

```
#include "minerule/Algorithms/CCSMSequence.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/Database/SourceRow.hpp"
#include "time.h"
```

Namespaces

- namespace minerule

7.218 CCSMSequence.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/CCSMSequence.hpp"
00017
00018 #include "minerule/Database/SourceRowColumnIds.hpp"
00019 #include "minerule/Database/SourceRow.hpp"
00020
00021 #include "time.h"
00022
00023
00024 namespace minerule{
00025
00026         void CCSMSequence::read(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes){
00027         SourceRow hbsr(rs,rowDes);
00028         ItemType gid;
00029         if(!rs->isAfterLast())
00030                 gid = hbsr.getGroup();
00031         while (!rs->isAfterLast() && gid == hbsr.getGroup()) {
00032                 seq->push_back( hbsr.getBody() );
00033
00034                 rs->next();
00035                 if(!rs->isAfterLast())
00036                         hbsr.init(rs,rowDes);
00037         }
00038   }
00039
00040
00041   /*
00042         void CCSMSequence::read(const std::string& s){
00043         std::string empty("");
00044                 size_t i=0;
00045                 while (s[i]==' ')
00046                         i++;
00047                 if (i==s.size())
00048                         return;
00049         std::string temp("");
00050                 for (;i<s.size();i++){
00051                         if (s[i]==' '){
00052                                 if (temp!=empty){
00053                                         seq->push_back(temp);
00054                                 }
00055                                 temp="";
00056                         }else {
00057                                 temp=temp+s[i];
00058                         }
00059                 }
00060                 if (temp!=empty)
00061                         seq->push_back(temp);
00062
00063         }
00064   */
00065         bool CCSMSequence::operator<(const CCSMSequence& in){
00066                 return lexicographical_compare(seq->begin(),
00067                                                seq->end(),
00068                                                in.seq->begin(),
00069                                                in.seq->end());
00070
00071         };
00072
00073         bool CCSMSequence::operator==(const CCSMSequence& in){
```

```
00074                 bool first=
00075                         lexicographical_compare(seq->begin(),
00076                                                 seq->end(),
00077                                                 in.seq->begin(),
00078                                                 in.seq->end());
00079                 bool second=
00080                         lexicographical_compare(in.seq->begin(),
00081                                                 in.seq->end(),
00082                                                 seq->begin(),
00083                                                 seq->end());
00084                 return (!first)&&(!second);
00085
00086         };
00087
00088         //il second e il singleton
00089         //il first e la sequenza da estendere
00090         CCSMSequence* CCSMSequence::merge(CCSMSequence* first, CCSMSequence* second, int min_gap ,int
      max_gap, double threshold){
00091
00092                 CCSMSequence* ris= new CCSMSequence(*first);//=first;
00093                 ris->setLastItem(second);
00094                 //cout«"ecco il LAST ITEM"«std::endl;
00095                 //cout«ris->getLastItem()->toStdString()«std::endl;
00096                 //List_Type::iterator it;
00097                 /*
00098                 for (it = second->seq->begin();it!=second->seq->end();it++)
00099                         ris->seq->push_back(*it);
00100                 */
00101                 ris->seq->push_back((*second->seq)[0]);
00102
00103                 ris->count=0;
00104                 BitVector::iterator it_ris, it_second;
00105                 it_ris=ris->listSid.begin();
00106                 it_second = second->listSid.begin();
00108 /*
00109                 for (size_t i=0;i<first->listSid.size();i++,++it_second,++it_ris){
00110                         bool ris_i =(*it_ris)&(*it_second);
00111                         (*it_ris)=ris_i;
00112                         if (ris_i)
00113                                 ris->count++;
00114                 }
00115                 if (ris->count<threshold){
00116                         //      std::cout«"ESCO 1"«std::endl;
00117                                 return ris;
00118                 }
00120
00121                 it_ris=ris->listSid.begin();
00122                 it_second = second->listSid.begin();
00123                 ris->count=0;
00124 */
00125                 for (size_t i=0;i<first->listSid.size();i++,++it_second,++it_ris){
00126                         bool ris_i =(*it_ris)&(*it_second);
00127                         (*it_ris)=ris_i;
00128
00129                         Eid_List *f=first->eid_vector[i];
00130                         Eid_List *s=second->eid_vector[i];
00131                         if (ris_i && (*s)[s->size()-1].first > (*f)[0].second ){
00132                                 ris->count++;
00133                         }else
00134                                 (*it_ris)=ABSENT;
00135                 }
00136
00137                 if (ris->count<threshold){
00138                         //      std::cout«"ESCO 2"«std::endl;
00139                                 return ris;
00140                 }
00141
00142                 it_ris=ris->listSid.begin();
00143                 bool canContinue=true;
00144                 for (size_t i=0;i<first->listSid.size()&&canContinue;++i,++it_ris){
00145                         Eid_List* event_id;
00146                         if ((*it_ris)){//se la sequenza è presente allora calcolo intersezione tra le
      event_list
00147
      event_id=CCSMSequence::mergeEidRtoLeft(*first->eid_vector[i],*second->eid_vector[i],min_gap,max_gap);
00148                         }
00149                         else
00150                                 event_id=new Eid_List();
00151                         //CORREZIONE DELla lista dei SID per tener conto dell ordine
00152                         //siccome se second appare prima di first il sid per quella transazione e
      messo ad uno erroneamente
00153                         //xcio lo correggo controllando se sono stati generati event_id
00154                         //se non sono stati generati setto il listSid[i] ad ABSENT ed incremento count
      solo quando ho degli event_id
00155
00156                         if ((*it_ris) && event_id->size()==0){
00157                                 (*it_ris)=ABSENT;
```

```
00158                                   ris->count--;
00159                           }
00160
00161                           if (ris->count<threshold){
00162                                   canContinue=false;
00163                           }
00164                           ris->eid_vector.push_back(event_id);
00165                   }
00166                   //cout«"fusioni di event list "«cc«std::endl;
00167                   return ris;
00168
00169           };
00170
00171
00172
00173
00174
00175           //in questo caso il second è il singleton che cerchiamo di aggiungere alla sequenza
00176           //stiamo estendendo la sequenza da sinistra verso destra, aggiungendo al fondo un elemento
00177           std::vector<std::pair<int,int> >* CCSMSequence::mergeEidRtoLeft(const Eid_List& first, const
      Eid_List& second, int min_gap  , int max_gap){
00178           Eid_List* ris= new Eid_List();
00179           //ris->reserve(10000);
00180           int last_eid_seq;
00181           int eid_singleton;
00182           int first_eid_seq;
00183           Eid_List::const_iterator it_first, it_second;
00184           it_first = first.begin();
00185           if (second[second.size()-1].first<=(*it_first).second){
00186                   //cout«"escamotage"«std::endl;
00187                   return ris;
00188           }
00189           bool continue_f=true;
00190           bool continue_s=true;
00191
00192           for (it_first=first.begin();it_first!=first.end()&&continue_f;++it_first){
00193
00194                   last_eid_seq=(*it_first).second;
00195                   first_eid_seq=(*it_first).first;
00196
00197                   if (second[second.size()-1].first>last_eid_seq )
00198                   {       continue_s=true;
00199
00200
00201                           for
      (it_second=second.begin();it_second!=second.end()&&continue_s;++it_second){
00202                                   eid_singleton=(*it_second).first;
00203                                   int diff_event= eid_singleton-last_eid_seq;
00204                                   //se l'ultimo elemento analizzato della Eid_List dell elemento con cui
      si estende in coda
00205                                   //e in posizione > max_gap+1 rispetto all'elemento della Eid_List del
      primo elemento
00206                                   //si smette di analizzare gli elementi del secondo elemento siccome
      non potranno più essere inseriti
00207                                   //in coda per generare nuove sequenze
00208                                   if ( diff_event > max_gap+1 )
00209                                           continue_s=false;
00210                                   //se al massimo il numero degli elementi tra l'ultimo elemento della
      sequenza
00211                                   //e il singleton e <= max_gap
00212                                   //&&   il singleton nella sequenza compare dopo l'ultimo elemento
      della sequenza
00213                                   //&&   il singleton sia distante almeno mig_gap dalla sequenza presa
      in considerazione
00214                                   if( eid_singleton > last_eid_seq &&
00215                                       eid_singleton-max_gap-1<=last_eid_seq &&
00216                                       eid_singleton-last_eid_seq-1>=min_gap){
00217                                                           //cout«"inserisco coppia
      "«first_eid_seq«","«eid_singleton«std::endl;
00218                                                           //cout«"inserito"«std::endl;
00219
      ris->push_back(std::pair<int,int>(first_eid_seq,eid_singleton));
00220                                   }
00221                           }
00222                   }else
00223                           continue_f=false;
00224
00225           }
00226           return ris;
00227           }
00228
00229
00230 }// END NAMESPACE
```

7.219 /Users/esposito/Software/minerule/src/Algorithms/ConstrItemSetsExtraction.cpp File Reference

```
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
```

Namespaces

- namespace minerule

Macros

- #define MROPTIONFILE "mroptions"

7.219.1 Macro Definition Documentation

7.219.1.1 MROPTIONFILE

```
#define MROPTIONFILE "mroptions"
```

Definition at line 23 of file ConstrItemSetsExtraction.cpp.

7.220 ConstrItemSetsExtraction.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 // care.cpp
00017
00018
00019 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00020 #include "minerule/Database/PrepareDataUtils.hpp"
00021 #include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
00022
00023 #define MROPTIONFILE "mroptions"
00024
00025 namespace minerule {
00026   void ConstrItemSetsExtraction::prepareData() {
00027     MineruleOptions& mrOptions = MineruleOptions::getSharedOptions();
00028
00029     options.setSupport( minerule.getParsedMinerule().sup );
```

```
00030     options.setConfidence( minerule.getParsedMinerule().conf );
00031
00032     MinMaxPair bodyCards(minerule.getParsedMinerule().bodyCardinalities);
00033     bodyCards.applyConstraints(mrOptions.getParsers().getBodyCardinalities());
00034     options.setBodyCardinalities(bodyCards);
00035
00036     MinMaxPair headCards(minerule.getParsedMinerule().headCardinalities);
00037     headCards.applyConstraints(mrOptions.getParsers().getHeadCardinalities());
00038     options.setHeadCardinalities(headCards);
00039
00040     sourceTable = new SourceTable(*this);
00041           bodyIterator = sourceTable->newIterator(SourceTable::BodyIterator);
00042
00043     MRLog() « "Building db queries" « std::endl;
00044     MRLog() « "Executing queries" « std::endl;
00045
00046     connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00047     connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00048     connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00049     //connection.setHeadCardinalities(minerule.getParsedMinerule().headCardinalities);
00050     connection.setHeadCardinalities(MinMaxPair(0,1000));
00051     connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00052     connection.init();
00053   }
00054
00055   void ConstrItemSetsExtraction::mineRules() {
00056     MRLogPush("Starting ConstrItemSetsExtraction mining algorithm...");
00057
00058     MRLog() « "Preparing data sources..." « std::endl;
00059     prepareData();
00060
00061     float support = options.getSupport();
00062     int maxBody = options.getBodyCardinalities().getMax();
00063
00064     ItemType gid1;
00065     BodyMap bodyMap(connection,1);
00066
00067     int howManyGroups = 0;
00068     int totalGroups = sourceTable->getTotGroups();
00069     int howManyRows = 0;
00070
00071     while (!bodyIterator.isAfterLast()) {
00072                 ItemType gid = bodyIterator->getGroup();
00073                 Transaction t1, t2;
00074
00075                 t1.loadBody(gid,bodyIterator);
00076                 howManyRows += bodyMap.add(t1,howManyGroups);
00077                 howManyGroups++;
00078     }
00079     MRLog() « "Total rows: " « howManyRows « std::endl;
00080     MRLog() « "Total groups: " « totalGroups « std::endl;
00081     MRLogPop();
00082
00083     MRLogPush("Starting itemset extraction...");
00084
00085     bodyMap.updateCount();
00086     MRLog() « "Total bodies before pruning: " « bodyMap.size() « std::endl;
00087     bodyMap.pruneMap(support*totalGroups,true);
00088     bodyMap.setTotalGroups(totalGroups);
00089     std::vector<BodyMap::iterator> v;
00090     std::multimap<int, BodyMap::iterator> temp;
00091     for (BodyMap::iterator i = bodyMap.begin(); i!=bodyMap.end(); i++)
00092         if (!i->second.done) temp.insert(std::pair<int, BodyMap::iterator>(i->second.count(),i));
00093     for (std::multimap<int, BodyMap::iterator>::iterator i = temp.begin(); i!=temp.end(); i++)
00094         v.insert(v.end(),i->second);
00095     MRLog() « "Total bodies after pruning: " « v.size() « std::endl;
00096
00097     NewRule r;
00098     NewRuleSet rs;
00099     int nrules = bodyMap.generateStartItemSets(rs,r,v,maxBody,support*totalGroups,-1);
00100     MRLog() « "After extracting itemsets: " « nrules « std::endl;
00101     MRLogPop();
00102
00103     connection.finalize();
00104   }
00105
00106 } //end namespace
```

7.221 /Users/esposito/Software/minerule/src/Algorithms/ConstrTree.cpp File Reference

```
#include "minerule/Algorithms/ConstrTree.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include <iterator>
```

Namespaces

- namespace minerule

7.222 ConstrTree.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/ConstrTree.hpp"
00017 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019 #include "minerule/Database/PrepareDataUtils.hpp"
00020 #include <iterator>
00021
00022
00023
00024
00025 namespace minerule {
00026
00027   //per algo costruttivo: nella struttura non metto i supporti a zero
00028   void ConstrTree::insertRulesInStructure(){
00029     QueryResult::Iterator qit;
00030     //il primo paramentro e' il nome della tabella dei ris della vecchia query
00031     OptimizerCatalogue::getMRQueryResultIterator(minerule->getOptimizationInfo().minerule.tab_result,
      qit, -1, 0.0);
00032     Head* newhead;
00033     while( qit.next()) {
00034       Rule r;
00035       //crea body e head che sono due vettori ItemSet
00036       //ovvero due vettori di ItemType
00037       //ogni ItemType contiene un puntatore ad un SourceRowElement
00038       qit.getRule(r);
00039       //insertRuleInStructure(r);
00040       //body e head sono due puntatori a ItemSet
00041       newhead=root->insertItemSetB(r.getBody(),0.0);
00042       newhead->insertItemSetH(r.getHead(),0.0);
00043     }
00044
00045   }
00046
00047
00048
00049         //   void ConstrTree::adjustSuppMIndex(){
00050         //
00051         //     while( mb2->current() != mb2->end() ) {
00052         //      std::string g=mb2->getCurrentGID();
00053         //       ItemSet* body=new ItemSet();
```

```
00054         //       //cout«"in adjustSupp:"«std::endl;
00055         //       for(; mb2->current() != mb2->end() && mb2->getCurrentGID()==g;(*mb2)++){
00056         // ItemType gid(*SourceRowElement::deserializeElementFromString("n "+mb2->getCurrentGID()));
00057         // ItemType item(*SourceRowElement::deserializeElementFromString("n "+mb2->getCurrentItem()));
00058         // //cout«item.asString()«" "«gid.asString()«std::endl;
00059         // body->push_back(item);
00060         //       }
00061         //
00062         //
00063         //       if (g!=mh2->getCurrentGID())
00064         // this->root->findBodiesInTree(body);
00065         //       else{
00066         // ItemSet* head=new ItemSet();
00067         // for(;mh2->current() != mh2->end() && mh2->getCurrentGID()==g; (*mh2)++){
00068         //   ItemType item(*SourceRowElement::deserializeElementFromString("n
      "+mh2->getCurrentItem()));
00069         //   head->push_back(item);
00070         // }
00071         // this->root->findRulesInTree(body,head);
00072         // delete head;
00073         //       }
00074         //       delete body;
00075         //     }
00076         //   }
00077
00078   void ConstrTree::adjustSuppRSet(){
00079     ItemType gid;
00080     ItemType gidb;
00081     ItemType item;
00082
00083     bool bodynotend=rb2->next();
00084     bool headnotend=rh2->next();
00085     while( bodynotend ) {
00086
00087       SourceRow curRowb(rb2, bodyDes);
00088       //std::string g=mb2->getCurrentGID();
00089       gidb=curRowb.getGroup();
00090
00091       ItemSet* body=new ItemSet();
00092       //cout«"in adjustSupp:"«std::endl;
00093
00094       while(bodynotend && ItemType(curRowb.getGroup())==gidb){
00095         body->push_back(curRowb.getBody());
00096         if((bodynotend=rb2->next())) {
00097           curRowb.init(rb2, bodyDes);
00098         }
00099       }
00100
00101       if(headnotend) {
00102         SourceRow curRowh(rh2, headDes);
00103         if (gidb!=curRowh.getGroup())
00104           this->root->findBodiesInTree(body);
00105         else{
00106           ItemSet* head=new ItemSet();
00107           while(headnotend && ItemType(curRowh.getGroup())==gidb ){
00108             head->push_back(curRowh.getHead());
00109             if ((headnotend=rh2->next())) {
00110               curRowh.init(rh2, headDes);
00111             }
00112           }
00113
00114           this->root->findRulesInTree(body,head);
00115           delete head;
00116         }
00117       }
00118       delete body;
00119     }
00120   }
00121
00122
00123   void ConstrTree::adjustSupp(){
00124     MRLogPusher _("Starting the mining algorithm...");
00125
00126     MRLog() « "Reading previous result and preparing data structures..." « std::endl;
00127     insertRulesInStructure();
00128
00129     Connection connection;
00130     connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00131     connection.useMRDBConnection(
00132        MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00133     connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule->getParsedMinerule()));
00134
00135     // if (mb2!=NULL  &&  mh2!=NULL) {
00136     //    adjustSuppMIndex();
00137     //    std::vector<ItemType> body;
00138     //    getRoot()->extractRules(body,
```

```
00139     //                        minerule->getParsedMinerule().sup,
00140     //                        minerule->getParsedMinerule().conf,
00141     //                        ngroups,
00142     //                        &connection);
00143     //  }
00144     //  else {
00145       if (rb2!=NULL  &&  rh2!=NULL) {
00146         MRLog() << "Evaluating constraints and adjusting the support counts in the data structure..." <<
      std::endl;
00147         adjustSuppRSet();
00148   std::vector<ItemType> body;
00149
00150         MRLog() << "Pruning rules and writing results..." << std::endl;
00151         getRoot()->extractRules(body,
00152                                 minerule->getParsedMinerule().sup,
00153                                 minerule->getParsedMinerule().conf,
00154                                 ngroups,
00155                                 &connection);
00156       } else
00157         throw MineruleException (MR_ERROR_INTERNAL,
00158                                  " cannot create rules ");
00159     // }
00160   }
00161
00162
00163   size_t ConstrTree::buildAttrStr(const ParsedMinerule::AttrVector& attr,
00164                                                       size_t startIndex,
00165                                                       std::string& attrStr,
00166                                                       std::vector<int>& des) const {
00167     ParsedMinerule::AttrVector::const_iterator it = attr.begin();
00168     for( ; it!=attr.end(); it++ ) {
00169       if(it!=attr.begin()) {
00170         attrStr+=",";
00171       }
00172
00173       attrStr+=*it;
00174       des.push_back(++startIndex);
00175     }
00176
00177     return startIndex;
00178   }
00179
00180
00181   // Preprocessing functions
00182
00183   std::string ConstrTree::buildQry( const std::string& groupAttrStr,
00184                                                     const std::string& attrStr,
00185                                                     const std::string& constraints) const {
00186
00187     return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00188                   "FROM "+minerule->getParsedMinerule().tab_source+" "+
00189                   (constraints.size()>0 ?
00190                   "WHERE "+constraints+" " :
00191                   "")
00192                   +"ORDER BY "+groupAttrStr);
00193   }
00194
00195
00196   void ConstrTree::prepareData() {
00197       MRLogPusher _("Building source information");
00198
00199           MRLog() << "Separating the constraints in the HEAD and BODY parts..."<<std::endl;
00200           std::string bodyConstraints;
00201           std::string headConstraints;
00202           HeadBodyPredicatesSeparator::separate(minerule->getParsedMinerule().mc->l_and,
00203                   bodyConstraints,
00204                           headConstraints);
00205
00206           MRLog() << "Building db queries" << std::endl;
00207           size_t index;
00208           std::string groupAttr;
00209           std::string bodyAttr;
00210           std::string headAttr;
00211           index=buildAttrStr(minerule->getParsedMinerule().ga, 0, groupAttr, bodyDes.groupElems );
00212
00213           headDes.groupElems=bodyDes.groupElems;
00214
00215           buildAttrStr(minerule->getParsedMinerule().ba, index, bodyAttr, bodyDes.bodyElems);
00216           buildAttrStr(minerule->getParsedMinerule().ha, index, headAttr, headDes.headElems);
00217
00218           std::string bodyQry = buildQry( groupAttr, bodyAttr, bodyConstraints);
00219
00220           std::string headQry = buildQry( groupAttr, headAttr, headConstraints);
00221
00222           MRLog() << "Body query" << bodyQry << std::endl;
00223           MRLog() << "Head query" << headQry << std::endl;
00224           MRLog() << "Executing queries" << std::endl;
```

```
00225           mrdb::Connection* con = MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00226
00227           stateb2 = con->createStatement();
00228           rb2 = stateb2->executeQuery(bodyQry);
00229
00230
00231           stateh2 = con->createStatement();
00232           rh2 = stateh2->executeQuery(headQry);
00233           ngroups = PrepareDataUtils::evaluateTotGroups(minerule->getParsedMinerule());
00234   }
00235
00236   void ConstrTree::execute()
00237     {
00238     assert( minerule->getOptimizationInfo().relationship == OptimizedMinerule::Dominance );
00239     assert( minerule->getParsedMinerule().mc!=NULL &&
00240             minerule->getParsedMinerule().mc->next==NULL);
00241
00242     MRLogPusher _("This is the Context Dependent Constructive Mining Algorithm...");
00243
00244         prepareData();
00245     adjustSupp();
00246   }
00247 } // namespace
```

7.223 /Users/esposito/Software/minerule/src/Algorithms/DestrTree.cpp File Reference

```
#include "minerule/Algorithms/DestrTree.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include <iterator>
#include "minerule/Database/PrepareDataUtils.hpp"
#include <cmath>
```

Namespaces

- namespace minerule

7.224 DestrTree.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/DestrTree.hpp"
00017 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019 #include <iterator>
00020 #include "minerule/Database/PrepareDataUtils.hpp"
00021 #include <cmath>
00022
00023
00024
00025
```

```
00026 namespace minerule {
00027
00028         void DestrTree::insertRulesInStructure(){
00029                 QueryResult::Iterator qit;
00030
00031
      OptimizerCatalogue::getMRQueryResultIterator(minerule->getOptimizationInfo().minerule.tab_result,
      qit, -1, 0.0);
00032                 Head* newhead;
00033                 while( qit.next()) {
00034                         Rule r;
00035                         qit.getRule(r);
00036                         double suppr=round(r.getSupport()*ngroups);
00037                         double suppb=round(suppr/r.getConfidence());
00038                         newhead=root->insertItemSetB(r.getBody(),suppb);
00039                         newhead->insertItemSetH(r.getHead(),suppr);
00040                 }
00041
00042         }
00043
00044
00045         // void DestrTree::adjustSuppMIndex(){
00046         //      MRLog() << "findRulesInTree da adjustSuppMIndex()..."<<std::endl;
00047         //      bool notend=1;
00048         //     //body e head servono per capire quando devo andare avanti nella rispettiva lista
      oppure no
00049         //      bool newbody=1;
00050         //      bool newhead=1;
00051         //      ItemSet* b1=new ItemSet();
00052         //      ItemSet* b1nb2=new ItemSet();
00053         //      ItemSet* h1=new ItemSet();
00054         //      ItemSet* h1nh2=new ItemSet();
00055         //      while( notend ) {
00056         //              std::string gb,gh;
00057         //              if (newbody){delete b1; delete b1nb2;
00058         //                      ItemSet* b1=new ItemSet();
00059         //                      ItemSet* b1nb2=new ItemSet();
00060         //                      gb=mb1->getCurrentGID();
00061         //                      for(; mb1->current() != mb1->end() &&
      mb1->getCurrentGID()==gb;(*mb1)++){
00062         //                              ItemType
      item(*SourceRowElement::deserializeElementFromString("n "+mb1->getCurrentItem()));
00063         //                              b1->push_back(item);
00064         //                      }
00065         //                      for(; mb1nb2->current() != mb1nb2->end() &&
      mb1->getCurrentGID()==gb;(*mb1nb2)++){
00066         //                              ItemType
      item(*SourceRowElement::deserializeElementFromString("n "+mb1->getCurrentItem()));
00067         //                              b1nb2->push_back(item);
00068         //                      }
00069         //              }
00070         //              if (newhead){delete h1; delete h1nh2;
00071         //                      ItemSet* h1=new ItemSet();
00072         //                      ItemSet* h1nh2=new ItemSet();
00073         //                      gh=mb1->getCurrentGID();
00074         //                      for(; mh1->current() != mh1->end() &&
      mh1->getCurrentGID()==gh;(*mh1)++){
00075         //                              ItemType
      item(*SourceRowElement::deserializeElementFromString("n "+mh1->getCurrentItem()));
00076         //                              h1->push_back(item);
00077         //                      }
00078         //                      for(; mh1nh2->current() != mh1nh2->end() &&
      mh1->getCurrentGID()==gh;(*mh1nh2)++){
00079         //                              ItemType
      item(*SourceRowElement::deserializeElementFromString("n "+mh1->getCurrentItem()));
00080         //                              h1nh2->push_back(item);
00081         //                      }
00082         //              }
00083         //
00084         //              if (gb>gh) {
00085         //
00086         //                      root->findRulesInTree(NULL, NULL, h1, h1nh2);
00087         //                      newbody=0;newhead=1;
00088         //              }else if (gb<gh) {
00089         //                      root->findRulesInTree(b1, b1nb2, NULL, NULL);
00090         //                      newbody=1;newhead=0;
00091         //
00092         //              }
00093         //              else if (gb==gh) {
00094         //
00095         //                      root->findRulesInTree(b1, b1nb2, h1, h1nh2);
00096         //                      newbody=1;newhead=1;
00097         //
00098         //              }
00099         //
00100         //              if (mb1->current()==mb1->end() && mh1->current()==mh1->end()) notend=0;
00101         //      }
```

```
00102         // }
00103
00104
00105
00106         void DestrTree::adjustSuppRSet(){
00107                 MRLog() « "findRulesInTree da adjustSuppRSet()..."«std::endl;
00108                 ItemType gb;
00109                 ItemType gh;
00110                 bool newbody=1;
00111                 bool newhead=1;
00112                 bool notend=1;
00113                 bool b1notend=rb1->next();
00114                 bool h1notend=rh1->next();
00115                 bool b1nb2notend=rb1nb2->next();
00116                 bool h1nh2notend=rh1nh2->next();
00117                 ItemSet* b1=new ItemSet();
00118                 ItemSet* b1nb2=new ItemSet();
00119                 ItemSet* h1=new ItemSet();
00120                 ItemSet* h1nh2=new ItemSet();
00121
00122                 while( notend ) {
00123                         if (b1notend && newbody){
00124                                 delete b1; delete b1nb2;
00125                                 b1=new ItemSet();
00126                                 b1nb2=new ItemSet();
00127                                 SourceRow curRowb1(rb1, bodyDes);
00128                                 gb=curRowb1.getGroup();
00129
00130                                 while(b1notend && ItemType(curRowb1.getGroup())==gb){
00131                                         b1->push_back(curRowb1.getBody());
00132                                         if((b1notend=rb1->next())) {
00133                                                 curRowb1.init(rb1, bodyDes);
00134                                         }
00135                                 }
00136                                 if(b1nb2notend){
00137                                         SourceRow curRowb12(rb1nb2, bodyDes);
00138                                         while(b1nb2notend && ItemType(curRowb12.getGroup())==gb){
00139                                                 b1nb2->push_back(curRowb12.getBody());
00140                                                 if((b1nb2notend=rb1nb2->next())) {
00141                                                         curRowb12.init(rb1nb2, bodyDes);
00142                                                 }
00143                                         }
00144                                 }
00145                         }
00146
00147
00148
00149                         if(h1notend && newhead) {
00150                                 delete h1; delete h1nh2;
00151                                 h1=new ItemSet();
00152                                 h1nh2=new ItemSet();
00153                                 SourceRow curRowh1(rh1, headDes);
00154                                 gh=curRowh1.getGroup();
00155
00156                                 while(h1notend && ItemType(curRowh1.getGroup())==gh ){
00157                                         h1->push_back(curRowh1.getHead());
00158                                         if ((h1notend=rh1->next())) {
00159                                                 curRowh1.init(rh1, headDes);
00160                                         }
00161                                 }
00162                                 if(h1nh2notend){
00163                                         SourceRow curRowh12(rh1nh2, headDes);
00164                                         while(h1nh2notend && ItemType(curRowh12.getGroup())==gh ){
00165                                                 h1nh2->push_back(curRowh12.getHead());
00166                                                 if ((h1nh2notend=rh1nh2->next())) {
00167                                                         curRowh12.init(rh1nh2, headDes);
00168                                                 }
00169                                         }
00170                                 }
00171                         }
00172
00173
00174                         if (!(gb<gh) && !(gb==gh)) {
00175                                 root->findRulesInTree(NULL, NULL, h1, h1nh2);
00176                                 newbody=0;newhead=1;
00177                         } else if (gb<gh) {
00178                                 root->findRulesInTree(b1, b1nb2, NULL, NULL);
00179                                 newbody=1;newhead=0;
00180                         } else if (gb==gh) {
00181                                 root->findRulesInTree(b1, b1nb2, h1, h1nh2);
00182                                 newbody=1;newhead=1;
00183                         }
00184
00185                         if (!b1notend && !h1notend) notend=0;
00186                 }
00187         }
00188
```

```
00189
00190         void DestrTree::adjustSupp(){
00191                 MRLogPusher _("Starting the mining algorithm...");
00192
00193                 MRLog() « "Preparing the data structures..." « std::endl;
00194                 insertRulesInStructure();
00195                 MRLog() « "Done!" « std::endl;
00196
00197                 Connection connection;
00198                 connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00199                 connection.useMRDBConnection(
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00200                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule->getParsedMinerule()));
00201
00202                 if (rb1!=NULL  &&  rh1!=NULL) {
00203                                 MRLog() « "Adjusting support..." « std::endl;
00204                                 adjustSuppRSet();
00205                                 MRLog() « "Done!" « std::endl;
00206
00207                                 std::vector<ItemType> body;
00208                                 MRLog() « "Extracting rules..." « std::endl;
00209                                 getRoot()->extractRules(body, minerule->getParsedMinerule().sup,
      minerule->getParsedMinerule().conf, ngroups, &connection);
00210                                 MRLog() « "Done!" « std::endl;
00211                 } else throw MineruleException (MR_ERROR_INTERNAL, " cannot create rules ");
00212         }
00213
00214
00215         size_t DestrTree::buildAttrStr(const ParsedMinerule::AttrVector& attr,
00216                 size_t startIndex,
00217                         std::string& attrStr,
00218         std::vector<int>& des) const {
00219                 ParsedMinerule::AttrVector::const_iterator it = attr.begin();
00220                 for( ; it!=attr.end(); it++ ) {
00221                         if(it!=attr.begin()) {
00222                                 attrStr+=",";
00223                         }
00224
00225                         attrStr+=*it;
00226                         des.push_back(++startIndex);
00227                 }
00228
00229                 return startIndex;
00230         }
00231
00232
00233  // Preprocessing functions
00234
00235         std::string DestrTree::buildQry( const std::string& groupAttrStr,const std::string&
      attrStr,const std::string& constraints) const {
00236
00237                 return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00238                         "FROM "+minerule->getParsedMinerule().tab_source+" "+
00239                         (constraints.size()>0 ? "WHERE "+constraints+" " :"")
00240                         +"ORDER BY "+groupAttrStr);
00241         }
00242
00243
00244         std::string DestrTree::buildQry1NotQry2( const std::string& groupAttrStr,const std::string&
      attrStr,const std::string& constraint1,  const std::string& constraint2) const {
00245
00246                 return std::string("SELECT "+groupAttrStr+","+attrStr+" "
00247                         "FROM "+minerule->getParsedMinerule().tab_source+" "
00248                         "WHERE "+constraint1+ (constraint2.empty() ? " " : " AND NOT("+constraint2+")
      ") +
00249                         "ORDER BY "+groupAttrStr);
00250         }
00251
00252         void DestrTree::prepareData() {
00253                 MRLogPush("Building source information");
00254
00255                 MRLog() « "Separating the constraints in the HEAD and BODY parts..."«std::endl;
00256                 std::string q1BodyConstraints;
00257                 std::string q1HeadConstraints;
00258                 std::string q2BodyConstraints;
00259                 std::string q2HeadConstraints;
00260
      HeadBodyPredicatesSeparator::separate(minerule->getOptimizationInfo().minerule.mc!=NULL?
      minerule->getOptimizationInfo().minerule.mc->l_and:NULL, q1BodyConstraints, q1HeadConstraints);
00261
00262                 MRLog() « "Separating the constraints in the HEAD and BODY parts (query 2)..." «
      std::endl;
00263
00264                 HeadBodyPredicatesSeparator::separate(minerule->getParsedMinerule().mc->l_and,
      q2BodyConstraints, q2HeadConstraints);
00265
```

```
00266                 MRLog() « "Building db queries" « std::endl;
00267                 size_t index;
00268                 std::string groupAttr;
00269                 std::string bodyAttr;
00270                 std::string headAttr;
00271                 index=buildAttrStr(minerule->getParsedMinerule().ga,0,groupAttr,bodyDes.groupElems );
00272 
00273                 headDes.groupElems=bodyDes.groupElems;
00274 
00275                 buildAttrStr(minerule->getParsedMinerule().ba,index,bodyAttr,bodyDes.bodyElems);
00276                 buildAttrStr(minerule->getParsedMinerule().ha,index,headAttr,headDes.headElems);
00277 
00278                 std::string bodyQry1 =buildQry( groupAttr,bodyAttr,q1BodyConstraints);
00279                 std::string headQry1 =buildQry( groupAttr,headAttr,q1HeadConstraints);
00280 
00281                 std::string bodyQry1NotQry2 = buildQry1NotQry2(
      groupAttr,bodyAttr,q1BodyConstraints,q2BodyConstraints);
00282                 std::string headQry1NotQry2 = buildQry1NotQry2( groupAttr, headAttr,
      q1HeadConstraints, q2HeadConstraints);
00283 
00284                 MRLog() « "Executing queries" « std::endl;
00285                 mrdb::Connection* con =
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00286 
00287 
00288                 stateb1 = con->createStatement();
00289                 MRDebug("bodyQry1:"+bodyQry1);
00290                 rb1 = stateb1->executeQuery(bodyQry1);
00291 
00292                 stateh1 = con->createStatement();
00293                 MRDebug("headQry1:"+headQry1);
00294                 rh1 = stateh1->executeQuery(headQry1);
00295 
00296 
00297                 stateb1nb2 = con->createStatement();
00298                 MRDebug("bodyQry1NotQry2:"+bodyQry1NotQry2);
00299                 rb1nb2 = stateb1nb2->executeQuery(bodyQry1NotQry2);
00300 
00301                 stateh1nh2 = con->createStatement();
00302                 MRDebug("headQry1NotQry2:"+headQry1NotQry2);
00303                 rh1nh2 = stateh1nh2->executeQuery(headQry1NotQry2);
00304 
00305                 ngroups = PrepareDataUtils::evaluateTotGroups(minerule->getParsedMinerule());
00306 
00307                 MRLogPop();
00308         }
00309 
00310 
00311         void DestrTree::execute()
00312         {
00313                 assert( minerule->getOptimizationInfo().relationship == OptimizedMinerule::Dominance
      );
00314                 assert( minerule->getParsedMinerule().mc!=NULL &&
      minerule->getParsedMinerule().mc->next==NULL);
00315 
00316                 MRLogPush("This is the Context Dependent Destructive Mining Algorithm...");
00317                 prepareData();
00318                 adjustSupp();
00319                 MRLogPop();
00320         }
00321 } // namespace
```

7.225 /Users/esposito/Software/minerule/src/Algorithms/FSMiner.cpp File Reference

```
#include "minerule/Algorithms/FSMiner.hpp"
#include "minerule/Algorithms/FSTree.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Database/PrepareDataUtils.hpp"
#include <stdexcept>
```

Namespaces

- namespace minerule

7.226 FSMiner.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/FSMiner.hpp"
00017 #include "minerule/Algorithms/FSTree.hpp"
00018 #include "minerule/Utils/MineruleLogs.hpp"
00019 #include "minerule/Database/PrepareDataUtils.hpp"
00020 #include <stdexcept>
00021
00022 //#include "sequence.hpp"
00023
00024 namespace minerule {
00025
00026         void
00027         FSMiner::prepareData() {
00028                 PrepareDataUtils pdu(minerule.getParsedMinerule(), this->sourceTableRequirements());
00029                 const ParsedMinerule& pm = minerule.getParsedMinerule();
00030                 std::string groupAttrList = pdu.buildAttrListDescription(pm.ga);
00031                 std::string ordAttrList = pdu.buildAttrListDescription(pm.oa);
00032
00033                 std::string sqlQuery =
00034                         "SELECT "+ groupAttrList+","
00035                         + pdu.buildAttrListDescription(pm.ba) + ","
00036                         + ordAttrList + " "
00037                         + "FROM "+ pm.tab_source + " "
00038                         + "ORDER BY "+groupAttrList+","+ordAttrList;
00039
00040                 size_t last_elem;
00041
00042                 last_elem = rowDes.setGroupElems(1,pm.ga.size());
00043
00044                 rowDes.setBodyElems(last_elem+1, pm.ba.size());
00045
00046
      connection.useMRDBConnection(MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00047                 connection.setOutTableName(minerule.getParsedMinerule().tab_result);
00048                 connection.setBodyCardinalities(minerule.getParsedMinerule().bodyCardinalities);
00049
00050             throw std::runtime_error("Algorithm to be updated to use new createResiltTables API");
00051             // connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
      minerule.getParsedMinerule()));
00052
00053                 MRDebug() << "FSMiner query:" << sqlQuery.c_str() << std::endl;
00054
00055                 statement = connection.getMRDBConnection()->prepareStatement(sqlQuery.c_str());
00056
00057         }
00058
00059
00060         void
00061         FSMiner::mineRules(){
00062                 MRLogPush("This is FSMiner. Starting...");
00063
00064                 prepareData();
00065                 FSTree* h=new FSTree(minerule);
00066
00067                 mrdb::ResultSet* rs = statement->executeQuery();
00068                 MRLogPush("Building frequent bigram link table...");
00069                 h->createLinkSTable(rs, rowDes);
00070                 MRLogPop();
00071
00072                 h->setThreshold(options.getSupport());
00073                 MRLog() << "Support threshold is:" << h->getThreshold() << std::endl;
00074
00075                 MRLogPush("Building unfrequent bigram link table...");
00076                 h->createLinkNSTable();
00077                 MRLogPop();
00078
00079                 delete rs;
00080                 rs = statement->executeQuery();
```

```
00081
00082                 MRLogPush("Building FSTree...");
00083                 h->construct_Tree(rs, rowDes);
00084                 MRLogPop();
00085
00086                 delete rs;
00087
00088                 MRLogPush("Starting core mining...");
00089                 h->mine();
00090                 MRLogPop();
00091
00092                 MRLogPop();
00093
00094                 std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00095                 std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* ris = h->getResult();
00096         //std::string s=""
00097                 std::cout<<"ris->size "<<ris->size()<<std::endl;
00098                 MRLogPush(std::string("contiguos frequent sequence find:
       ")+Converter(ris->size()).toString());
00099                 for (res_it=ris->begin();res_it!=ris->end();res_it++){
00100                         FSTreeSequence temp=(*res_it).first;
00101                         MRLogPush(temp.toStdString());
00102                 }
00103
00104
00105         // thrashing the thrashable
00106                 delete ris;
00107                 delete h;
00108                 delete statement;
00109         }
00110
00111         } // namespace
```

7.227 /Users/esposito/Software/minerule/src/Algorithms/FSTree.cpp File Reference

```
#include <fstream>
#include <map>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include "minerule/Algorithms/FSTree.hpp"
```

Namespaces

- namespace minerule

7.228 FSTree.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <fstream>
00017 #include <map>
00018 #include <iostream>
00019 #include <string>
00020 #include <vector>
00021 #include <algorithm>
00022 #include "minerule/Algorithms/FSTree.hpp"
00023
00024
00025
00026
00027 namespace minerule {
00028
00029 void FSTree::insertTree(FSTreeNode* r, FSTreeSequence* t){
00030   //se la sequenza e composta da un solo elemento
00031   //la inserisco come figlio
00032
00033   if (t->size()==1){
00034     appendChild(r,t->removeHead());
00035   }
00036   else{
00037     //altrimenti creo un nuovo nodo temp con il primo item della sequenza
00038     //creo il collegamento nella lista e inserisco il nodo con appendChild
00039     //richiamo insertTree sul resto della sequenza
00040
00041     FSTreeNode* nod = appendChild(r,t->removeHead());
00042
00043     insertTree(nod,t);
00044   }
00045 }
00046
00047 void FSTree::createLinkNSTable(){
00048   //COSTRUISCO MAPPA DEI LINK FREQUENTI E DEI LINK NON FREQUENTI RISPETTO ALLA THRESHOLD
00049   //std::vector<sequence*>::iterator it;
00050   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator it;
00051   int count;
00052   for (it=link_S->begin();it!=link_S->end();it++){
00053
00054     count = (*it).second;
00055     if (count<threshold){
00056       (*link_NS)[(*it).first]=count;
00057     }
00058   }
00059 }
00060
00061 void FSTree::createLinkSTable(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes) {
00062   FSTreeSequence* input = new FSTreeSequence();
00063   rs->next();
00064  while(!rs->isAfterLast()) {
00065     input->read(rs,rowDes);
00066     if (input->size()>1){
00067       insertLink(input);
00068     }
00069
00070
00071     input->svuota();
00072     num++;
00073     //canRead=rs->next();
00074   }
00075
00076   MRLog() « link_S->size() «" bigrams have been read." « std::endl;
00077   std::cout« link_S->size()«" bigrams have been read." « std::endl;
00078   delete input;
00079 }
00080
00081 void FSTree::insertLink(FSTreeSequence* s){
00082   FSTreeSequence* out;
00083   for (size_t i=0;i<s->size()-1;i++){
00084     out=new FSTreeSequence(*s,i,2);
00085     (*link_S)[*out]=(*link_S)[*out]+1;
00086     delete out;
00087   }
00088 }
00089
00090 void FSTree::construct_Tree(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes){
00091   std::vector<FSTreeSequence*>* collezione;
00092   FSTreeSequence* seq;
00093   size_t num=0;
00094   rs->next();
00095   while (!rs->isAfterLast()){
00096     num++;
00097     seq=new FSTreeSequence();
00098     seq->read(rs,rowDes);
00099
00100     if (seq->size()>1){
00101       collezione=frazione(seq);
```

```
00102
00103       for (size_t i=0; i<collezione->size();i++){
00104         if (((*collezione)[i])->size()>1){
00105           insertTree(root,(*collezione)[i]);
00106         }
00107         delete (*collezione)[i];
00108       }
00109       delete collezione;
00110     }
00111     delete seq;
00112   }
00113 }
00114  std::vector<FSTreeSequence* >* FSTree::frazionа(FSTreeSequence *s){
00115   std::vector<int> vec;
00116   std::vector<FSTreeSequence*>* res = new std::vector<FSTreeSequence*>();
00117
00118
00119   //creo un vector con i riferimenti alla posizione in cui ho trovato l'inizio delle
00120   //seq lunghe due non a supp sufficiente
00121   FSTreeSequence* te;
00122   for (size_t i =0;i<s->size()-1;i++){
00123     te=new FSTreeSequence(*s,i,2);
00124     if ((*link_NS)[*te]!=0)
00125       vec.push_back(i);
00126     delete te;
00127   }
00128   //ordino il vettore di riferimenti all'inizio di sequenze non frequenti
00129   sort(vec.begin(),vec.end());
00130
00131   int nchar;
00132   //se non ci sono sequenze e la sequenza e piu lunga di 1 allora ritorno un vector in cui ho inserito
      un solo
00133   //elemento: la sequenza iniziale
00134
00135   if (vec.size()==0){
00136     if (s->size()>1){
00137       FSTreeSequence* sost=new FSTreeSequence(*s);
00138       res->push_back(sost);
00139     }
00140     return res;
00141   }
00142
00143   FSTreeSequence *temp;
00144   //inserisco la prima sottosequenza nel vettore risultato
00145   temp=new FSTreeSequence(*s,0,vec[0]+1);
00146
00147   res->push_back(temp);
00148   //se il vector vec e piu lungo di 1
00149   for (size_t i=0;i<vec.size()-1;i++){
00150     //skip di quelle posizioni consecutive
00151     if (vec[i]+1!=vec[i+1]){
00152       //numero di caratteri da prendere
00153       nchar=vec[i+1]-vec[i];
00154       //creo la sottosequenza che parte da vec[i]1 ed e lunga nchar
00155       //lunga almeno 2
00156       if (nchar>1){
00157         temp=new FSTreeSequence(*s,vec[i]+1,nchar);
00158         res->push_back(temp);
00159       }
00160     }
00161   }
00162   nchar=s->size()-(vec[vec.size()-1]+1);
00163   //se non mi trovo gia alla fine della sequenza
00164   //faccio lo stesso per l'ultima sottosequenza
00165   if (nchar>1){
00166     FSTreeSequence* last=new FSTreeSequence(*s,vec[vec.size()-1]+1,nchar);
00167     res->push_back(last);
00168
00169   }
00170   return res;
00171 }
00172
00173
00174 void FSTree::resetN_nodi(){
00175   n_nodi=0;
00176 }
00177
00178 int FSTree::getN_nodi(){
00179   return n_nodi;
00180 }
00181
00182 int FSTree::countPath(FSTreeNode* n){
00183   //la radice di tutte le sequenze dell'albero e ad altezza -1
00184   //il primo elemento e ad altezza 0
00185   FSTreeNode* temp= n;
00186   int c=0;
00187   while (temp->getParent()!=root){
```

```
00188     c++;
00189     temp=temp->getParent();
00190   }
00191   return c;
00192 }
00193
00194 void FSTree::addResult(FSTreeNode* n){
00195   FSTreeSequence* s;
00196   //setto il primo UpperLimit per prendere come prima sequenza quella lunga 2
00197   FSTreeNode* UpperLimit=n->getParent();
00198   UpperLimit=UpperLimit->getParent();
00199   FSTreeNode* temp;
00200   int c = countPath(n);
00201   temp=n;
00202   s=new FSTreeSequence();
00203   //inserisco l'elemento coda presente in tutte le sottosequenze di questo cammino
00204   s->insertHead(temp->getLabel());
00205   if (c>=max_length && max_length !=0)
00206     c=max_length-1;
00207
00208   for (int i=0;i<c;i++){//prendo tutte le sequenze di lunghezza >=2
00209     temp=temp->getParent();
00210     s->insertHead(temp->getLabel());
00211     (*result)[*s]=(*result)[*s]+n->getCount();
00212     UpperLimit=UpperLimit->getParent();//estendo l' UpperLimit per prendere la sequenza lunga |s|+1
00213   }
00214   delete s;
00215 }
00216
00217
00218
00219 void FSTree::stampa(FSTreeNode* n){
00220   if (n!=root)
00221     std::cout<<n->getLabel()<<":"<<n->getCount()<<" ";
00222   std::vector<FSTreeNode*>* temp= n->getChild();
00223   if (temp->size()==0)//se non ha figli stampo un \n e poi esco dal metodo
00224     std::cout<<std::endl;
00225   else {
00226     for (size_t i=0;i<temp->size();i++){
00227       stampa((*temp)[i]);
00228     }
00229   }
00230 }
00231
00232 void FSTree::addResults(std::vector<FSTreeNode*>* vec){
00233   FSTreeNode* temp;
00234   for (size_t i=0;i<vec->size();i++)    {
00235     temp= (*vec)[i];//inserisco ogni elemento del vettore nel risultato
00236     addResult(temp);
00237   }
00238 }
00239
00240 void FSTree::mine(){
00241   //threshold=tsl*num/100;
00242
00243   // std::vector<sequence*>::iterator m_it;
00244   std::map<FSTreeSequence, std::vector<FSTreeNode*>*, FSTreeSequence::less_sequence>::iterator m_it;
00245   //itero sulla mappa che contiene le sequenze lunghe 2 e i puntatori alle sequenze
00246   std::cout<<"prima del ciclo di mine"<<std::endl;
00247   std::cout<<"nella m_list ci sono "<<m_list.size()<<" sequenze"<<std::endl;
00248   //cout<<"nella link_NS ci sono "<<link_NS->size()<<" sequenze"<<std::endl;
00249   std::cout<<"nella link_S ci sono "<<link_S->size()<<" sequenze"<<std::endl;
00250
00251   for (m_it=m_list.begin();m_it!=m_list.end();m_it++){
00252     //addResults(m_list->getList(*(*m_it)));
00253     addResults((*m_it).second);
00254   }
00255   std::cout<<"possibili stringhe frequenti da scremare ulteriormente result.size()"<<std::endl;
00256   std::cout<<result->size()<<std::endl;
00257   //std::vector<sequence*>::iterator res_it;
00258   //std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00259   // itero sulla mappa che contiene il risultato per stamparlo
00260   //int count=0;
00261
00262   //sequenceCount* t= new sequenceCount(*result);
00263   /*
00264     std::map<sequence,int,sequence::less_sequence>* t = new
      std::map<sequence,int,sequence::less_sequence>(*result);
00265     for (res_it=t->begin();res_it!=t->end();res_it++){
00266     count=(*res_it).second;
00267     if (count>=threshold){
00268     sequence temp=(*res_it).first;
00269     temp.stampa();
00270     std::cout<<": "<<count<<std::endl;
00271     }
00272     }
00273     delete t;*/
```

```
00274   std::cout«"threshold: "«threshold«std::endl;
00275 }
00276
00277 //aggiungo la sequenza alla lista delle sequenze lunghe 2
00278 void FSTree::addList(FSTreeNode* father, FSTreeNode* children){
00279   //creo la sequenza lunga 2
00280   FSTreeSequence* m_string=new FSTreeSequence();
00281   m_string->add(father->getLabel());
00282   m_string->add(children->getLabel());
00283   //se la sequenza ha come father la radice allora esco dal metodo
00284   if (father==root) {
00285     delete m_string;
00286     return;
00287   }
00288   //se l'elemento non e gia presente lo aggiungo e creo il collegamento
00289   // inserendo il riferimento al dodo
00290   //m_list->add(m_string,children);
00291   if (m_list[*m_string]==0)
00292     m_list[*m_string]=new std::vector<FSTreeNode*>();
00293   (m_list[*m_string])->push_back(children);
00294   delete m_string;
00295 }
00296
00297
00298
00299
00300 FSTreeNode* FSTree::appendChild(FSTreeNode* r, const ItemType& t){
00301   bool notFound=1;
00302   //int i=0;
00303   FSTreeNode* nod = NULL;
00304   std::vector<FSTreeNode*>* ve = r->getChild();
00305   //Cerco l'item t della sequenza tra i figli di r
00306   for (size_t i=0;i<ve->size()&&notFound;i++){
00307     nod = (*ve)[i];
00308     if (nod->getLabel()==t){
00309       notFound=0;
00310     }
00311   }
00312   //se non l'ho trovata creo un nuovo dodo e l'ho aggiungo all albero
00313   if (notFound){
00314     //creo il nuovo nodo
00315     FSTreeNode* children = new FSTreeNode(t);
00316     n_nodi++;
00317     children->setParent(r);
00318     //aggiungo children ai figli di r
00319     std::vector<FSTreeNode*>* temp=r->getChild();
00320     temp->push_back(children);
00321
00322     children->setCount(children->getCount()+1);
00323     /*formo una sequenza e aggiungo la sequenza
00324       alla lista che attraversa l'albero e che ha come riferimento iniziale la std::mappa
00325       il metodo addList controlla che non si formino sequenze lunghe uno e controlla anche
00326       quando si arriva alla radice
00327     */
00328
00329     addList(r,children);
00330
00331     //ritorno il puntatore al dodo creato
00332     return children;
00333   }
00334   //se l'ho trovato incremento il contatore di quello trovato
00335   else{
00336     nod->setCount(nod->getCount()+1);
00337     return nod;
00338   }
00339
00340
00341 }
00342
00343
00344 FSTree::~FSTree(){
00345   //cout«"sono nel distruttore di FSTree"«std::endl;
00346   delete root;
00347   //    std::cout«"esco dal distruttore di FSTree"«std::endl;
00348   delete link_NS;
00349   delete link_S;
00350
00351   //delete s_result;
00352   //delete m_list;
00353
00354   //delete s_result;
00355   delete result;
00356
00357 }
00358
00359 double FSTree::getThreshold(){
00360   return threshold;
```

```
00361 }
00362
00363
00364 std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* FSTree::getResult(){
00365   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>::iterator res_it;
00366   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence> t(*result);
00367   std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>* t2 = new
       std::map<FSTreeSequence,int,FSTreeSequence::less_sequence>();
00368   int count;
00369   for (res_it=t.begin();res_it!=t.end();res_it++){
00370     count=(*res_it).second;
00371     if (count>=threshold){
00372       FSTreeSequence temp=(*res_it).first;
00373       (*t2)[temp]=count;
00374     }
00375   }
00376   return t2;
00377 }
00378
00379 } // namespace
```

7.229 /Users/esposito/Software/minerule/src/Algorithms/FSTree↩ Node.cpp File Reference

```
#include "minerule/Algorithms/FSTreeNode.hpp"
#include <iostream>
```

Namespaces

- namespace minerule

7.230 FSTreeNode.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/FSTreeNode.hpp"
00017 #include <iostream>
00018
00019
00020 namespace minerule {
00021
00022 FSTreeNode::FSTreeNode(const ItemType& l){
00023   child = new std::vector<FSTreeNode*>();
00024   count=0;
00025   label=l;
00026 }
00027
00028 FSTreeNode::FSTreeNode(){
00029   child = new std::vector<FSTreeNode*>();
00030   count=0;
00031 }
00032
00033 FSTreeNode::FSTreeNode(FSTreeNode* p){
00034   parent = p;
```

```
00035   count=0;
00036   child = new std::vector<FSTreeNode*>();
00037 }
00038
00039 FSTreeNode::~FSTreeNode(){
00040   FSTreeNode* temp;
00041   for (size_t i=0;i<child->size();i++){
00042     temp=(*child)[i];
00043     delete temp;
00044   }
00045   delete child;
00046
00047 }
00048
00049 const ItemType& FSTreeNode::getLabel() {
00050   return label;
00051 }
00052
00053 std::vector<FSTreeNode*>* FSTreeNode::getChild(){
00054   return child;
00055 }
00056
00057 int FSTreeNode::getCount() {
00058   return count;
00059 }
00060
00061 FSTreeNode* FSTreeNode::getParent() {
00062   return parent;
00063 }
00064
00065 void FSTreeNode::setCount(int n) {
00066   count = n;
00067 }
00068
00069 void FSTreeNode::setParent(FSTreeNode* s) {
00070   parent = s;
00071 }
00072
00073 void FSTreeNode::insertChild(FSTreeNode* s) {
00074   child->push_back(s);
00075 }
00076
00077 } // namespace
```

7.231 /Users/esposito/Software/minerule/src/Algorithms/FSTree↩ Sequence.cpp File Reference

```
#include "minerule/Algorithms/FSTreeSequence.hpp"
#include <iostream>
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
```

Namespaces

- namespace minerule

7.232 FSTreeSequence.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Dino Ienco (dino.ienco@teledetection.fr)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
```

```
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/FSTreeSequence.hpp"
00017
00018 #include <iostream>
00019 #include "minerule/Database/SourceRowColumnIds.hpp"
00020 #include "minerule/Database/SourceRow.hpp"
00021 #include "minerule/Utils/MineruleLogs.hpp"
00022
00023
00024
00025 namespace minerule {
00026
00027   FSTreeSequence::ItemVector* FSTreeSequence::getFSTreeSequence(){
00028 #warning CONTROLLARE CHE LA COPIA SIA VERAMENTE UTILE (NON SI PUO SEMPLICEMENTE RESTITUIRE seq?).
00029     return new ItemVector(*seq);
00030   }
00031
00032   FSTreeSequence::~FSTreeSequence(){
00033     delete seq;
00034   }
00035
00036   void FSTreeSequence::add(const ItemType& s){
00037     seq->push_back(s);
00038   }
00039
00040   FSTreeSequence::FSTreeSequence(const FSTreeSequence &in, size_t start, size_t n_item){
00041     seq= new ItemVector();
00042     ItemVector::const_iterator it = in.seq->begin();
00043
00044     for(size_t i=0; i<start; ++i) {
00045       ++it;
00046     }
00047
00048
00049     for (size_t i=start;(i<in.seq->size())&&(i<start+n_item);++i) {
00050       seq->push_back( *it );
00051       ++it;
00052     }
00053   }
00054
00055   void FSTreeSequence::stampa(){
00056     ItemVector::const_iterator it;
00057     for(it=seq->begin(); it!=seq->end(); ++it) {
00058       MRLog() << *it << std::endl;
00059     }
00060   }
00061
00062   size_t FSTreeSequence::size(){
00063     return seq->size();
00064   }
00065
00066    ItemType FSTreeSequence::removeHead(){
00067     ItemType res = seq->front();
00068     seq->erase(seq->begin());
00069     return res;
00070   }
00071
00072   void FSTreeSequence::insertHead(const ItemType& s){
00073     seq->insert(seq->begin(),s);
00074   }
00075
00076  std::string FSTreeSequence::toStdString(){
00077     ItemVector::const_iterator it;
00078    std::string ris="";
00079     for (it=seq->begin();it!=seq->end();it++)
00080       ris = ris+" "+(*it).getSQLData();
00081
00082     /*   std::string ris="";
00083     if (seq->size()>0)
00084       ris=(*seq)[0];
00085     for (size_t i=1;i<seq->size();i++)
00086     ris=ris+"->"+(*seq)[i]; */
00087     return ris;
00088   }
00089
00090   void FSTreeSequence::svuota(){
00091     delete seq;
00092     seq=new ItemVector();
00093   }
```

```
00094
00095   bool FSTreeSequence::operator==(FSTreeSequence& s){
00096     return *seq==*s.seq;
00097     /*
00098
00099     if (seq->size()!=s.size())
00100       return false;
00101
00102     ItemVector* vec = s.getFSTreeSequence();
00103     for (size_t i=0; i < vec->size();i++)
00104       if ((*vec)[i]!=(*seq)[i]){
00105         delete vec;
00106         return false;
00107       }
00108     delete vec;
00109     return true; */
00110
00111   }
00112
00113   void FSTreeSequence::read(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes){
00114         SourceRow hbsr(rs,rowDes);
00115         ItemType gid;
00116         if(!rs->isAfterLast())
00117                 gid = hbsr.getGroup();
00118         while (!rs->isAfterLast() && gid == hbsr.getGroup()) {
00119                 seq->push_back( hbsr.getBody() );
00120
00121                 rs->next();
00122                 if(!rs->isAfterLast())
00123                         hbsr.init(rs,rowDes);
00124         }
00125   }
00126
00127   bool FSTreeSequence::operator<(const FSTreeSequence& in) const {
00128     return lexicographical_compare(seq->begin(),
00129                                    seq->end(),
00130                                    in.seq->begin(),
00131                                    in.seq->end());
00132 /*    ItemVector* vec1 =seq;
00133     ItemVector* vec2 = in.seq;
00134     if (vec1->size()<vec2->size())
00135       return true;
00136     if (vec1->size()>vec2->size())
00137       return false;
00138
00139     size_t i=0;
00140     for (;i<vec1->size()&&((*vec1)[i]==(*vec2)[i]);i++);
00141     // for senza corpo
00142
00143     if (i==vec1->size()) return false;
00144     else return (*vec1)[i]<(*vec2)[i]; */
00145   }
00146
00147
00148 } // namespace
```

7.233 /Users/esposito/Software/minerule/src/Algorithms/IDIncremental↩ Algorithm.cpp File Reference

```
#include <memory>
#include <iterator>
#include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Utils/SQLUtils.hpp"
#include "minerule/Database/Connection.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
```

Namespaces

- namespace minerule

7.234 IDIncrementalAlgorithm.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <memory>
00017 #include <iterator>
00018 #include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
00019 #include "minerule/Utils/MineruleOptions.hpp"
00020 #include "minerule/Utils/SQLUtils.hpp"
00021 #include "minerule/Database/Connection.hpp"
00022 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00023
00024 #include <memory>
00025
00026 namespace minerule {
00027
00028         void
00029         IDIncrementalAlgorithm::getItemInfos( std::string& itemDescr, SourceRowColumnIds& hbsr ) const
      {
00030                 assert(attrList!=NULL);
00031                 ParsedMinerule::AttrVector::const_iterator it;
00032                 int n;
00033                 for(it=attrList->begin(), n=1; it!=attrList->end(); it++,n++) {
00034                         if(itemDescr!="")
00035                                 itemDescr+=",";
00036
00037                         itemDescr+=*it;
00038                         hbsr.bodyElems.push_back(n);
00039                 }
00040         }
00041
00042
00043         std::set<ItemType>*
00044         IDIncrementalAlgorithm::fillValidItems(const std::string& constraints) const {
00045                 if(constraints=="")
00046                         return NULL;
00047
00048                 std::set<ItemType>* result = new std::set<ItemType>;
00049
00050 #ifdef MRUSERWARNING
00051 #warning In the future we will assume to have a normalized database. \
00052                 Then we will need to make the following selection over the correct \
00053                 relation of the db instead of aover the current source table (which \
00054                 is needingly larger).
00055 #endif
00056                 std::string itemDescr;
00057                 SourceRowColumnIds hbsr;
00058
00059                 getItemInfos(itemDescr,hbsr);
00060
00061                 std::string query =
00062                         "SELECT DISTINCT "+itemDescr+" "+
00063                         "FROM "+minerule->getParsedMinerule().tab_source+" "+
00064                         (constraints.size()>0 ? "WHERE "+constraints : "");
00065
00066                 mrdb::Connection* con =
00067                         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00068
00069                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00070                 std::auto_ptr<mrdb::ResultSet> rs(state->executeQuery(query.c_str()));
00071
00072                 while(rs->next()) {
00073                         SourceRow sr(rs.get(), hbsr);
00074                         result->insert(sr.getBody());
00075                 };
00076
00077                 return result;
00078         }
00079
00080
00097 bool IDIncrementalAlgorithm::checkInclusion(const std::set<ItemType>& validOnes, const ItemSet&
       foundOnes) const {
```

```
00098                 ItemSet::const_iterator it;
00099                 for(it=foundOnes.begin();it!=foundOnes.end();it++) {
00100                         if(validOnes.find(*it)==validOnes.end())
00101                                 return false;
00102                 }
00103
00104                 return true;
00105         }
00106
00107         bool
00108         IDIncrementalAlgorithm::checkInValidRules(const ValidRules& validRules, Rule& r) const {
00109                 ValidRules::const_iterator it = validRules.begin();
00110                 bool found = false;
00111                 while(it!=validRules.end() && !found) {
00112                         found=
00113                                 (it->first==NULL || checkInclusion(*it->first, r.getBody())) &&
00114                                         (it->second==NULL|| checkInclusion(*it->second, r.getHead()));
00115
00116                         it++;
00117                 }
00118
00119                 return found;
00120         }
00121
00122
00123         void IDIncrementalAlgorithm::filterQueries(const ValidRules& validRules) {
00124                 OptimizerCatalogue& cat =
    MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00125
00126                 std::string resName =
    cat.getResultsetName(minerule->getOptimizationInfo().minerule.tab_result );
00127
00128                 QueryResult::Iterator qri;
00129                 qri.init(resName, minerule->getOptimizationInfo().minerule.sup,
    minerule->getOptimizationInfo().minerule.conf);
00130
00131                 MRLog() « "tab results:" « resName « std::endl;
00132                 MRLog() « "Orig supp:"   « minerule->getOptimizationInfo().minerule.sup « std::endl;
00133                 MRLog() « "Orig conf:"   « minerule->getOptimizationInfo().minerule.conf « std::endl;
00134
00135                 MRLog() « "Connection ..."«std::endl;
00136                 Connection connection;
00137                 connection.setOutTableName(minerule->getParsedMinerule().tab_result);
00138                 connection.setBodyCardinalities(minerule->getParsedMinerule().bodyCardinalities);
00139                 connection.setHeadCardinalities(minerule->getParsedMinerule().headCardinalities);
00140                 connection.useMRDBConnection(
    MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00141                 connection.createResultTables(SourceRowMetaInfo(connection.getMRDBConnection(),
    minerule->getParsedMinerule()));
00142
00143                 size_t rcount = 0;
00144                 while(qri.next()) {
00145                         Rule r;
00146                         qri.getRule(r);
00147                         if(checkInValidRules(validRules, r))
00148                                 connection.insert(r.getBody(), r.getHead(), r.getSupport(),
    r.getConfidence());
00149
00150                         rcount++;
00151                 }
00152                 MRLog() « "Processed #" « rcount « " rules."«std::endl;
00153         }
00154
00155
00156         void
00157         IDIncrementalAlgorithm::execute() {
00158                 assert(minerule->getParsedMinerule().mc!=NULL);
00159
00160                 MRLogPush("This is IDIncrementalAlgorithm. Starting...");
00161
00162                 ValidRules validRules;
00163                 list_OR_node* it;
00164                 MRLogPush("Building the list of items satisfying the constraints");
00165                 size_t conjNum=0;
00166                 for(it=minerule->getParsedMinerule().mc; it!=NULL; it=it->next) {
00167                         MRLog() « "Conjunct number:" « ++conjNum « std::endl;
00168                         std::string bodyConstraints;
00169                         std::string headConstraints;
00170                         HeadBodyPredicatesSeparator::separate(it->l_and, bodyConstraints,
    headConstraints);
00171
00172     // a little bit ugly, but useful. The fillValidItems will use the attrList
00173     // in order to select the correct portion of the database needed to build
00174     // the items in the head/body part of the rule. We need to set it to the
00175     // correct list before calling fillValidItems
00176                         attrList = &minerule->getParsedMinerule().ba;
00177                         std::set<ItemType>* validBodies = fillValidItems( bodyConstraints );
```

```
00178                         attrList = &minerule->getParsedMinerule().ha;
00179                         std::set<ItemType>* validHeads  = fillValidItems( headConstraints );
00180                         attrList = NULL;
00181                         validRules.push_back(ValidRule(validBodies,validHeads));
00182                 }
00183
00184                 MRLogPop(); // Building the list of items satisfying the constraints
00185
00186                 MRLogPush("Filtering past results...");
00187                 filterQueries(validRules);
00188                 MRLogPop(); // Filtering past results...
00189
00190   //trashing the trashable
00191                 MRLogPush("Cleaning up...");
00192                 ValidRules::const_iterator rule_it;
00193                 for(rule_it=validRules.begin();rule_it!=validRules.end();rule_it++) {
00194                         if(rule_it->first!=NULL)
00195                                 delete rule_it->first;
00196
00197                         if(rule_it->second!=NULL)
00198                                 delete rule_it->second;
00199                 }
00200                 MRLogPop(); // Cleaning up...
00201
00202
00203                 MRLogPop();  // This is IDIncrementalAlgorithm. Starting...
00204         }
00205
00206
00207 } // namespace
```

7.235 /Users/esposito/Software/minerule/src/Algorithms/IncrAlgoClasses.cpp File Reference

```
#include "minerule/Algorithms/IncrAlgoClasses.hpp"
#include "minerule/Database/ItemType.hpp"
#include <iostream>
#include <algorithm>
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Database/Connection.hpp"
#include <iterator>
```

Namespaces

- namespace minerule

7.236 IncrAlgoClasses.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Arianna Gallo (gallo.arianna@gmail.com)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/IncrAlgoClasses.hpp"
```

```
00017 #include "minerule/Database/ItemType.hpp"
00018 #include <iostream>
00019 #include <algorithm>
00020 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00021 #include "minerule/Database/Connection.hpp"
00022 #include <iterator>
00023
00024
00025 namespace minerule {
00026
00027  size_t Body::countb=0;
00028  size_t Head::counth=0;
00029
00030  Body::Body(){
00031   NRB=new RowBContainer;
00032   ItemType* tmp=new ItemType();
00033   ancestor=*tmp;
00034   countb++;
00035   delete tmp;
00036  }
00037
00038  Body::~Body(){
00039     countb--;
00040     RowBContainer::iterator it;
00041     it=NRB->begin();
00042     while(it!=NRB->end()){
00043       delete (it->second);
00044       it++;
00045     }
00046     delete NRB;
00047   }
00048
00049  Head::Head(){
00050   NR=new RowContainer;
00051   counth++;
00052  }
00053
00054  Head::~Head(){
00055     counth--;
00056     RowContainer::iterator it;
00057     it=NR->begin();
00058     while(it!=NR->end()){
00059       delete (it->second);
00060       it++;
00061     }
00062     delete NR;
00063   }
00064
00065
00066  //INSERIMENTO DI ItemSet BODY
00067
00068  //funzione ricorsiva
00069  Head* Body::insertItemSetB(ItemSet::iterator ite,
00070                             ItemSet::iterator itend, double s ){
00071   RowBContainer::iterator itr;
00072   itr=NRB->find(*ite);
00073   Head* hr=NULL;
00074   //se non c'e' lo inserisco
00075   if(itr==NRB->end()){
00076    (*NRB)[*ite]=new NodeRowB();
00077    itr=NRB->find(*ite);
00078    assert(itr!=NRB->end());
00079    //cout«"body "«itr->first«std::endl;
00080
00081    ite++;
00082    if(ite!=itend){
00083      Body* child=itr->second->setChild();
00084      child->setAncestor(itr->first);
00085      hr=child->insertItemSetB(ite,itend,s);
00086      return hr;
00087     }else itr->second->setSupp(s);
00088   }
00089   //se l'ho trovato
00090   else{
00091    //cout«"body "«itr->first«std::endl;
00092    ite++;
00093    //se ci sono ancora figli da inserire
00094    if(ite!=itend){
00095     Body* child=itr->second->getChild();
00096     //se ha gia' figli
00097     if(child!=NULL){
00098      hr=child->insertItemSetB(ite,itend,s);
00099      return hr;}
00100     //se non ha figli per Rob:es.se ho inserito A->B e poi voglio inserire AC->F A non ha figli
00101     else{
00102      Body* child=itr->second->setChild();
00103      hr=child->insertItemSetB(ite,itend,s);
```

```
00104       return hr;
00105       }
00106     }else itr->second->setSupp(s);
00107    }
00108    if(ite==itend) {
00109     Head* h=itr->second->setHead();
00110     h->setAncestor(itr->first);
00111     return h;
00112    }
00113    return hr;
00114   }
00115
00116
00117   Head* Body::insertItemSetB(ItemSet& SREV,double supp){
00118    ItemSet:: iterator ite=SREV.begin();
00119    ItemSet:: iterator itend=SREV.end();
00120    Head* h;
00121    h=insertItemSetB(ite, itend, supp);
00122    return h;
00123   }
00124
00125
00126   //INSERIMENTO DI ItemSet HEAD
00127
00128   //funzione ricorsiva
00129   void Head::insertItemSetH(ItemSet:: iterator ite,
00130                             ItemSet:: iterator itend, double supp){
00131    RowContainer::iterator itr;
00132    itr=NR->find(*ite);
00133
00134    //se non c'e' lo inserisco
00135    if(itr==NR->end()){
00136     (*NR)[*ite]=new NodeRow();
00137     itr=NR->find(*ite);
00138     itr->second->setSupp(supp);
00139     //cout«"                head "«itr->first«std::endl;
00140     //se ci sono ancora elementi nel vettore, vuol dire che devo inserire i figli
00141     ite++;
00142     if(ite!=itend){
00143       Head* tmph=itr->second-> setChild();
00144       tmph->setAncestor(itr->first);
00145       tmph->insertItemSetH(ite,itend,supp);
00146       }
00147    }
00148
00149    //se l'ho trovato
00150    else{
00151     //cout«"                head "«itr->first«std::endl;
00152     //se ci sono ancora figli da inserire
00153     ite++;
00154     if(ite!=itend){
00155      Head* child=itr->second->getChild();
00156       //se ha gia' figli
00157      if(child!=NULL)
00158       child->insertItemSetH(ite,itend,supp);
00159      //se non ha figli
00160      else{
00161       Head* child=itr->second->setChild();
00162       child->insertItemSetH(ite,itend,supp);
00163       }
00164     }else itr->second->setSupp(supp);
00165    }
00166    //itr->second->setSupp(supp);
00167  }
00168
00169   void Head::insertItemSetH(ItemSet& SREV,double supp){
00170    ItemSet:: iterator ite=SREV.begin();
00171    ItemSet:: iterator itend=SREV.end();
00172    insertItemSetH(ite,itend,supp);
00173   }
00174
00175
00176
00177   //funzione ricorsiva
00178   void Body::findBodiesInTree(ItemSet* body){
00179
00180    RowBContainer::iterator ite=NRB->begin();
00181    ItemSet::iterator itfind;
00182    while(ite!=NRB->end()){
00183      itfind=find(body->begin(),body->end(),ite->first);
00184      //se l'ho trovato
00185      if (itfind!=body->end()) {
00186        ite->second->incremSupp();
00187        //cout«"ho increm supp di body"«std::endl;
00188        Body* c=ite->second->getChild();
00189        //se ha figli
00190        if (c!=NULL) c->findBodiesInTree(body);
```

```
00191     }
00192   ite++;
00193   }
00194  }
00195
00196
00197   //funzione ricorsiva
00198  void Head::findHead(ItemSet* head){
00199   RowContainer::iterator ite=NR->begin();
00200   ItemSet::iterator itfind;
00201   //  ItemSet::iterator prova=head->begin();
00202   /*  std::cout«"elementi in cui faccio ricerca:"«std::endl;
00203   while(prova!=head->end()){
00204   std::cout«(*prova)«", ";
00205   prova++;
00206   }
00207   std::cout«std::endl;*/
00208
00209   while(ite!=NR->end()){
00210     //    std::cout«"ora cerco "«ite->first«std::endl;
00211     itfind=find(head->begin(),head->end(),ite->first);
00212     if(itfind!=head->end()) {//se l'ho trovato
00213       ite->second->incremSupp();
00214       //cout«"ho increm supp di rule"«std::endl;
00215       Head* c=ite->second->getChild();
00216       //se ha figli
00217       if (c!=NULL) {
00218         c->findHead(head);
00219       }
00220     }//else std::cout«"non trovato"«std::endl;
00221   ite++;
00222   }
00223  }
00224
00225 //questa serve per l'algo incrementale distruttivo
00226 //arrivo qui quando il body della regola non soddisfa B2, quindi non mi interessa piu' che
00227 //H2 sia soddisfatto
00228  void Head::findHead2(ItemSet* h1){
00229   RowContainer::iterator ite=NR->begin();
00230   ItemSet::iterator itfind;
00231   while(ite!=NR->end()){
00232     //cout«"esamino head "«ite->first«std::endl;
00233     itfind=find(h1->begin(),h1->end(),ite->first);
00234     if(itfind!=h1->end()) {//se l'ho trovato
00235       ite->second->decremSupp();
00236       //cout«"decremento head "«ite->first«std::endl;
00237       Head* c=ite->second->getChild();
00238       //se ha figli
00239       if (c!=NULL) {
00240         c->findHead2(h1);
00241       }
00242     }//else std::cout«"non trovato"«std::endl;
00243   ite++;
00244   }
00245  }
00246
00247 //questa serve per l'algo incrementale distruttivo
00248 //arrivo qui quando il body soddisfa la regola
00249  void Head::findHead2bis(ItemSet* h1nh2,ItemSet* h1){
00250   RowContainer::iterator ite=NR->begin();
00251   ItemSet::iterator itfind;
00252   while(ite!=NR->end()){
00253     //cout«"esamino head "«ite->first«std::endl;
00254     itfind=find(h1nh2->begin(),h1nh2->end(),ite->first);
00255     if(itfind!=h1nh2->end()) {//se l'ho trovato
00256       ite->second->decremSupp();
00257       //cout«"decremento head "«ite->first«std::endl;
00258       Head* c=ite->second->getChild();
00259       //se ha figli
00260       if (c!=NULL) {
00261         c->findHead2(h1);
00262       }
00263     }else {
00264       itfind=find(h1->begin(),h1->end(),ite->first);
00265       if(itfind!=h1->end()) {//se l'ho trovato
00266        Head* c=ite->second->getChild();
00267        //se ha figli
00268        if (c!=NULL) {
00269          c->findHead2bis(h1nh2,h1);
00270        }
00271       }
00272      }
00273   ite++;
00274   }
00275  }
00276
00277
```

```
00278
00279 //questa serve per l'algo incrementale distruttivo
00280 //funzione ricorsiva
00281  void Body::findChildInTree(ItemSet* b1, ItemSet* h1){
00282
00283   RowBContainer::iterator ite=NRB->begin();
00284   ItemSet::iterator itfind;
00285   //cout<<"esamino body "<<ite->first<<std::endl;
00286   while(ite!=NRB->end()){
00287     itfind=find(b1->begin(),b1->end(),ite->first);
00288     //se l'ho trovato
00289     if (itfind!=b1->end()) {
00290       ite->second->decremSupp();
00291       //cout<<"                        decremento body "<<ite->first<<std::endl;
00292       //cout<<"ho decrem supp di body"<<std::endl;
00293       Head* h=ite->second->getHead();
00294       if (h!=NULL) {
00295         h->findHead2(h1);
00296       }
00297       Body* c=ite->second->getChild();
00298       //se ha figli
00299       if (c!=NULL) c->findChildInTree(b1,h1);
00300     }
00301   ite++;
00302   }
00303  }
00304
00305
00306
00307
00308
00309  void Body::findRulesInTree(ItemSet* body,ItemSet* head){
00310
00311
00312    //ItemSet:: iterator itb=body->begin();
00313    //ItemSet:: iterator ity=body->end();
00314
00315   RowBContainer::iterator ite=NRB->begin();
00316   ItemSet::iterator itfind;
00317   while(ite!=NRB->end()){
00318     itfind=find(body->begin(),body->end(),ite->first);
00319     if(itfind!=body->end()) {//se l'ho trovato
00320       ite->second->incremSupp();
00321       //cout<<"ho decrem suppb da findRules.."<<std::endl;
00322       Head* h=ite->second->getHead();
00323
00324       /*      std::cout << "heads:";
00325       copy( head->begin(),
00326             head->end(),
00327             std::ostream_iterator<ItemType>(cout, " "));
00328             std::cout << "end";*/
00329
00330
00331       //cout<<h<<std::endl;
00332       if (h!=NULL) {
00333         h->findHead(head);
00334       }
00335       Body* c=ite->second->getChild();
00336       //se ha figli
00337       if (c!=NULL) {
00338         c->findRulesInTree(body,head);
00339       }
00340     }
00341   ite++;
00342   }
00343  }
00344
00345
00346 //questa serve per l'algo incrementale distruttivo
00347  void Body::findRulesInTree(ItemSet* b1,ItemSet* b1nb2,
00348                             ItemSet* h1,ItemSet* h1nh2){
00349   RowBContainer::iterator ite=NRB->begin();
00350   ItemSet::iterator itfind;
00351   while(ite!=NRB->end()){
00352     //cout<<"esamino body "<<ite->first<<std::endl;
00353     itfind=find(b1nb2->begin(),b1nb2->end(),ite->first);
00354
00355     if(itfind!=b1nb2->end()) {//se l'ho trovato
00356       ite->second->decremSupp();
00357       //cout<<"                        decremento body "<<ite->first<<std::endl;
00358       Head* h=ite->second->getHead();
00359       if (h!=NULL) {
00360         h->findHead2(h1);
00361       }
00362       Body* c=ite->second->getChild();
00363       //se ha figli
00364       if (c!=NULL) {
```

```
00365         c->findChildInTree(b1,h1);
00366       }
00367
00368     }else{//non l'ho trovato
00369       itfind=find(b1->begin(),b1->end(),ite->first);
00370       if(itfind!=b1->end()) {//se non soddisfa B1andnotB2 e soddisfa B1 allora soddisfa B2
00371         Head* h=ite->second->getHead();
00372         if (h!=NULL) {
00373           h->findHead2bis(h1nh2,h1);
00374         }
00375         Body* c=ite->second->getChild();
00376         //se ha figli
00377         if (c!=NULL) {
00378
00379           c->findRulesInTree(b1,b1nb2,h1,h1nh2);
00380         }
00381       }
00382
00383     }
00384   ite++;
00385   }
00386  }
00387
00388
00389
00390 void Head::extractRules(const std::vector<ItemType>& body,
00391                 std::vector<ItemType>& head,
00392                         double thrR, double thrB, double suppB, int ngroups,
00393                         Connection* pconnection){
00394  typedef std::map<ItemType,NodeRow*> RowContainer;
00395
00396  RowContainer::iterator it;
00397  Head* htmp;
00398  double s;
00399  double tmp;
00400  it=this->NR->begin();
00401  while (it!=this->NR->end()){
00402    htmp=it->second->getChild();
00403    //se ha figli richiamo la funzione
00404    if (htmp!=NULL){
00405       head.push_back(it->first);
00406       htmp->extractRules(body,head,thrR,thrB,suppB,ngroups,pconnection);
00407       s=it->second->getSupp();
00408       double sup=s/ngroups;
00409       tmp=(s/suppB);
00410       if((sup>=thrR)/*&&(tmp>=thrB)*/){
00411         /*        std::cout«"body: "«std::endl;
00412         std::vector<ItemType>::const_iterator itv=body.begin();
00413         while (itv!=body.end()){
00414           std::cout«" "«(*itv)«std::endl;
00415           itv++;
00416           }
00417         std::cout«"suppb: "«tmp«std::endl;
00418         std::cout«"       head: "«std::endl;
00419         itv=head.begin();
00420         while (itv!=head.end()){
00421           std::cout«"               "«(*itv)«std::endl;
00422           itv++;
00423           }
00424         std::cout«"suppr: "«sup«std::endl;
00425         */
00426         pconnection->insert(body,head,sup,tmp);
00427       }
00428    } else { //se invece non ne ha stampo la regola se conf e supp sono sufficienti
00429      head.push_back(it->first);
00430      s=it->second->getSupp();
00431      double sup=s/ngroups;
00432      tmp=(s/suppB);
00433      if((sup>=thrR)/*&&(tmp>thrB)*/){
00434        // std::cout«"body: "«std::endl;
00435        // std::vector<ItemType>::const_iterator itv=body.begin();
00436        //while (itv!=body.end()){
00437         // std::cout«" "«(*itv)«std::endl;
00438         // itv++;
00439       // }
00440       // std::cout«"suppb: "«tmp«std::endl;
00441       // std::cout«"       head: "«std::endl;
00442       // itv=head.begin();
00443       //  while (itv!=head.end()){
00444         // std::cout«"               "«(*itv)«std::endl;
00445         // itv++;
00446       // }
00447       // std::cout«"suppr: "«sup«std::endl;
00448        pconnection->insert(body,head,sup,tmp);
00449      }
00450    }
00451    head.pop_back();
```

```
00452    it++;
00453  }
00454
00455 }
00456
00457
00458
00459 void Body::extractRules(std::vector<ItemType>& body,
00460                         double thrR, double thrB, int ngroups,
00461                         Connection* pconnection){
00462  typedef std::map<ItemType,NodeRowB*> RowBContainer;
00463
00464  RowBContainer::iterator itB;
00465  Body* btmp;
00466  itB=this->NRB->begin();
00467  std::vector<ItemType> head;
00468  while (itB!=this->NRB->end()){
00469    double suppb=itB->second->getSupp();
00470    body.push_back(itB->first);
00471    Head* h=itB->second->getHead();
00472    if (h!=NULL){
00473         itB->second->getHead()->extractRules(body,head,thrR,thrB,suppb,ngroups,pconnection);
00474    }
00475    btmp=itB->second->getChild();
00476    if (btmp!=NULL){
00477      btmp->extractRules(body,thrR,thrB,ngroups,pconnection);
00478    }
00479    body.pop_back();
00480    itB++;
00481  }
00482 }
00483
00484 void Body::provaStampaLiv1(){
00485  RowBContainer::iterator it=NRB->begin();
00486  while(it!=NRB->end()){
00487    std::cout«"elemento "«it->first«" "«std::endl;
00488    std::cout«"child: "«it->second->getChild()«std::endl;
00489    std::cout«"head: "«it->second->getHead()«std::endl;
00490    it++;
00491  }
00492
00493 }
00494
00495
00496
00497
00498
00499 }//namespace
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
```

7.237 /Users/esposito/Software/minerule/src/Algorithms/Incremental↩ Algorithm.cpp File Reference

```
#include "minerule/Algorithms/IncrementalAlgorithm.hpp"
#include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
#include "minerule/Algorithms/ResultCombinator.hpp"
#include "minerule/Algorithms/ConstrTree.hpp"
#include "minerule/Algorithms/DestrTree.hpp"
```

Namespaces

- namespace minerule

7.238 IncrementalAlgorithm.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/IncrementalAlgorithm.hpp"
00017 #include "minerule/Algorithms/IDIncrementalAlgorithm.hpp"
00018 #include "minerule/Algorithms/ResultCombinator.hpp"
00019 #include "minerule/Algorithms/ConstrTree.hpp"
00020 #include "minerule/Algorithms/DestrTree.hpp"
00021
00022 namespace minerule {
00023
00024         IncrementalAlgorithm*
00025         IncrementalAlgorithm::newIncrementalAlgorithm(const OptimizedMinerule& mr) {
00026     // if mr has only ItemDependent constraints
00027                 MRLog() « "Checking if the current minerule is item dependent..." « std::endl;
00028
00029                 if( mr.hasIDConstraints() ) {
00030                         MRLog() « "The minerule is item dependent!" « std::endl;
00031                         if( mr.getOptimizationInfo().relationship==OptimizedMinerule::Combination )
00032                                 return new ResultCombinator(mr);
00033                         else
00034                                 return new IDIncrementalAlgorithm(mr);
00035                 }
00036
00037                 MRLog() « "The minerule is NOT item dependent!" « std::endl;
00038
00039                 if( mr.getParsedMinerule().mc!=NULL && mr.getParsedMinerule().mc->next==NULL ) {
00040
00041                         IncrementalAlgorithm* incrAlgo = NULL;
00042
00043                         switch(
      MineruleOptions::getSharedOptions().getOptimizations().getIncrementalAlgorithm() ) {
00044                                 case MineruleOptions::Optimizations::ConstructiveAlgo:
00045                                         incrAlgo = new ConstrTree(mr);
00046                                         break;
00047                                 case MineruleOptions::Optimizations::DestructiveAlgo:
00048                                         incrAlgo = new DestrTree(mr);
00049                                         break;
00050                                 case MineruleOptions::Optimizations::AutochooseIncrAlgo:
00051                                         incrAlgo = new ConstrTree(mr);
00052                                         break;
00053                         }
00054
00055                         return incrAlgo;
00056                 } else {
00057                         MRLog() « "The needed incremental algorithms has not been integrated" «
      std::endl
00058                                 « " to the system yet." « std::endl;
00059                         return NULL;
00060                 }
00061         }
00062
00063 } // namespace
```

7.239 /Users/esposito/Software/minerule/src/Algorithms/Mining↩ AlgorithmBase.cpp File Reference

```
#include "minerule/Algorithms/MiningAlgorithmBase.hpp"
#include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
#include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
#include "minerule/Algorithms/Care.hpp"
#include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
```

Namespaces

- namespace minerule

7.240 MiningAlgorithmBase.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/MiningAlgorithmBase.hpp"
00017 #include "minerule/Algorithms/BFSWithGidsNoCross.hpp"
00018 #include "minerule/Algorithms/BFSWithGidsAndCross.hpp"
00019 #include "minerule/Algorithms/Care.hpp"
00020 #include "minerule/Algorithms/ConstrItemSetsExtraction.hpp"
00021
00022 namespace minerule {
00023
00024 MiningAlgorithmBase*
00025 MiningAlgorithmBase::algorithmForType(AlgorithmTypes t, const OptimizedMinerule& mr) {
00026   switch(t) {
00027   case ATNone:
00028     return new MiningAlgorithm(mr);
00029   case ATBFSWithGidsNoCross:
00030     return new BFSWithGidsNoCross(mr);
00031   case ATBFSWithGidsAndCross:
00032     return new BFSWithGidsAndCross(mr);
00033   case ATCare:
00034     return new CARE(mr);
00035   case ATConstrainedItemsets:
00036     return new ConstrItemSetsExtraction(mr);
00037   default: throw MineruleException( MR_ERROR_INTERNAL, "Requested unknown mining algorithm type");
00038   }
00039 }
00040
00041 }
```

7.241 /Users/esposito/Software/minerule/src/Algorithms/ResultCombinator.cpp File Reference

```
#include "minerule/Algorithms/ResultCombinator.hpp"
#include "minerule/Result/QueryResult.hpp"
```

Namespaces

- namespace minerule

7.242 ResultCombinator.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/ResultCombinator.hpp"
00017 #include "minerule/Result/QueryResult.hpp"
00018
00019 namespace minerule {
00020
00021         void
00022         ResultCombinator::execute() {
00023                 MRLogPusher log1("ResultCombinator incremental algorithm starting...");
00024
00025                 if( minerule==NULL )
00026                         throw MineruleException(MR_ERROR_INTERNAL,
00027                                 "ResultCombinator::execute called, but the "
00028                                 "optimized minerule is NULL");
00029
00030                 const std::vector<ParsedMinerule>& minerules=
00031                         minerule->getOptimizationInfo().minerulesToCombine;
00032                 const GAQueryCombinator::QueryOrList& formula =
00033                         minerule->getOptimizationInfo().combinationFormula;
00034
00035                 QueryResult::ResultSet<QueryResult::SortBodyHead> result;
00036
00037                 GAQueryCombinator::QueryOrList::const_iterator OrIt;
00038                 for( OrIt=formula.begin();
00039                 OrIt!=formula.end();
00040                 OrIt++) {
00041                         MRLogPusher log2("Analyzing disjunct");
00042                         if(OrIt!=formula.begin()) {
00043                                 MRLog()<<"Analyzing next conjunct"<<std::endl;
00044                         }
00045
00046                         GAQueryCombinator::QueryAndList::const_iterator AndIt =
00047                                 OrIt->begin();
00048
00049                         QueryResult::ResultSet<QueryResult::SortBodyHead> curConj;
00050
00051                         MRLogPush("Loading first query from disk");
00052                         if( AndIt!=OrIt->end() ) {
00053                                 curConj.load( minerules[*AndIt].tab_result );
00054                                 AndIt++;
00055                         }
00056                         MRLogPop();
00057
00058                         MRLog() << "Current conjunct current result set size:"
00059                                 << curConj.size() << std::endl;
00060
00061                         MRLog() << "Intersecting with other queries:" << std::endl;
00062                         for( ;  AndIt!=OrIt->end();  AndIt++) {
00063                                 MRLogPusher log3("Analyzing conjunct");
00064
00065                                 MRLogPush("Loading query from disk.");
00066                                 QueryResult::ResultSet<QueryResult::SortBodyHead>  q;
00067                                 q.load( minerules[*AndIt].tab_result );
00068                                 MRLogPop();
00069
00070                                 MRLogPush("Intersecting...");
00071                                 curConj.inplace_intersect(q);
00072                                 MRLogPop();
00073
00074                                 MRLog() << "Current conjunct current result set size:"
00075                                         << curConj.size() << std::endl;
00076                         }
00077
00078                         MRLogPush("Merging...");
00079                         result.inplace_union(curConj);
00080                         MRLogPop();
00081
00082                         MRLog() << "Current result set size:"  << result.size() << std::endl;
```

```
00083                 }
00084
00085                 result.save(minerule->getParsedMinerule());
00086         }
00087
00088
00089 } // namespace
```

7.243 /Users/esposito/Software/minerule/src/Algorithms/STSMiner.cpp File Reference

```
#include "minerule/Algorithms/STSMiner.hpp"
```

Namespaces

- namespace minerule

7.244 STSMiner.cpp

Go to the documentation of this file.

```
00001 /*************************************/
00002 /*             STSequences miner         */
00003 /*************************************/
00004
00005 /* La seguente classe crea un oggetto STSMiner il quale data la MINE SEQUENCE in input elabora
00006 in automatico delle query SQL per estrarre le sequenze frequenti dal db delegandogli così il grosso
       del lavoro*/
00007
00008 /*
00009 L'algoritmo accetta un attributo per l'ordinamento ed un attributo per gli elementi e restituisce
00010 sequenze di coppie <el_attr,ord_attr> che rispettano il vincolo del supporto e del gap inteso come gap
00011 tra i valori dell'attributo di ordinamento tra due elementi(coppie) successivi della sequenza.
00012
00013 QUERY SYNTAX:
00014
00015 MINE SEQUENCE l_min..l_max dest_table AS
00016 SELECT [DISTINCT] bodyAttrList, SUPPORT
00017 WITH GAP min..max
00018 FROM table
00019 [HAVING CONDITIONS]
00020 [WHERE SQL_BOOL_COND]
00021 GROUP BY groupAttrList
00022 ORDER BY ordAttrList
00023
00024 CONDITIONS -> CONDITIONS AND CONDITIONS | ( CONDITIONS ) | COND
00025 COND -> DISTANCE RANGE | BEM_CLAUSE RANGE
00026 DISTANCE -> FUN_NAME(ELEM1,ELEM2)
00027 RANGE -> BETWEEN const1 AND const2 | OP const
00028 OP -> = | > | < | >= | <= | !=
00029 BEM_CLAUSE -> BEGIN.ATTR | END.ATTR | [MIN..MAX] MID.ATTR
00030 ATTR è un qualunque attributo della table sorgente
00031 SQL_BOOL_COND-> ....
00032 */
00033
00034 #include "minerule/Algorithms/STSMiner.hpp"
00035
00036 namespace minerule {
00037
00038 /*****************************************
00039  subquery= (SELECT row_number() OVER(ORDER BY trackid,sighting_event_id) AS rn, trackid,
       subarea_of_reference, sighting_event_id
00040                     FROM (SELECT trackid, subarea_of_reference , sighting_event_id
00041                           FROM mission_M_003
00042                           UNION
00043                               (SELECT (SELECT MIN(trackid)-1 FROM mission_M_00i) AS trackid,
00044                                       (SELECT MIN(subarea_of_reference) FROM mission_M_00i) AS
       subarea_of_reference,
```

```
00045                                       (SELECT MIN(sighting_event_id) FROM mission_M_00i) AS
      sighting_event_id
00046                               UNION
00047                               SELECT  (SELECT MAX(trackid)+1 FROM mission_M_00i) AS trackid,
00048                                       (SELECT MAX(subarea_of_reference) FROM mission_M_00i) AS
      subarea_of_reference,
00049                                       (SELECT MAX(sighting_event_id) FROM mission_M_00i) AS
      sighting_event_id)
00050                           ) AS SRC_SELECTION
00051                           GROUP BY trackid, subarea_of_reference, sighting_event_id
00052                           ORDER BY trackid,sighting_event_id
00053                     )
00054
00055 INSERT INTO test1_Source
00056
00057 SELECT pos_not_distinct.trackid, pos_not_distinct.subarea_of_reference, pos_not_distinct.pos,
      pos_distinct.pos
00058 FROM
00059 (SELECT trackid, subarea_of_reference,  row_number() OVER(PARTITION BY trackid ORDER BY trackid,
      sighting_event_id) AS POS , sighting_event_id
00060 FROM mission_M_00i
00061 GROUP BY trackid, subarea_of_reference, sighting_event_id
00062 ORDER BY trackid,sighting_event_id) AS pos_not_distinct
00063
00064 JOIN     <--iff DISTINCT=true
00065
00066 ( SELECT  _s1.trackid,  _s1.subarea_of_reference,  _s2.sighting_event_id_MIN,
      _s1.sighting_event_id_MAX, _s1.pos
00067  FROM (SELECT s.*,row_number() OVER(PARTITION BY  s.trackid) AS POS
00068        FROM (SELECT  s1.trackid, s1.subarea_of_reference,  s1.sighting_event_id AS
      sighting_event_id_MAX
00069              FROM subquery AS s1,
00070                   subquery AS s2
00071                    WHERE s1.rn+1=s2.rn
00072                        AND (
00073                                  ( (  s1.trackid=s2.trackid) AND (
      s1.subarea_of_reference!=s2.subarea_of_reference ))
00074                             OR ( s1.trackid!=s2.trackid)
00075                             )
00076                    ORDER BY  s1.trackid, s1.sighting_event_id
00077              ) AS s
00078       ) AS _s1
00079
00080       JOIN
00081
00082      (SELECT s.*,row_number() OVER(PARTITION BY  s.trackid) AS POS
00083       FROM (SELECT  s2.trackid, s2.subarea_of_reference,  s2.sighting_event_id AS
      sighting_event_id_MIN
00084             FROM subquery AS s1,
00085                  subquery AS s2
00086                     WHERE s1.rn+1=s2.rn
00087                               AND (
00088                                     ( (  s1.trackid=s2.trackid) AND (
      s1.subarea_of_reference!=s2.subarea_of_reference ))
00089                                    OR ( s1.trackid!=s2.trackid)
00090                                   )
00091                    ORDER BY  s1.trackid, s1.sighting_event_id
00092            ) AS s
00093     ) AS _s2
00094
00095      ON  _s1.trackid=_s2.trackid
00096        AND _s1.subarea_of_reference=_s2.subarea_of_reference
00097        AND _s1.pos=_s2.pos
00098      ORDER BY trackid,pos
00099 ) AS pos_distinct
00100
00101 ON
00102 pos_not_distinct.trackid=pos_distinct.trackid
00103 AND  pos_not_distinct.subarea_of_reference=pos_distinct.subarea_of_reference
00104 AND  pos_not_distinct.sighting_event_id>=pos_distinct.sighting_event_id_MIN
00105 AND pos_not_distinct.sighting_event_id<=pos_distinct.sighting_event_id_MAX;
00106   ****************************************/
00107
00114 bool STSMiner::extract1Sequences() {
00115
00116     std::string sqlQuery;
00117     std::string src_schema = "";
00118     std::string attr_string = "";
00119     std::string attr_string2 = "";
00120
00121     for(int i = 0; i < attr_list.size(); ++i){
00122         src_schema += attr_list[i] + " " + get_type(attr_list[i]) + ", " ;
00123         if(containedIn(attr_list[i],pm.ga) || containedIn(attr_list[i],pm.oa) ||
      containedIn(attr_list[i],pm.ba))
00124             attr_string += ", " + attr_list[i];
00125         attr_string2 += attr_list[i] + ", ";
00126     }
```

```
00127
00128     std::string o_not_b="";
00129     for(int i = 0; i < pm.oa.size(); ++i)
00130         if(!containedIn(pm.oa[i],attr_list))
00131             o_not_b += ", " + pm.oa[i];
00132
00133     std::string src_selection= "(SELECT ";
00134     if(containedIn(attr_list[0],pm.ga)
00135       || containedIn(attr_list[0],pm.oa)
00136       || containedIn(attr_list[0],pm.ba))
00137         src_selection+= attr_list[0];
00138     for(int i = 1; i < attr_list.size(); ++i)
00139         if(containedIn(attr_list[i],pm.ga) || containedIn(attr_list[i],pm.oa) ||
      containedIn(attr_list[i],pm.ba))
00140             src_selection+= ", "+attr_list[i];
00141     src_selection+= " "+o_not_b+" FROM "+pm.tab_source+" UNION "
00142             + " ("+union_rows()+")) AS SRC_SELECTION";//le due righe fittizie di cui non dobbiamo più
      fare la insert
00143
00144     sqlQuery = "CREATE TABLE " + pm.tab_result + "_Source(" + src_schema + " POS INTEGER";
00145
00146     if(DISTINCT)
00147         sqlQuery += ", POS_D INTEGER);";
00148     else
00149         sqlQuery += ");";
00150
00151     execQuery(sqlQuery); //creo la src
00152
00153     //la select di cui fare la insert con logica not_distinct!
00154     sqlQuery= "SELECT " + attr_string2 + " row_number() OVER(PARTITION BY "
00155             + groupAttrList + " ORDER BY " + groupAttrList + ", " + ordAttrList+ ") AS POS ";
00156     if(DISTINCT)
00157         sqlQuery += o_not_b;
00158     sqlQuery +=  " FROM " + pm.tab_source;
00159
00160     if(pm.filter_condition.size()>0) {
00161         sqlQuery += " WHERE " + pm.filter_condition;
00162         if(DISTINCT){
00163             //nel caso distinct per calcolare la tabella pos_distinct mi serve aggiungere due righe
      fittizie
00164             //che nella tabella pos_distinct non compaiono ma per non includerle in questa aggiungo
      una condizione
00165             //che esclude la track minima e la track massima, cioè quelle che aggiungo
00166             for(int i = 0; i < pm.ga.size(); ++i) {
00167                 sqlQuery += " AND " + pm.ga[i] + ">(SELECT MIN(" + pm.ga[i] + ") FROM " +
      src_selection + ")";
00168                 sqlQuery += " AND " + pm.ga[i] + "<(SELECT MAX(" + pm.ga[i] + ") FROM " +
      src_selection + ")";
00169             }
00170         }
00171     }
00172     sqlQuery += " GROUP BY ";
00173     for(int i = 0; i < attr_list.size()-1; ++i)
00174         sqlQuery += attr_list[i] + ", ";
00175     sqlQuery += attr_list[attr_list.size()-1] + o_not_b;
00176     sqlQuery += " ORDER BY " + groupAttrList + ","+ordAttrList;
00177
00178     std::string pos_not_distinct = " (" + sqlQuery + ") AS pos_not_distinct ";
00179     std::string pos_distinct = "";
00180
00181     //se siamo con logica distinct dobbiamo fare il join con la tabella che incrementa la pos al
      cambio di body
00182     if(DISTINCT) {
00183         sqlQuery= "SELECT s.*,row_number() OVER(PARTITION BY ";
00184         for(int i = 0; i < pm.ga.size()-1; ++i)
00185             sqlQuery += " s."+pm.ga[i]+", ";
00186         sqlQuery += " s."+pm.ga[pm.ga.size()-1]+") AS POS FROM (SELECT ";
00187
00188         std::string select_s1=sqlQuery;
00189         std::string select_s2=sqlQuery;
00190
00191         for(int i = 0; i < pm.ga.size(); ++i){
00192             select_s1 += " s1."+pm.ga[i]+",";
00193             select_s2 += " s2."+pm.ga[i]+",";
00194         }
00195         for(int i = 0; i < pm.ba.size(); ++i){
00196             select_s1 += " s1."+pm.ba[i]+", ";
00197             select_s2 += " s2."+pm.ba[i]+", ";
00198         }
00199         for(int i = 0; i < pm.oa.size()-1; ++i){
00200             select_s1 += " s1."+pm.oa[i]+" AS "+pm.oa[i]+"_MAX, ";
00201             select_s2 += " s2."+pm.oa[i]+" AS "+pm.oa[i]+"_MIN, ";
00202         }
00203         select_s1 += " s1."+pm.oa[pm.oa.size()-1]+" AS "+pm.oa[pm.oa.size()-1]+"_MAX FROM ";
00204         select_s2 += " s2."+pm.oa[pm.oa.size()-1]+" AS "+pm.oa[pm.oa.size()-1]+"_MIN FROM ";
00205
00206         std::string sqlSubQuery = "(SELECT row_number() OVER(ORDER BY "
```

```
00207                 + groupAttrList+","+ordAttrList
00208                 + ") AS rn" + attr_string + o_not_b
00209                 + "  FROM " + src_selection;
00210         sqlSubQuery += " GROUP BY ";
00211         for(int i = 0; i < pm.ga.size(); ++i)
00212             sqlSubQuery+= pm.ga[i]+", ";
00213         for(int i = 0; i < pm.ba.size(); ++i)
00214             if(!containedIn(pm.ba[i],pm.ga) && !containedIn(pm.ba[i],pm.oa))
00215                 sqlSubQuery+= pm.ba[i]+", ";
00216         sqlSubQuery+= pm.oa[0];
00217         for(int i = 1; i < pm.oa.size() ; ++i)
00218             if(!containedIn(pm.oa[i],pm.ga))
00219                 sqlSubQuery+= ", "+pm.oa[i];
00220 
00221         sqlSubQuery += " ORDER BY " + groupAttrList+","+ordAttrList + ")" ;
00222 
00223         sqlQuery = sqlSubQuery+" AS s1, ";
00224         sqlQuery += sqlSubQuery+" AS s2 WHERE s1.rn+1=s2.rn AND ((";
00225         sqlQuery += " ( ";
00226         for(int i = 0; i < pm.ga.size()-1; ++i)
00227             sqlQuery += " s1."+pm.ga[i]+"=s2."+pm.ga[i]+" AND ";
00228         sqlQuery += " s1."+pm.ga[pm.ga.size()-1]+"=s2."+pm.ga[pm.ga.size()-1];
00229         sqlQuery += ") AND ( ";
00230 
00231         for(int i = 0; i < pm.ba.size()-1; ++i)
00232             sqlQuery += " s1."+pm.ba[i]+"!=s2."+pm.ba[i]+" OR ";
00233         sqlQuery += " s1."+pm.ba[pm.ba.size()-1]+"!=s2."+pm.ba[pm.ba.size()-1]+" )) OR (";
00234 
00235         for(int i = 0; i < pm.ga.size()-1; ++i)
00236             sqlQuery += " s1."+pm.ga[i]+"!=s2."+pm.ga[i]+" OR ";
00237         sqlQuery += " s1."+pm.ga[pm.ga.size()-1]+"!=s2."+pm.ga[pm.ga.size()-1]+"))";
00238 
00239         sqlQuery += " ORDER BY ";
00240         for(int i = 0; i < pm.ga.size(); ++i)
00241             sqlQuery += " s1."+pm.ga[i]+",";
00242         for(int i = 0; i < pm.oa.size()-1; ++i)
00243             sqlQuery += " s1."+pm.oa[i]+", ";
00244         sqlQuery += " s1."+pm.oa[pm.oa.size()-1];
00245 
00246         sqlQuery += ") AS s ";
00247 
00248         select_s1 += sqlQuery;
00249         select_s2 += sqlQuery;
00250 
00251         select_s1 = "("+select_s1+") AS _s1";
00252         select_s2 = "("+select_s2+") AS _s2";
00253 
00254         sqlQuery =" SELECT ";
00255         //tutto da _s1 e da _s2 solo oa[i]_MIN
00256 
00257         for(int i = 0; i < pm.ga.size(); ++i)
00258             sqlQuery += " _s1."+pm.ga[i]+", ";
00259 
00260         for(int i = 0; i < pm.ba.size() ; ++i)
00261             sqlQuery += " _s1."+pm.ba[i]+", ";
00262 
00263         for(int i = 0; i < pm.oa.size(); ++i) {
00264             sqlQuery += " _s2."+pm.oa[i]+"_MIN, ";
00265             sqlQuery += " _s1."+pm.oa[i]+"_MAX, ";
00266         }
00267         sqlQuery += "_s1.pos FROM "+select_s1+" JOIN "+select_s2+" ON ";
00268 
00269         for(int i = 0; i < pm.ga.size(); ++i)
00270             sqlQuery += " _s1."+pm.ga[i]+"=_s2."+pm.ga[i];
00271 
00272         for(int i = 0; i < pm.ba.size(); ++i)
00273             sqlQuery += " AND _s1."+pm.ba[i]+"=_s2."+pm.ba[i];
00274 
00275         sqlQuery += " AND _s1.pos=_s2.pos ORDER BY "+groupAttrList+",pos ";
00276 
00277         pos_distinct = " (" + sqlQuery + ") AS pos_distinct ";
00278 
00279         //QUERY FINALE
00280         sqlQuery = "INSERT INTO " + pm.tab_result + "_Source SELECT ";
00281         for(int i = 0; i < attr_list.size(); ++i)
00282             sqlQuery += "pos_not_distinct."+attr_list[i]+", ";
00283         sqlQuery += "pos_not_distinct.pos, pos_distinct.pos FROM "
00284                 + pos_not_distinct + "JOIN " + pos_distinct + " ON ";
00285         for(int i = 0; i < pm.ga.size(); ++i)
00286             sqlQuery += " pos_not_distinct." + pm.ga[i] + "=pos_distinct." + pm.ga[i] + " AND ";
00287         for(int i = 0; i < pm.ba.size(); ++i)
00288             sqlQuery += " pos_not_distinct." + pm.ba[i] + "=pos_distinct." + pm.ba[i] + " AND ";
00289         for(int i = 0; i < pm.oa.size()-1; ++i)
00290             sqlQuery += " pos_not_distinct." + pm.oa[i] + ">=pos_distinct." + pm.oa[i]
00291                     + "_MIN AND pos_not_distinct." + pm.oa[i] + "<=pos_distinct." + pm.oa[i] + "_MAX
    AND";
00292         sqlQuery += " pos_not_distinct." + pm.oa[pm.oa.size()-1] + ">=pos_distinct." +
```

```
      pm.oa[pm.oa.size()-1]
00293                 + "_MIN AND pos_not_distinct." + pm.oa[pm.oa.size()-1]
00294                 + "<=pos_distinct." + pm.oa[pm.oa.size()-1] + "_MAX;";
00295     }//end if DISTINCT
00296     else
00297         sqlQuery = "INSERT INTO "+pm.tab_result+"_Source " + sqlQuery;
00298
00299     //eseguiamo la query main
00300     execQuery(sqlQuery);
00301
00302     bool at_least_one=!emptyTable(pm.tab_result + "_Source");
00303     sqlQuery="";
00304
00305     if(pm.seq_bem_vect.size()>0 && at_least_one)
00306         at_least_one = !sourceFilter();
00307
00308     if(at_least_one) {
00309         dropTable(pm.tab_result + "_Seq1");
00310         dropTable(pm.tab_result + "_matcher");
00311
00312         sqlQuery= "CREATE TABLE " + pm.tab_result + "_Seq1(ID SERIAL PRIMARY KEY ";
00313         for(int i = 0; i < pm.ba.size(); ++i)
00314             sqlQuery += ", " + pm.ba[i] + " " + get_type(pm.ba[i]);
00315         sqlQuery += ");";
00316         sqlQuery += "CREATE TABLE " + pm.tab_result + "_matcher(SEQ_ID numeric";
00317         for(int i = 0; i < pm.ga.size(); ++i)
00318             sqlQuery += ", "+pm.ga[i]+" "+ get_type(pm.ga[i]);
00319         sqlQuery += "); CREATE INDEX " + pm.tab_result + "_Source_full_index ON "
00320                 + pm.tab_result + "_Source(" + bodyAttrList + ",pos);";
00321         execQuery(sqlQuery);
00322         setSeqIDProp();
00323         return false;
00324     }
00325     return true;
00326 }
00327
00332 bool STSMiner::sourceFilter() {
00333     if(pm.seq_bem_vect.size()==0)
00334         return emptyTable(pm.tab_result + "_Source");
00335
00336     std::string sq = "DELETE FROM "+ pm.tab_result + "_Source WHERE (" + groupAttrList + ") NOT IN(
      SELECT";
00337     for(int i = 0; i < pm.ga.size()-1; ++i)
00338         sq += " s1." + pm.ga[i] + ",";
00339     sq += " s1." + pm.ga[pm.ga.size()-1] + " FROM ";
00340     for(int i=0;i<pm.seq_bem_vect.size();++i){
00341         sq += pm.tab_source + " s"+Converter(i+1).toString();
00342         if(i<pm.seq_bem_vect.size()-1) sq += ", ";
00343     }
00344     sq += " WHERE ";
00345     for(int i = 0; i < pm.ga.size(); ++i) {
00346         for(int j = 1; j<pm.seq_bem_vect.size(); ++j)
00347             sq += " s1." + pm.ga[i] + "=s"+Converter(j+1).toString()+"." + pm.ga[i] + " AND ";
00348     }
00349     for(int i = 0; i < pm.seq_bem_vect.size(); ++i) {
00350         Bem_cond* it = pm.seq_bem_vect[i];
00351         do {
00352             //cerchiamo l'attr per capire se è un varchar e aggiungere gli apostrofi
00353             std::string tmp = it->attr;
00354             bool varchar_type = false;
00355             if(textAttr(it->attr))
00356                 varchar_type = true;
00357             if(varchar_type){
00358                 if(it->op.compare("BETWEEN") == 0) {
00359                     unsigned pos = tmp.find("AND");
00360                     std::string const1 = tmp.substr(0,pos-1);
00361                     std::string const2 = tmp.substr(pos+4);
00362                     tmp = "'" + const1 + "' AND '" + const2 + "'";
00363                 }
00364                 else tmp = "'" + it->val + "'";
00365             }
00366             else tmp = it->val;
00367             sq += " s" + Converter(i+1).toString() + "." + it->attr + " " + it->op + " " + tmp;
00368             it = it->and_c;
00369             if(it != NULL)
00370                 sq += " AND ";
00371         } while(it);
00372         if(i < pm.seq_bem_vect.size()-1)
00373             sq += " AND ";
00374     }
00375     sq += ");";
00376
00377     execQuery(sq);
00378
00379     return emptyTable(pm.tab_result + "_Source");
00380 }
00381
```

```
00387 std::string STSMiner::union_rows() {
00388
00389     std::string insert_q_min="SELECT ";
00390     std::string insert_q_max="SELECT ";
00391     bool semicolon= false;
00392
00393     for(int i=0; i<attr_list.size(); ++i) {
00394         if(semicolon) {
00395             insert_q_min += ",";
00396             insert_q_max += ",";
00397             semicolon= false;
00398         }
00399         if(containedIn(attr_list[i],pm.ga)) {
00400             if(!textAttr(attr_list[i])) {
00401                 insert_q_min += "(SELECT MIN("+attr_list[i]+")-1 FROM "+pm.tab_source+") AS
      "+attr_list[i];
00402                 insert_q_max += "(SELECT MAX("+attr_list[i]+")+1 FROM "+pm.tab_source+") AS
      "+attr_list[i];
00403             }
00404             else {
00405                 insert_q_min += "(SELECT SUBSTRING(MIN("+attr_list[i]+") FROM 1 FOR CHAR_LENGTH(MIN("
00406                         +attr_list[i]+"))-1) FROM "+pm.tab_source+") AS "+attr_list[i];
00407                 insert_q_max += "(SELECT SUBSTRING(MAX("+attr_list[i]+") FROM 1 FOR
     CHAR_LENGTH(MAX("+attr_list[i]
00408                         +"))-1)|| 'Z' FROM "+pm.tab_source+") AS "+attr_list[i];
00409             }
00410             semicolon= true;
00411         }
00412         else if(containedIn(attr_list[i],pm.ba))
00413         {
00414             insert_q_min += "(SELECT MIN("+attr_list[i]+") FROM "+pm.tab_source+") AS "+attr_list[i];
00415             insert_q_max += "(SELECT MAX("+attr_list[i]+") FROM "+pm.tab_source+") AS "+attr_list[i];
00416             semicolon= true;
00417         }
00418     }
00419
00420     for(int i = 0; i < pm.oa.size() ; ++i) {
00421         if(!containedIn(pm.oa[i],attr_list)) {
00422             if(semicolon) {
00423                 insert_q_min += ",";
00424                 insert_q_max += ",";
00425             }
00426             insert_q_min += "(SELECT MIN("+pm.oa[i]+") FROM "+pm.tab_source+") AS "+pm.oa[i];
00427             insert_q_max += "(SELECT MAX("+pm.oa[i]+") FROM "+pm.tab_source+") AS "+pm.oa[i];
00428
00429         }
00430     }
00431
00432     return insert_q_min+" UNION "+insert_q_max;
00433 }
00434
00435 /*********************************
00436 INSERT INTO example_Seq1(subarea_id)
00437 SELECT subarea_id
00438 FROM example_Source
00439 GROUP BY subarea_id
00440 HAVING COUNT( DISTINCT (trackid)) >=0.02;
00441 CREATE INDEX example_Seq1_index ON example_Seq1(ID,subarea_id);
00442   *******************************/
00443 int STSMiner::pruning1Seq(){
00444
00445         std::string sqlQuery = "INSERT INTO " + pm.tab_result + "_Seq1(" + bodyAttrList + ") SELECT "
00446                 + bodyAttrList + " FROM "+ pm.tab_result + "_Source GROUP BY " + bodyAttrList;
00447         if(pm.sup>(1.0/grpId_count))
00448             sqlQuery  += " HAVING COUNT(DISTINCT GRPID) >=" +
      Converter(pm.sup*grpId_count).toString();
00449
00450         sqlQuery += "; CREATE INDEX " + pm.tab_result + "_Seq1_index ON " + pm.tab_result +
     "_Seq1(ID," + bodyAttrList + ");";
00451         execQuery(sqlQuery);
00452
00453         mrdb::ResultSet* r = execQr("SELECT MAX(ID) FROM " + pm.tab_result + "_Seq1");
00454         if(r->next())
00455             bvptr_length = r->getInt(1);
00456         else
00457             bvptr_length = 0;
00458
00459         delete r;
00460
00461         return bvptr_length;
00462 }
00463
00464
00465 /*********************************
00466   SELECT s1.id,s_1.groupAttr[, s_1.bodyAttr]
00467   FROM Source s_1 JOIN Seq1 s1 ON
00468     s_1.bodyAttr = s1.bodyAttr
```

```
00469   GROUP BY s1.id[, s_1.bodyAttr]
00470   ORDER BY s1.id;
00471  ********************************/
00477 void STSMiner::createBitVectors() {
00478
00479     int id_old= -1;
00480     int seq_id_i;
00481     int id;
00482     std::string query;
00483
00484     for(int i = 0; i < bvptr_length; ++i)
00485        bvptr.push_back(new BitString(max_trackid-min_trackid+1));
00486
00487     if(context_free_BEM) {
00488         for(int i = 0; i < bvptr_length; ++i)
00489             bem_vect.push_back(new BitString(bem_vect_length));
00490         if(active_count) {
00491             std::vector<std::pair<int,int> > tmp;
00492             for(int i = 0; i<bem_vect_length-2; ++i)
00493                 tmp.push_back(std::pair<int,int>(0,0));
00494             for(int i = 0; i < bvptr_length; ++i)
00495                 mid_count.push_back(tmp);
00496         }
00497     }
00498     query ="SELECT s1.id, s_1.GRPID";
00499     if(context_free_BEM)
00500         for(int i = 0; i < pm.ba.size(); ++i)
00501             query += ", s_1." + pm.ba[i];
00502     query += " FROM " + pm.tab_result + "_Source s_1 JOIN " + pm.tab_result + "_Seq1 s1 ON ";
00503     for(int i = 0; i < pm.ba.size()-1; ++i)
00504         query += " s_1." + pm.ba[i] + "=s1." + pm.ba[i] + " AND ";
00505     query += " s_1." + pm.ba[pm.ba.size()-1] + "=s1." + pm.ba[pm.ba.size()-1]
00506             + " GROUP BY s1.id,s_1.GRPID";
00507     if(context_free_BEM)
00508         for(int i = 0; i < pm.ba.size(); ++i)
00509             query += ",s_1." + pm.ba[i];
00510     query += " ORDER BY s1.id";
00511     mrdb::ResultSet* ris =execQr(query);
00512
00513     while(ris->next()) {
00514
00515         id = ris->getInt(1);
00516         seq_id_i= ris->getInt(2)-min_trackid;
00517
00518         if(context_free_BEM && id_old!=id) {
00519             std::vector<std::string> attrVect;
00520             for(int i = 2; i < 3+pm.ba.size(); ++i)
00521                 attrVect.push_back(ris->getString(i));
00522             initbemvect(id,attrVect);
00523         }
00524         if(((context_free_BEM && bem_vect[id-1]->count(true)==bemCF_constraints) || !context_free_BEM)
00525                 && !context_dep_BEM)
00526                 (bvptr[id-1])->set(seq_id_i,true);
00527
00528         id_old=id;
00529     }
00530     //USCITI DAL CICLO DEVO ANCORA SETTARE IL BIT PER L'ULTIMA COPPIA grpId,elId!
00531     if( id_old!=-1 &&
00532         ((context_free_BEM && bem_vect[id-1]->count(true)==bemCF_constraints) || !context_free_BEM)
00533             && !context_dep_BEM)
00534             (bvptr[id-1])->set(seq_id_i,true);
00535
00536     delete ris;
00537
00538     if(context_dep_BEM)
00539         createBitVectors_context_dep();
00540 }
00541
00542 void STSMiner::createBitVectors_context_dep() {
00543
00544     std::vector<std::pair<int, int>* > tmp_pos_1;
00545     std::vector<Constraints*> c_v;
00546     std::string query;
00547
00548     query ="SELECT s1.id, s_1.POS ";
00549     for(int i = 0; i < attr_list.size(); ++i)
00550         query += ",s_1." + attr_list[i];
00551
00552     query += " FROM " + pm.tab_result + "_Source s_1 JOIN " + pm.tab_result + "_Seq1 s1 ON ";
00553
00554     for(int i = 0; i < pm.ba.size()-1; ++i)
00555             query += " s_1." + pm.ba[i] + "=s1." + pm.ba[i] + " AND ";
00556
00557     query += " s_1." + pm.ba[pm.ba.size()-1] + "=s1." + pm.ba[pm.ba.size()-1]
00558             + " ORDER BY s1.id,s_1.GRPID,s_1.pos";
00559
00560     mrdb::ResultSet* ris = execQr(query);
```

```
00561
00562     int id_old= -1;
00563
00564     while(ris->next()) {
00565
00566         int id = ris->getInt(1);
00567         int pos = ris->getInt(2);
00568
00569         if(id>id_old && id_old>0) {
00570             pos_lists_singleton.push_back(tmp_pos_l);
00571             tmp_pos_l.clear();
00572             constr_vect_singleton.push_back(c_v);
00573             c_v.clear();
00574         }
00575         id_old=id;
00576
00577         int seq_id_i = ris->getInt(3)-min_trackid;
00578         tmp_pos_l.push_back(new std::pair<int,int>(seq_id_i+min_trackid, pos));
00579
00580         bool check_bem_cd= false;
00581         std::vector<std::string> attrVect;
00582         for(int i = 3; i < 3+attr_list.size(); ++i)
00583             attrVect.push_back(ris->getString(i));
00584         c_v.push_back(init_c_v(attrVect, &check_bem_cd));
00585         if(((context_free_BEM && bem_vect[id-1]->count(true)==bemCF_constraints) || !context_free_BEM)
     &&
00586                 check_bem_cd )
00587             (bvptr[id-1])->set(seq_id_i,true);
00588     }
00589     constr_vect_singleton.push_back(c_v);
00590     pos_lists_singleton.push_back(tmp_pos_l);
00591
00592     /****************
00593     print_singleton_data();
00594     /*****************/
00595     delete ris;
00596 }
00597
00598 Constraints* STSMiner::init_c_v(std::vector<std::string> AttrList, bool* check_bem_cd) {
00599     Constraints* c = new Constraints();
00600     c->bem.push_back(initbemvect_context_dep(AttrList));
00601     *check_bem_cd = *check_bem_cd || c->bem[c->bem.size()-1].count(true)==bemCD_constraints;
00602     if(active_count)
00603         updateMidCounters_singleton(c);
00604     return c;
00605 }
00606
00610 void STSMiner::updateMidCounters_singleton(Constraints* c) {
00611     std::vector<std::pair<int, int> > mid_c_tmp;
00612     for(int j=1;j<bem_vect_length-1;++j) {
00613         int tmp = c->bem[c->bem.size()-1].test(j) ? 1 : 0;
00614         mid_c_tmp.push_back(std::pair<int,int>(tmp,0));
00615     }
00616     c->mid_counters.push_back(mid_c_tmp);
00617 }
00618
00619 BitString STSMiner::initbemvect_context_dep(std::vector<std::string> attrVal) {
00620
00621     BitString output(bem_vect_length);
00622
00623     if(pm.seq_bem_vect.size()>0){
00624         std::string attr_curr_val="";
00625         int mid_list_counter=0;
00626
00627         for(int j=0;j<pm.seq_bem_vect.size();++j) {
00628
00629             attr_curr_val="";
00630             bool ok=true;
00631             Bem_cond* it=pm.seq_bem_vect[j];
00632             bool context_dep = false;
00633             bool context_free = false;
00634
00635             do {
00636                 if(!containedIn(it->attr,pm.ba)) {
00637                     context_dep = true;
00638                     attr_curr_val = find_val(it->attr,attrVal);
00639                     ok = ok && checkConstraint(it->op,attr_curr_val,it->val);
00640                 }
00641                 else
00642                     context_free = true;
00643                 it = it->and_c;
00644             } while(it);
00645
00646             /*x vale 0 se i vincoli erano sul BEGIN,bem_vect_length-1 se erano sull'end,
00647             tra 1 e bem_vect_length-2 se erano su un mid */
00648             int x;
00649             if(pm.seq_bem_vect[j]->type.compare("MID")==0)
```

```
00650                 x=++mid_list_counter;
00651             else if (pm.seq_bem_vect[j]->type.compare("BEGIN")==0)
00652                 x = 0;
00653             else x = bem_vect_length-1;
00654 
00655             if(!context_dep && context_free)
00656                 output.set(x,false);
00657             else
00658                 output.set(x,ok);
00659             delete it;
00660         }
00661     }
00662     return output;
00663 }
00664 
00665 void STSMiner::initbemvect(int id,std::vector<std::string> attrVal) {
00666 
00667     if(pm.seq_bem_vect.size()>0){
00668         std::string attr_curr_val="";
00669         int mid_list_counter=0;
00670 
00671         for(int j=0;j<pm.seq_bem_vect.size();++j) {
00672 
00673             attr_curr_val="";
00674             bool ok=true;
00675             Bem_cond* it=pm.seq_bem_vect[j];
00676             bool context_dep = false;
00677             bool context_free = false;
00678 
00679             do {
00680                 if(containedIn(it->attr,pm.ba)) {
00681                     context_free = true;
00682                     attr_curr_val = find_val(it->attr,attrVal);
00683                     ok = ok && checkConstraint(it->op,attr_curr_val,it->val);
00684                 }
00685                 else
00686                     context_dep = true;
00687                 it = it->and_c;
00688             } while(it);
00689 
00690             /*vale 0 se i vincoli erano sullo BEGIN,bem_vect_length-1 se erano sull'end,
00691             tra 1 e bem_vect_length-2 se erano su un mid */
00692             int x;
00693             if(pm.seq_bem_vect[j]->type.compare("MID")==0)
00694                 x=++mid_list_counter;
00695             else if (pm.seq_bem_vect[j]->type.compare("BEGIN")==0)
00696                 x = 0;
00697             else x = bem_vect_length-1;
00698 
00699             if(context_dep && !context_free)
00700                 (bem_vect[id-1])->set(x,false);
00701             else {
00702                 (bem_vect[id-1])->set(x,ok);
00703                 if(active_count && x!=0 && x!=bem_vect_length-1 && ok) {//siamo in un caso mid
00704                     mid_count[id-1][mid_list_counter-1].first++;
00705                     mid_count[id-1][mid_list_counter-1].second++;
00706                 }
00707             }
00708         }
00709     }
00710 }
00711 
00712 /*************************************
00713   SELECT _s1.ID AS ID_1, _s2.ID AS ID_2, _s1.trackid, _s2.POS, _s1.POS [,_s2.POS_D]
00714   FROM
00715      (SELECT s_1.id,s1.*
00716       FROM example_Seq1 s_1
00717           JOIN
00718            example_Source s1
00719           ON  s_1.subarea_id=s1.subarea_id
00720       ) AS _s1,
00721       (SELECT s_2.id,s2.*
00722        FROM example_Seq1 s_2
00723           JOIN
00724            example_Source s2
00725           ON  s_2.subarea_id=s2.subarea_id
00726        ) AS _s2
00727     WHERE  _s1.trackid=_s2.trackid AND
00728            _s2.pos_d-_s1.pos_d>=1+0 AND
00729            _s2.pos_d-_s1.pos_d<=1+100
00730     ORDER BY ID_1, ID_2,  _s1.trackid,  _s2.POS, _s1.POS
00731   *************************************/
00732 bool STSMiner::extract2Sequences() {
00733 
00734     std::string sqlQuery = "CREATE TABLE " + pm.tab_result
00735             + "_Seq2 (ID SERIAL PRIMARY KEY,ID_1 integer,ID_2 integer);";
00736     execQuery(sqlQuery);
```

```
00737
00738     sqlQuery = "SELECT _s1.ID AS ID_1, _s2.ID AS ID_2, _s1.GRPID, _s2.POS, _s1.POS ";
00739     if(DISTINCT)
00740         sqlQuery += ",_s2.POS_D ";
00741     sqlQuery += " FROM (SELECT s_1.id,s1.* FROM "
00742                 + pm.tab_result + "_Seq1 s_1 JOIN "+ pm.tab_result + "_Source s1 ON " ;
00743     for(int i = 0; i < pm.ba.size()-1; ++i)
00744         sqlQuery += " s_1." + pm.ba[i] + "=s1." + pm.ba[i] + " AND ";
00745     sqlQuery += " s_1." + pm.ba[pm.ba.size()-1] + "=s1." + pm.ba[pm.ba.size()-1]
00746                 + " ) AS _s1, (SELECT s_2.id,s2.* FROM " + pm.tab_result
00747                 + "_Seq1 s_2 JOIN "+ pm.tab_result + "_Source s2 ON ";
00748     for(int i = 0; i < pm.ba.size()-1; ++i)
00749         sqlQuery += " s_2." + pm.ba[i] + "=s2." + pm.ba[i] + " AND ";
00750     sqlQuery += " s_2." + pm.ba[pm.ba.size()-1] + "=s2." + pm.ba[pm.ba.size()-1] + " ) AS _s2 WHERE "
00751                 + " _s1.GRPID=_s2.GRPID AND ";
00752     if(DISTINCT)
00753         sqlQuery += " _s2.pos_d-_s1.pos_d>=1+" + Converter(pm.sequenceAllowedGaps.getMin()).toString()
00754                 + " AND _s2.pos_d-_s1.pos_d-1<=" +
      Converter(pm.sequenceAllowedGaps.getMax()).toString();
00755     else
00756         sqlQuery += " _s2.pos-_s1.pos-1>=" + Converter(pm.sequenceAllowedGaps.getMin()).toString()
00757                 + " AND _s2.pos-_s1.pos-1<=" + Converter(pm.sequenceAllowedGaps.getMax()).toString();
00758
00759     if(pm.seq_dist_vect.size()>0)
00760         for(int i=0; i< pm.seq_dist_vect.size();++i) {
00761             sqlQuery += " AND " + pm.seq_dist_vect[i]->function + "(";
00762             for(int j=0; j<pm.seq_dist_vect[i]->attr.size();j++)
00763                 sqlQuery += "_s1." + pm.seq_dist_vect[i]->attr[j]+", ";
00764             for(int j=0; j<pm.seq_dist_vect[i]->attr.size()-1;j++)
00765                 sqlQuery += "_s2." + pm.seq_dist_vect[i]->attr[j]+", ";
00766             sqlQuery+= "_s2." + pm.seq_dist_vect[i]->attr[pm.seq_dist_vect[i]->attr.size()-1]
00767                     + ") " + pm.seq_dist_vect[i]->range;
00768         }
00769     sqlQuery += " ORDER BY ID_1, ID_2, _s1.GRPID, _s2.POS, _s1.POS ";
00770
00771     return pruningSeq2(sqlQuery);
00772 }
00773
00774 /*************************************
00775 SELECT s1.ID AS ID_1, s2.ID AS ID_2
00776 FROM example_Seq2 s1, example_Seq2 s2
00777 WHERE s1.ID_2=s2.ID_1
00778 GROUP BY s1.ID,s2.ID
00779 ORDER BY ID_1,ID_2
00780   *************************************/
00781 bool STSMiner::extractKSequences(int k) {
00782
00783         std::string strk_1 = Converter(k-1).toString();
00784
00785         std::string query = "CREATE TABLE " + pm.tab_result + "_Seq" + Converter(k).toString()
00786                 + " (ID SERIAL PRIMARY KEY,ID_1 integer,ID_2 integer)";
00787         execQuery(query);
00788
00789         query = "SELECT s1.ID AS ID_1, s2.ID AS ID_2 FROM " + pm.tab_result + "_Seq" + strk_1 + " s1,
      "
00790                  + pm.tab_result + "_Seq" + strk_1
00791                  + " s2 WHERE s1.ID_2=s2.ID_1 GROUP BY s1.ID,s2.ID ORDER BY ID_1,ID_2 ";
00792
00793         return pruningSeqK(k,query);
00794 }
00795
00796 bool STSMiner::pruningSeq2(std::string sqlQuery){
00797     std::vector<int> pos_d_1;
00798     std::vector<std::vector<int> > prefix_pos_1;
00799     std::vector<int> new_prefix_list;
00800     Constraints* constr;
00801     std::vector<Constraints*> c_v;
00802     std::vector<bool> closed_seq;
00803     std::vector<std::pair<int,int>* > ris_1;
00804     BitString new_bem_string;
00805     std::vector<std::pair<int,int> > new_mid_count_vect;
00806     int size_before = constr_vect.size();
00807     int size_before_results = results.size();
00808     int size_before_bvptr = bvptr.size();
00809     int size_before_pos = pos_lists.size();
00810     int size_before_prefix_pos = prefix_pos_lists.size();
00811     int size_before_bem_vect = 0;
00812     int size_before_mid_count = 0;
00813     if(context_free_BEM) {
00814         size_before_bem_vect = bem_vect.size();
00815         if(active_count)
00816             size_before_mid_count = mid_count.size();
00817     }
00818     int size_before_pos_d = 0;
00819     if(DISTINCT)
00820         size_before_pos_d = pos_d_lists.size();
00821
```

```
00822     int id1_old = -1;
00823     int id2_old = -1;
00824     int pos_old = -1;
00825     int grp_old = -1;
00826     int prefix_pos_old = -1;
00827     int id1,id2;
00828     bool check_bem=true;
00829 
00830     BitString bem_cf(bem_vect_length);
00831     BitString bem_cd(bem_vect_length);
00832 
00833     mrdb::ResultSet* rs = execQr(sqlQuery);
00834 
00835     for(int i = 0; i < bvptr.size(); ++i)
00836        closed_seq.push_back(true);
00837 
00838     bool first = true ;
00839     bool check_bem_cd = false;
00840     BitString tmp0(max_trackid-min_trackid+1);//tmp0 lo setto solo se i vincoli sono rispettati
00841     int frequence=0;//mi conta il supporto visto che devo salvare anche le seq che non rispettano i
      vicnoli ma hanno enough support
00842     if(context_dep_BEM)
00843         constr = new Constraints();
00844 
00845     while(rs->next()) {
00846 
00847         id1 = rs->getInt(1);
00848         id2 = rs->getInt(2);
00849         int grp=rs->getInt(3)-min_trackid;
00850         int pos = rs->getInt(4);
00851         int prefix_pos = rs->getInt(5);
00852         int pos_d;
00853 
00854         if(DISTINCT)
00855             pos_d = rs->getInt(6);
00856 
00857         bool pair_change=false;
00858 
00859         if(id1!=id1_old || id2!=id2_old) {
00860             bool save_pattern = !first && (frequence>=pm.sup*grpId_count);
00861             pair_change=true;//this was into following if... but the pair has changed anyway...
00862 
00863             if(save_pattern) {
00864                 if(context_dep_BEM)
00865                     c_v.push_back(constr);
00866                 if(context_dep_dist)
00867                     prefix_pos_l.push_back(new_prefix_list);
00868                 saveSequenceData(2,id1_old,id2_old,&check_bem,tmp0,ris_l,pos_d_l,&new_bem_string,
00869                                  &closed_seq,prefix_pos_l,new_mid_count_vect,c_v);
00870             }
00871             frequence=0;
00872             new_mid_count_vect.clear();
00873             ris_l.clear();
00874             new_prefix_list.clear();
00875             prefix_pos_l.clear();
00876             if(context_dep_BEM)
00877                 constr= new Constraints();
00878             c_v.clear();
00879             tmp0.clear();
00880             pos_d_l.clear();
00881             check_bem=true;
00882             check_bem_cd=false;
00883         }
00884         id1_old = id1;
00885         id2_old = id2;
00886 
00887         if(context_free_BEM && ( first || pair_change))
00888             init_bem_bit_vector(2,id1,id2, &check_bem, &bem_cf,&new_bem_string,&new_mid_count_vect);
00889 
00890         if(pair_change || grp != grp_old || pos != pos_old) {
00891             frequence++;
00892             ris_l.push_back(new std::pair<int,int>(grp+min_trackid,pos));
00893             if(DISTINCT)
00894                 pos_d_l.push_back(pos_d);
00895             if(!pair_change && !first) {
00896                 if(context_dep_dist) {
00897                     prefix_pos_l.push_back(new_prefix_list);
00898                     new_prefix_list.clear();
00899                 }
00900                 if(context_dep_BEM) {
00901                     c_v.push_back(constr);
00902                     constr= new Constraints();
00903                 }
00904             }
00905             if(context_dep_dist)
00906                 new_prefix_list.push_back(prefix_pos);
00907             if(context_dep_BEM)
```

```
00908                 updateBemData(constr,id1,id2,prefix_pos,pos,grp,&check_bem_cd,&bem_cd);
00909             if(check_bem && (check_bem_cd || !context_dep_BEM))
00910                 tmp0.set(grp,true);
00911         }
00912         else if(prefix_pos_old != prefix_pos) {
00913             if(context_dep_dist)
00914                 new_prefix_list.push_back(prefix_pos);
00915             if(context_dep_BEM)
00916                 updateBemData(constr,id1,id2,prefix_pos,pos,grp,&check_bem_cd,&bem_cd);
00917         }
00918         grp_old = grp;
00919         pos_old = pos;
00920         first = false ;
00921     }
00922
00923     if(frequence>=pm.sup*grpId_count) {
00924         if(context_dep_BEM)
00925             c_v.push_back(constr);
00926         if(context_dep_dist)
00927             prefix_pos_l.push_back(new_prefix_list);
00928         saveSequenceData(2,id1_old,id2_old,&check_bem,tmp0,ris_l,pos_d_l,
00929                          &new_bem_string,&closed_seq,prefix_pos_l,new_mid_count_vect,c_v);
00930     }
00931
00932     if(min_seq_length<2 && pm.seq_dist_vect.size()==0 && max_MIN_COUNT==1)
00933         for(int i = 0; i < closed_seq.size() ; ++i)
00934             if(closed_seq[i] && bvptr[i]->count(true)>=pm.sup*grpId_count)
00935                 saveSingleton(i+1);
00936
00937     delete rs;
00938
00939     resetStructs(2,size_before_results,
00940                  size_before_bvptr,
00941                  size_before_pos,
00942                  size_before_prefix_pos,
00943                  size_before_bem_vect,
00944                  size_before_mid_count,
00945                  size_before_pos_d,
00946                  size_before);
00947
00948     /*******************
00949     print_structs_values();
00950     *******************/
00951
00952     bool go_on= bem_cf.count(true)==bemCF_constraints && bem_cd.count(true)==bemCD_constraints;
00953
00954     if(bvptr.size()>0 && go_on)
00955        return false;
00956
00957     return true;
00958 }
00959
00960 void STSMiner::updateBemData(Constraints* constr,int id1,int id2,int prefix_pos,
00961                              int pos,int grp,bool* check_bem_cd,BitString* bem_cd) {
00962
00963     int i = findIndex(id1,prefix_pos,grp+min_trackid);
00964     int j = findIndex(id2,pos,grp+min_trackid);
00965     constr->bem.push_back(inferBem(id1,id2,i,j));
00966     bool check_counters = true;
00967     if(active_count) {
00968         updateMidCounters(constr,id1,id2,i,j);
00969         check_counters = checkMidCounters(constr->mid_counters[constr->mid_counters.size()-1]);
00970      }
00971     *check_bem_cd = *check_bem_cd || (constr->bem[constr->bem.size()-1].count(true)==bemCD_constraints
       && check_counters);
00972     check_go_on(constr,bem_cd);
00973 }
00974
00979 void STSMiner::updateMidCounters(Constraints* constr,int id1,int id2, int i, int j) {
00980     std::vector<std::pair<int,int> > tmp;
00981     for(int h =1; h< bem_vect_length-1; ++h) {
00982         int mc_1 = constr_vect_singleton[id1-1][i]->mid_counters[0][h-1].first;
00983         int mc_2 = constr_vect_singleton[id2-1][j]->mid_counters[0][h-1].first;
00984         tmp.push_back(std::pair<int,int>(mc_1,mc_2));
00985     }
00986     constr->mid_counters.push_back(tmp);
00987 }
00988
00989 bool STSMiner::checkMidCounters(std::vector<std::pair<int,int> >mc_v) {
00990     int mc_ok=true;
00991     for(int i=0;i<mc_v.size();++i) {
00992         mc_ok = mc_ok && ((mc_v[i].first+mc_v[i].second)<=min_max_mid_count[i].getMax())
00993                 && ((mc_v[i].first+mc_v[i].second)>=min_max_mid_count[i].getMin());
00994     }
00995     return mc_ok;
00996 }
00997
```

```
01001 void STSMiner::check_go_on(Constraints* c,BitString* ris) {
01002     BitString bs_cd = c->bem[c->bem.size()-1];
01003     if(!active_count)
01004         ris->operator|=(bs_cd);
01005     else {
01006         ris->set(0,(ris->test(0) || bs_cd.test(0)));
01007         for(int i=1; i<bs_cd.size()-1; ++i) {
01008             int mc = c->mid_counters[c->mid_counters.size()-1][i-1].first
01009                     + c->mid_counters[c->mid_counters.size()-1][i-1].second;
01010             bool mc_ok = mc<=min_max_mid_count[i-1].getMax();
01011             ris->set(i,(ris->test(i) || (bs_cd.test(i) && mc_ok)));
01012         }
01013         ris->set(bs_cd.size()-1,(ris->test(bs_cd.size()-1) || bs_cd.test(bs_cd.size()-1)));
01014     }
01015 }
01016
01017 BitString STSMiner::inferBem(int id1,int id2, int i, int j) {
01018
01019     BitString output(bem_vect_length);
01020     BitString t(constr_vect_singleton[id1-1][i]->bem[0]);
01021     BitString t2(constr_vect_singleton[id2-1][j]->bem[0]);
01022     output.set(0,t.test(0));
01023     output.set(bem_vect_length-1,t2.test(bem_vect_length-1));
01024     for(int k=1;k<bem_vect_length-1;++k)
01025         output.set(k,(t.test(k) || t2.test(k)));
01026     return output;
01027 }
01028
01029 int STSMiner::findIndex(int id, int pos, int grpId) {
01030     for(int j = 0; j < pos_lists_singleton[id-1].size(); ++j)
01031         if(pos_lists_singleton[id-1][j]->first==grpId && pos_lists_singleton[id-1][j]->second==pos)
01032             return j;
01033     return -1;
01034 }
01035
01036 bool STSMiner::pruningSeqK(int k,std::string sqlQuery) {
01037
01038     std::vector<Constraints*> c_v;
01039     std::vector<std::vector<int> > new_prefix_pos_list;
01040     std::vector<bool> closed_seq;
01041     std::vector<std::pair<int,int>* > ris_l;
01042     BitString new_bem_string;
01043     std::vector<std::pair<int,int> > new_mid_count_vect;
01044     int size_before = constr_vect.size()>0 ? constr_vect.size() : 0;
01045     int size_before_results = results.size();
01046     int size_before_bvptr = bvptr.size();
01047     int size_before_pos = pos_lists.size();
01048     int size_before_prefix_pos = context_dep_dist ? prefix_pos_lists.size() : 0;
01049     int size_before_bem_vect = 0;
01050     int size_before_mid_count = 0;
01051     if(context_free_BEM) {
01052         size_before_bem_vect = bem_vect.size();
01053         if(active_count)
01054             size_before_mid_count = mid_count.size();
01055     }
01056     int size_before_pos_d = DISTINCT ? pos_d_lists.size() : 0;
01057     int id1,id2;
01058     bool check_bem = false;
01059     BitString bem_cf(bem_vect_length);
01060     BitString bem_cd(bem_vect_length);
01061
01062     mrdb::ResultSet* rs = execQr(sqlQuery);
01063
01064     reset_seqi_cache(k);
01065
01066     for(int i = 0; i < results.size(); ++i)
01067             closed_seq.push_back(true);
01068
01069     BitString tmp0(max_trackid-min_trackid+1);
01070     std::vector<int> new_pos_d_l;
01071
01072     while(rs->next()) {
01073
01074         id1 = rs->getInt(1);
01075         id2 = rs->getInt(2);
01076
01077         if(context_free_BEM)
01078             init_bem_bit_vector(k,id1,id2, &check_bem, &bem_cf,&new_bem_string,&new_mid_count_vect);
01079         check_bem = !context_free_BEM || check_bem;
01080
01081         if(check_pos_lists_with_gap(id1,id2,&ris_l, &tmp0, &new_prefix_pos_list, &new_pos_d_l,
      &check_bem,&c_v, &bem_cd) ) {
01082             saveSequenceData(k,id1,id2,check_bem,tmp0,ris_l,new_pos_d_l,&new_bem_string,
01083                              &closed_seq,new_prefix_pos_list,new_mid_count_vect,c_v);
01084         }
01085         new_prefix_pos_list.clear();
01086         new_pos_d_l.clear();
```

```
01087         new_mid_count_vect.clear();
01088         tmp0.clear();
01089         ris_l.clear();
01090         c_v.clear();
01091     }
01092 
01093     if(min_seq_length<k)
01094      for(int i = 0; i < closed_seq.size() ; ++i) {
01095          if(closed_seq[i] && bvptr[i]->count(true) >= pm.sup*grpId_count) {
01096              double sup = bvptr[i]->count(true)/(double)grpId_count;
01097              saveSequence(i+1,results[i],sup);
01098          }
01099      }
01100      delete rs;
01101 
01102      resetStructs(k,
01103                   size_before_results,
01104                   size_before_bvptr,
01105                   size_before_pos,
01106                   size_before_prefix_pos,
01107                   size_before_bem_vect,
01108                   size_before_mid_count,
01109                   size_before_pos_d,
01110                   size_before);
01111 
01112     /********************
01113     print_structs_values();
01114     ********************/
01115     bool go_on= (bem_cf.count(true)==bemCF_constraints) && (bem_cd.count(true)==bemCD_constraints);
01116 
01117     if(bvptr.size()>0 && go_on)
01118         return false;
01119     return true;
01120 }
01121 
01122 void STSMiner::resetStructs(int k,
01123                             int size_before_results,
01124                             int size_before_bvptr,
01125                             int size_before_pos,
01126                             int size_before_prefix_pos,
01127                             int size_before_bem_vect,
01128                             int size_before_mid_count,
01129                             int size_before_pos_d,
01130                             int size_before) {
01131     if(k>0)
01132         finalize_seqi(k);
01133     results.erase(results.begin(),results.begin()+size_before_results);
01134 
01135     deleter(bvptr.begin(),bvptr.begin()+size_before_bvptr); //delete objects
01136     bvptr.erase(bvptr.begin(),bvptr.begin()+size_before_bvptr);
01137 
01138     for(int i = 0; i < size_before_pos; i++) {
01139         deleter(pos_lists[i].begin(),pos_lists[i].end()); //delete objects
01140     }
01141     pos_lists.erase(pos_lists.begin(),pos_lists.begin()+size_before_pos);
01142 
01143     if(context_dep_dist)
01144 
      prefix_pos_lists.erase(prefix_pos_lists.begin(),prefix_pos_lists.begin()+size_before_prefix_pos);
01145 
01146     if(context_free_BEM) {
01147         deleter(bem_vect.begin(),bem_vect.begin()+size_before_bem_vect); //delete objects
01148         bem_vect.erase(bem_vect.begin(),bem_vect.begin()+size_before_bem_vect);
01149         if(active_count)
01150             mid_count.erase(mid_count.begin(),mid_count.begin()+size_before_mid_count);
01151     }
01152 
01153     if(DISTINCT)
01154         pos_d_lists.erase(pos_d_lists.begin(),pos_d_lists.begin()+size_before_pos_d);
01155 
01156     if(context_dep_BEM) {
01157         for(int i = 0; i < size_before; i++) {
01158             deleter(constr_vect[i].begin(),constr_vect[i].end()); //delete objects
01159         }
01160         constr_vect.erase (constr_vect.begin(),constr_vect.begin()+size_before);
01161     }
01162     if(k>2)
01163         dropTable(pm.tab_result+"_Seq"+Converter(k-1).toString());
01164     else
01165         dropTable(pm.tab_result+"_Source");
01166 }
01167 
01168 void STSMiner::saveSequenceData(int k,int& id1, int& id2, bool check_bem,
01169                       BitString tmp0,
01170                       std::vector<std::pair<int,int>* > ris_l,
01171                       std::vector<int> pos_d_l,
01172                       BitString* new_bem_string,
```

```
01173                       std::vector<bool>* closed_seq,
01174                       std::vector<std::vector<int> > prefix_pos_l,
01175                       std::vector<std::pair<int,int> > new_count,
01176                       std::vector<Constraints*> c_v) {
01177
01178     BitString tmp(tmp0);
01179     if(check_bem || (!check_bem && !checkBem(id1,k))) {
01180         BitString tmp_x(*bvptr[id1-1]);
01181         if(tmp_x.count(true)==tmp.count(true))
01182             (*closed_seq)[id1-1] = false;
01183     }
01184     if(check_bem || (!check_bem && !checkBem(id2,k))) {
01185         BitString tmp_y(*bvptr[id2-1]);
01186         if(tmp_y.count(true)==tmp.count(true))
01187             (*closed_seq)[id2-1] = false;
01188     }
01189
01190     if(k==2) {
01191         std::vector<std::string> t;
01192         t.push_back(Converter(id1).toString());
01193         t.push_back(Converter(id2).toString());
01194         results.push_back(t);
01195     }
01196     else
01197         results.push_back(copy_and_merge(results[id1-1],results[id2-1]));
01198
01199     bvptr.push_back(new BitString(tmp0));
01200     pos_lists.push_back(ris_l);
01201
01202     if(context_dep_dist)
01203         prefix_pos_lists.push_back(prefix_pos_l);
01204     if(DISTINCT)
01205         pos_d_lists.push_back(pos_d_l);
01206     if(context_free_BEM) {
01207         bem_vect.push_back(new BitString(*new_bem_string));
01208         if(active_count)
01209             mid_count.push_back(new_count);
01210     }
01211     if(context_dep_BEM)
01212         constr_vect.push_back(c_v);
01213
01214     seqicache « id1 « "\t"« id2 « std::endl;
01215 }
01216
01217
01218 bool STSMiner::checkBem(int id, int k) {
01219
01220     bool check_mid_count= true;
01221     bool cf = true;
01222     if(context_free_BEM) {
01223         if(active_count) {
01224             for(int i=1;i<bem_vect_length-1;++i) {
01225                 int mc = mid_count[id-1][i-1].first + mid_count[id-1][i-1].second;
01226                 check_mid_count = check_mid_count && mc>=min_max_mid_count[i-1].getMin() &&
      mc<=min_max_mid_count[i-1].getMax();
01227             }
01228         }
01229         cf = ((bem_vect[id-1]->count(true) == bemCF_constraints) && check_mid_count);
01230     }
01231
01232     bool cd = context_dep_BEM ? false : true;
01233     if(context_dep_BEM) {//aggiungi check sul count!!!!!
01234         if(k>2){
01235             int i=0;
01236             do {
01237                 for(int j=0; j<constr_vect[id-1][i]->bem.size(); ++j) {
01238                     bool mc_ok = true;
01239                     if(active_count)
01240                         for(int h=1; h<constr_vect[id-1][i]->mid_counters.size(); ++h) {
01241                             int mc = constr_vect[id-1][i]->mid_counters[j][h-1].first
01242                                    + constr_vect[id-1][i]->mid_counters[j][h-1].second;
01243                             mc_ok = mc_ok && mc<=min_max_mid_count[h-1].getMax();
01244                         }
01245                     cd = cd || ((constr_vect[id-1][i]->bem[j]).count(true)==bemCD_constraints &&
      mc_ok);
01246                 }
01247                 ++i;
01248             } while(!cd && i<constr_vect[id-1].size());
01249         }
01250         else{
01251             int i=0;
01252             do {
01253                 for(int j=0; j<constr_vect_singleton[id-1][i]->bem.size(); ++j) {
01254                     bool mc_ok = true;
01255                     if(active_count)
01256                         for(int h=1; h<constr_vect_singleton[id-1][i]->mid_counters.size(); ++h) {
01257                             int mc = constr_vect_singleton[id-1][i]->mid_counters[j][h-1].first
```

```
01258                                   + constr_vect_singleton[id-1][i]->mid_counters[j][h-1].second;
01259                               mc_ok = mc_ok && mc<=min_max_mid_count[i-1].getMax();
01260                           }
01261                       cd = cd ||
     ((constr_vect_singleton[id-1][i]->bem[j]).count(true)==bemCD_constraints && mc_ok);
01262                   }
01263                   ++i;
01264               } while(!cd && i<constr_vect_singleton[id-1].size());
01265           }
01266       }
01267
01268       return cf && cd;
01269   }
01270
01271
01272   void STSMiner::init_bem_bit_vector(int k,int id1,int id2,bool* check_bem,
01273                           BitString* bem_cf,
01274                           BitString* new_bem_vect,
01275                           std::vector<std::pair<int,int> >* new_count) {
01276
01277       BitString t(*bem_vect[id1-1]);
01278       BitString t2(*bem_vect[id2-1]);
01279
01280       new_bem_vect->set(0,t.test(0));
01281       new_bem_vect->set(bem_vect_length-1,t2.test(bem_vect_length-1));
01282       for(int i=1;i<bem_vect_length-1;++i){
01283           new_bem_vect->set(i,(t.test(i) || t2.test(i)));//se c'è almeno un mid-i...
01284           if(!active_count)
01285               bem_cf->set(i,bem_cf->test(i) || (t.test(i) || t2.test(i)));
01286       }
01287       bem_cf->set(0,bem_cf->test(0) || t.test(0));
01288       bem_cf->set(bem_vect_length-1,bem_cf->test(bem_vect_length-1) || t2.test(bem_vect_length-1));
01289
01290       bool check_mid_count=true;
01291       if(active_count) {
01292           int mc;
01293
01294           for(int i=1;i<bem_vect_length-1;++i) {
01295               std::pair<int,int> temp(0,0);
01296               //CALCOLIAMO IL COUNT DEL MID-i
01297               if(k==2) {
01298                   if(t.test(i))
01299                       temp.first = 1;
01300                   if(t2.test(i))
01301                       temp.second = 1;
01302               }
01303               else {
01304                   temp.first = mid_count[id1-1][i-1].first + mid_count[id1-1][i-1].second;
01305                   temp.second = mid_count[id2-1][i-1].second;
01306               }
01307               new_count->push_back(temp);
01308               mc=temp.first+temp.second;
01309               check_mid_count = check_mid_count && mc>=min_max_mid_count[i-1].getMin() &&
     mc<=min_max_mid_count[i-1].getMax();
01310               //se tutti gli el.i hanno già passato il count non si potranno formare altre seq valide!
01311               bem_cf->set(i, bem_cf->test(i) || ((t.test(i) || t2.test(i)) &&
     mc<=min_max_mid_count[i-1].getMax()));
01312           }
01313       }
01314       //non solo deve esserci almeno un mid ma deve avere almeno un elemento con count <= al max count
01315       //else significa che ha elemnti mid ma che ne ha troppi e non genererà mai una seq valida!
01316     *check_bem=(new_bem_vect->count(true) == bemCF_constraints) && check_mid_count;
01317   }
01318
01319
01320   bool STSMiner::check_pos_lists_with_gap(int id1, int id2,std::vector<std::pair<int,int>* >* ris_l,
01321                                           BitString* bv_constr,
01322                                           std::vector<std::vector<int> >* new_prefix_pos_list,
01323                                           std::vector<int>* new_pos_d_l,
01324                                           bool* BEM_context_free,
01325                                           std::vector<Constraints*>* c_v,
01326                                           BitString* bem_cd) {
01327   /*
01328           std::cout« "Checking pos lists..." «std::endl;
01329           std::cout«"POS_LIST of ID "+Converter(id1).toString()+":"«std::endl;
01330           std::cout«pos_list_string(pos_lists[id1-1]);
01331           std::cout«std::endl;
01332           std::cout«"POS_LIST of ID "+Converter(id2).toString()+":"«std::endl;
01333           std::cout«pos_list_string(pos_lists[id2-1]);
01334           std::cout«std::endl;
01335   */
01336       std::vector<std::pair<int,int>* >::iterator it1 = pos_lists[id1-1].begin();
01337       std::vector<std::pair<int,int>* >::iterator it2 = pos_lists[id2-1].begin();
01338       std::vector<std::pair<int,int>* >::iterator end1 = pos_lists[id1-1].end();
01339       std::vector<std::pair<int,int>* >::iterator end2 = pos_lists[id2-1].end();
01340       std::vector<int >::iterator it1d, it2d;
01341       if(DISTINCT) {
```

```
01342         it1d = pos_d_lists[id1-1].begin();
01343         it2d = pos_d_lists[id2-1].begin();
01344     }
01345     int grp1, grp2;
01346     int min_gap = pm.sequenceAllowedGaps.getMin();
01347     int max_gap = pm.sequenceAllowedGaps.getMax();
01348     int it2_c = 0;
01349     bool check_bem=false;
01350     BitString bv(max_trackid-min_trackid+1);
01351     std::vector<int> prefix_l;
01352 
01353     while(it2 != end2) {
01354         grp2 = (*it2)->first;
01355 
01356         while(it1 != end1 && (*it1)->first < grp2) {
01357             it1++;
01358             it1d++;
01359         }
01360         if(it1 == end1)
01361             break;
01362 
01363         grp1 = (*it1)->first;
01364 
01365         while(it2 != end2 && grp1 > (*it2)->first ) {
01366             it2++;
01367             it2_c++;
01368             it2d++;
01369         }
01370         if(it2 == end2)
01371             break;
01372 
01373         grp2 = (*it2)->first;
01374 
01375         if(grp1 == grp2) {
01376 
01377             std::vector<std::pair<int,int>* >::iterator it1_start_grp = it1;
01378             std::vector<int>::iterator it1d_start_grp;
01379             if(DISTINCT)
01380                 it1d_start_grp = it1d;
01381             int pos2 = DISTINCT ? *it2d : (*it2)->second;
01382             int pos1 = DISTINCT ? *it1d : (*it1)->second;
01383 
01384             if(pos1<pos2) {
01385                 while(pos2-max_gap-1 > pos1) {
01386                     if(it1+1 != end1 && (*(it1+1))->first == grp1) {
01387                         ++it1;
01388                         ++it1d;
01389                         pos1 = DISTINCT ? *it1d : (*it1)->second;
01390                     }
01391                     else break;
01392                 }
01393                 //devi controllare perchè è uscito.
01394                 if(it1 != end1 &&
01395                         (*it1)->first == grp1 &&
01396                         pos1 >= pos2-max_gap-1 &&
01397                         pos1 <= pos2-min_gap-1) {
01398                     //se è uscito perchè ha trovato la prima pos che rispetta il gap...
01399                     bool insert = true;
01400                     bool BEM_context_dep= context_dep_BEM ? false : true;
01401                     if(context_dep_dist || context_dep_BEM) {
01402                         //se non ci sono distanze contestuali basta che ci sia un singolo it1 che va
    bene rispetto al gap
01403                         //else ci serve controllare che it1 sia uguale ad uno dei prefix di it2 ed
    aggiungere id1 alla
01404                         //lista dei prefissi
01405                         do{
01406                             bool valid_prefix= false;
01407                             if(context_dep_dist) {
01408                                 for(int i = 0; i<prefix_pos_lists[id2-1][it2_c].size(); ++i)
01409                                     if(pos1 == prefix_pos_lists[id2-1][it2_c][i]) {
01410                                         prefix_l.push_back(pos1);
01411                                         valid_prefix = true;
01412                                     }
01413                                 insert = prefix_l.size()>0;
01414                             }
01415                             if(context_dep_BEM && (!context_dep_dist || (context_dep_dist &&
    valid_prefix))) {
01416                                 Constraints* temp= new Constraints();
01417                                 int it1_c = it1 - pos_lists[id1-1].begin();
01418                                 BEM_context_dep = BEM_context_dep || init_bemCD(id1, it1_c, id2,
01419                                                         (*it2)->second, grp2, temp, bem_cd);
01420                                 (*c_v).push_back(temp);
01421                             }
01422                             it1++;
01423                             it1d++;
01424                             if(it1 != end1)
01425                                 pos1 = DISTINCT ? *it1d : (*it1)->second;
```

```
01426
01427                         } while(it1 != end1 && (*it1)->first == grp1 && pos2-min_gap-1 <= pos1);
01428                     }
01429                     if(insert) {
01430                         bv.set(grp2-min_trackid,true);
01431                         (*ris_l).push_back(new std::pair<int,int>(grp2,(*it2)->second));
01432                         if(DISTINCT)
01433                             new_pos_d_l->push_back(*it2d);
01434                         if(context_dep_dist) {
01435                             new_prefix_pos_list->push_back(prefix_l);
01436                             prefix_l.clear();
01437                         }
01438                         if(*BEM_context_free && BEM_context_dep) {
01439                             bv_constr->set(grp2-min_trackid,true);
01440                             check_bem= true;
01441                         }
01442                     }
01443                 }
01444             }
01445             it1 = it1_start_grp;
01446             it1d = it1d_start_grp;
01447             it2++;
01448             it2d++;
01449             it2_c++;
01450         }
01451     }
01452
01453 /*
01454     std::cout<<"POS_LIST result: " + Converter(id2).toString() + ":"<<std::endl;
01455     std::cout<<pos_list_string(*ris_l);
01456     std::cout<<std::endl;
01457 */
01458
01459     (*BEM_context_free) = check_bem;
01460
01461     if( bv.count(true)>= (pm.sup*grpId_count))
01462         return true;
01463     return false;
01464 }
01465
01466 bool STSMiner::init_bemCD(int id1, int index_id1, int id2, int pos2, int grpId, Constraints*
       output,BitString* bem_cd) {
01467     //ci dice se almeno una delle bs inferite è tutta settata
01468     bool at_least_one = false;
01469     //1.PRENDI LE BS[] ASSOCIATE AD ID1, IN TRACK_i CON POS1
01470     std::vector<BitString> bs1= constr_vect[id1-1][index_id1]->bem;
01471
01472     int id2_singleton= (int)Converter(results[id2-1][results[id2-1].size()-1]).toLong();
01473     //2.PRENDI LA BS DI ID2
01474     int j= findIndex(id2_singleton,pos2,grpId);
01475     BitString bs2= constr_vect_singleton[id2_singleton-1][j]->bem[0];
01476     //3.CICLA SU BS1[]
01477     for(int i = 0; i<bs1.size(); ++i) {
01478         BitString temp = inferBemCD(bs1[i],bs2);
01479         output->bem.push_back(temp);
01480         bool mc_ok = true;
01481         if(active_count) {
01482             //per ogni bit mid di temp...
01483             std::vector<std::pair<int,int> >new_mc_v ;
01484             for(int h=1; h<bem_vect_length-1; ++h){
01485                 int mc2 = constr_vect_singleton[id2_singleton-1][j]->mid_counters[0][h-1].first;
01486                 int mc1 = constr_vect[id1-1][index_id1]->mid_counters[i][h-1].first
01487                         + constr_vect[id1-1][index_id1]->mid_counters[i][h-1].second;
01488                 new_mc_v.push_back(std::pair<int,int>(mc1,mc2));
01489                 mc_ok = mc_ok && ((mc1+mc2)<=min_max_mid_count[h-1].getMax()) &&
       (mc1+mc2>=min_max_mid_count[h-1].getMin());
01490             }
01491             output->mid_counters.push_back(new_mc_v);
01492         }
01493         at_least_one= at_least_one || (temp.count(true)==bemCD_constraints && mc_ok);
01494         check_go_on(output,bem_cd);
01495     }
01496     return at_least_one;
01497 }
01498
01499 BitString STSMiner::inferBemCD(BitString bs1,BitString bs2) {
01500     BitString output(bem_vect_length);
01501     output.set(0,bs1.test(0));
01502     output.set(bem_vect_length-1,bs2.test(bem_vect_length-1));
01503     for(int k=1;k<bem_vect_length-1;++k)
01504         output.set(k,(bs1.test(k) || bs2.test(k)));
01505
01506     return output;
01507 }
01508
01509 void STSMiner::saveSequences(int k){
01510
```

```
01511     if(k==1 && min_seq_length<2 && pm.seq_dist_vect.size()==0 && max_MIN_COUNT==1)
01512         for(int i = 0; i < bvptr.size() ; ++i)
01513             if(bvptr[i]->count(true)>=pm.sup*grpId_count)
01514                 saveSingleton(i+1);
01515 
01516     if(k>1 && min_seq_length<=k)
01517      for(int i = 0; i < bvptr.size() ; ++i)
01518          if(bvptr[i]->count(true) >= pm.sup*grpId_count) {
01519              double sup = bvptr[i]->count(true)/(double)grpId_count;
01520              saveSequence(i+1,results[i],sup);
01521          }
01522 }
01523 
01524 void STSMiner::delete_all(int k) {
01525     int size_before = constr_vect.size()>0 ? constr_vect.size() : 0;
01526     int size_before_results = results.size();
01527     int size_before_bvptr = bvptr.size();
01528     int size_before_pos = pos_lists.size();
01529     int size_before_prefix_pos = context_dep_dist ? prefix_pos_lists.size() : 0;
01530     int size_before_bem_vect = 0;
01531     int size_before_mid_count = 0;
01532     if(context_free_BEM) {
01533         size_before_bem_vect = bem_vect.size();
01534         if(active_count)
01535             size_before_mid_count = mid_count.size();
01536     }
01537     int size_before_pos_d = DISTINCT ? pos_d_lists.size() : 0;
01538 
01539     resetStructs(k,
01540                  size_before_results,
01541                  size_before_bvptr,
01542                  size_before_pos,
01543                  size_before_prefix_pos,
01544                  size_before_bem_vect,
01545                  size_before_mid_count,
01546                  size_before_pos_d,
01547                  size_before);
01548     //DELETE SINGLETON STRUCTS!
01549     if(context_dep_BEM) {
01550         for(int i = 0; i < pos_lists_singleton.size(); i++)
01551             deleter(pos_lists_singleton[i].begin(),pos_lists_singleton[i].end()); //delete objects
01552 
      pos_lists_singleton.erase(pos_lists_singleton.begin(),pos_lists_singleton.begin()+size_before_pos);
01553 
01554         for(int i = 0; i < constr_vect_singleton.size(); i++)
01555             deleter(constr_vect_singleton[i].begin(),constr_vect_singleton[i].end()); //delete objects
01556         constr_vect_singleton.erase
      (constr_vect_singleton.begin(),constr_vect_singleton.begin()+size_before);
01557     }
01558 }
01559 
01560 void STSMiner::dropResultsTables(int k) {
01561 
01562     dropTable(pm.tab_result +"_Seq");
01563     dropTable(pm.tab_result +"_Seq_support");
01564     dropTable(pm.tab_result +"_Seq1");
01565     dropTable(pm.tab_result +"_matcher");
01566 
01567     dropTable(pm.tab_result +"_Seq"+ Converter(k).toString());
01568     dropTable(pm.tab_result +"_Seq"+ Converter(k-1).toString());
01569 }
01570 
01571 
01572 void STSMiner::mineRules () {
01573 
01574     int k = 3;
01575 
01576     try {
01577         MRLogPop();
01578         std::cout«std::endl;
01579         MRLogPush("Starting STSMiner mining algorithm...");
01580         dropTable(pm.tab_result+"_matcher");
01581 
01582         std::string strK;
01583         bool EmptyResult = false;
01584 
01585         if(log)
01586             MRLog() « "Extracting 1-Sequences set..." « std::endl;
01587 
01588         if(!extract1Sequences()) {
01589 
01590             int singleton_num = pruning1Seq();
01591             if(singleton_num>0) {
01592                 createBitVectors();
01593                 if(singleton_num>1 && max_seq_length>1) {
01594                     if(log)
01595                         MRLog() « "Extracting 2-Sequences set..."«std::endl;
```

```
01596                     EmptyResult = extract2Sequences();
01597                     if(!EmptyResult && max_seq_length==2) {
01598                         saveSequences(2);
01599                         delete_all(2);
01600                     }
01601 
01602                     while(!EmptyResult && k <= max_seq_length) {
01603                         if(k%10 == 0)
01604                             MRLog()«"Please wait..."«std::endl;
01605                         strK = Converter(k).toString();
01606                         if(log)
01607                             MRLog() « "Extracting " + strK + "-Sequences set..."«std::endl;
01608                         EmptyResult = extractKSequences(k);
01609 
01610                         if(EmptyResult)
01611                             MRLog()« "Cannot generate sequences of length " + strK « std::endl;
01612                         else if(k==max_seq_length) {
01613                             saveSequences(k);
01614                             delete_all(k);
01615                         }
01616                         ++k;
01617                     }
01618                 }
01619                 else {
01620                     saveSequences(1);
01621                     delete_all(1);
01622                 }
01623             }//nothing generated...
01624             else
01625                 delete_all(1);
01626         }//Source table is empty
01627         else
01628             k=2;
01629         std::string empty_t_name=pm.tab_result +"_Seq"+ Converter(k-1).toString();
01630         dropTable(empty_t_name);
01631         dropTable(pm.tab_result+"_Source");
01632         MRLogPop();
01633         MRLogPush("Saving frequent patterns...");
01634         connection.finalize(true);
01635         finalize_matcher(seq_count>1);
01636         MRLogPop();
01637         show_Results(log);
01638         std::cout«std::endl;
01639         MRLog()«"Extracted "+Converter(seq_count-1).toString()+" sequences."«std::endl;
01640     } catch (const mrdb::SQLException& e) {
01641         std::cout«StringUtils::toBoldRed("MRDB Exception:");
01642         std::cout«e.what() «std::endl;
01643         dropResultsTables(k);
01644         delete_all(-1);
01645         throw e;
01646     } catch (const minerule::MineruleException& e) {
01647         std::cout«StringUtils::toBoldRed("Minerule Exception:");
01648         std::cout«e.what()«std::endl;
01649         dropResultsTables(k);
01650         delete_all(-1);
01651         throw e;
01652     } catch (const std::exception& e) {
01653         std::cout«StringUtils::toBoldRed("Exception:");
01654         std::cout«e.what()«std::endl;
01655         dropResultsTables(k);
01656         delete_all(-1);
01657         throw e;
01658     }   catch(...) {
01659         std::cout«StringUtils::toBoldRed("An unknown exception has been thrown... \n ");
01660         dropResultsTables(k);
01661         delete_all(-1);
01662         throw std::exception();
01663     }
01664 }
01665 
01666 }//end namespace
```

7.245 /Users/esposito/Software/minerule/src/Database/Connection.cpp File Reference

```
#include "minerule/Database/Connection.hpp"
#include <sstream>
#include <fstream>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <unistd.h>
```

Namespaces

- namespace minerule

7.246 Connection.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Database/Connection.hpp"
00002
00003 #include <sstream>
00004 #include <fstream>
00005 #include <sys/types.h>
00006 #include <sys/stat.h>
00007 #include <unistd.h>
00008
00009 namespace minerule {
00010
00011         void
00012         Connection::useMRDBConnection(mrdb::Connection* newConnection) {
00013           connection = newConnection;
00014         }
00015
00016         std::string Connection::getTableName(TableKind kind) const {
00017                 switch(kind) {
00018                         case RulesTable:
00019                                 return outTableName;
00020                                 break;
00021                         case HeadsTable:
00022                                 return outTableName + "_head_elems";
00023                                 break;
00024                         case BodiesTable:
00025                                 return outTableName + "_body_elems";
00026                                 break;
00027                         case SeqTable:
00028           return outTableName + "_Seq_support";
00029           break;
00030       case SeqElTable:
00031           return outTableName + "_Seq";
00032           break;
00033       case ElemTable:
00034           return outTableName + "_Seq1";
00035           break;
00036                         default:
00037                                 throw MineruleException(MR_ERROR_INTERNAL, "Unknown TableKind -- this
      is a bug, please report it!");
00038                 }
00039         }
00040
00041
00042
00043         bool Connection::tableExists(const char * tableName){
00044                         std::string table(tableName, strlen(tableName));
00045                         std::string query = "SELECT relname FROM pg_class WHERE upper(relname) =
      upper('"+table+"');";
00046                         mrdb::Statement* stmt=connection->createStatement();
00047                         mrdb::ResultSet* rs = stmt->executeQuery(query);
00048                         bool result = rs->next();
00049                         delete rs;
00050                         delete stmt;
00051                         return result;
00052         }
00053
00054         void Connection::deleteDestTables() {
00055           deleteTable(getTableName(RulesTable).c_str());
00056           deleteTable(getTableName(HeadsTable).c_str());
00057           deleteTable(getTableName(BodiesTable).c_str());
00058
00059       deleteTable(getTableName(SeqTable).c_str());
00060       deleteTable(getTableName(SeqElTable).c_str());
00061       deleteTable(getTableName(ElemTable).c_str());
00062     }
00063
```

```
00064     void Connection::deleteTable(const char * tableName)
00065     {
00066             mrdb::Statement* stmt=connection->createStatement();
00067             MRLog() « "Dropping table " « tableName « std::endl;
00068             stmt->execute(std::string("DROP TABLE IF EXISTS ")+tableName);
00069             delete stmt;
00070
00071     }
00072
00073
00074         void Connection::createResultTables(const SourceRowMetaInfo& srd){
00075
00076         std::cout«"Calling mine rule dest table function creation"«std::endl;
00077
00078         if(MineruleOptions::getSharedOptions().getSafety().getOverwriteHomonymMinerules())
00079                         deleteDestTables();
00080
00081                 mrdb::Statement* statement=connection->createStatement();
00082                 std::string create, create_index;
00083
00084                 // Creating the rules table
00085                 if(!tableExists(getTableName(RulesTable).c_str())){
00086                         create=std::string("CREATE TABLE ")+getTableName(RulesTable)+" (bodyId int,
    headId int, supp float, con float, cardBody int, cardHead int );";
00087                         statement->execute(create);
00088                 }
00089
00090                 // Creating the body elements table
00091                 if(!tableExists(getTableName(BodiesTable).c_str())){
00092                         create=std::string("CREATE TABLE ")+ getTableName(BodiesTable) + " (id int, "
    + srd.getBody().getSQLDataDefinition() +")";
00093                         create_index = " CREATE INDEX "+getTableName(BodiesTable)+"_index ON " +
    getTableName(BodiesTable) + " (id);";
00094
00095                         statement->execute(create);
00096                         statement->execute(create_index);
00097                 }
00098                 if(srd.getHead().getColumnsCount() > 0) {
00099                                 // Creating the head elements table
00100                                 if(!tableExists(getTableName(HeadsTable).c_str())){
00101                                         create=std::string("CREATE TABLE ")+ getTableName(HeadsTable)
    +  " (id int, " + srd.getHead().getSQLDataDefinition() +")";
00102
00103                                         create_index = " CREATE INDEX
    "+getTableName(HeadsTable)+"_index ON " + getTableName(HeadsTable) + " (id);";
00104
00105                                         statement->execute(create);
00106                                         statement->execute(create_index);
00107                                 }
00108
00109                                 std::string headInserterQuery = "INSERT INTO " +
    getTableName(HeadsTable) + " VALUES (?,"+ srd.getHead().questionMarks() +")";
00110
    dbInserter->setHeadInserter(connection->prepareStatement(headInserterQuery));
00111                 }
00112                 std::string bodyInserterQuery = "INSERT INTO " + getTableName(BodiesTable) + " VALUES
    (?," + srd.getBody().questionMarks() + ")";
00113                 dbInserter->setBodyInserter(connection->prepareStatement(bodyInserterQuery));
00114
00115                 delete statement;
00116
00117
00118          }
00119
00120         //overload for sequences
00121         void Connection::createResultTables(){
00122
00123                 std::cout«"Calling mine sequence dest table function creation"«std::endl;
00124
00125                 if(tableExists(getTableName(SeqTable).c_str()) &&
    MineruleOptions::getSharedOptions().getSafety().getOverwriteHomonymMinerules())
00126                                 deleteDestTables();
00127
00128                 mrdb::Statement* statement=connection->createStatement();
00129                 std::string create, create_index;
00130
00131                 // Creating the seq table
00132                 if(!tableExists(getTableName(SeqTable).c_str())){
00133                         create=std::string("CREATE TABLE ")+getTableName(SeqTable)+" (ID int, supp
    float );";
00134                         create_index = " CREATE INDEX "+getTableName(SeqTable)+"_index ON " +
    getTableName(SeqTable) + " (id);";
00135                         statement->execute(create);
00136                         statement->execute(create_index);
00137                 }
00138
00139                 // Creating the sequences' elements table
```

```
00140                 if(!tableExists(getTableName(SeqElTable).c_str())){
00141                         create=std::string("CREATE TABLE ")+ getTableName(SeqElTable) + " (id int,idEl
      int,pos int )";
00142                         create_index = " CREATE INDEX "+getTableName(SeqElTable)+"_index ON " +
      getTableName(SeqElTable) + " (id);";
00143
00144                         statement->execute(create);
00145                         statement->execute(create_index);
00146                 }
00147
00148                 std::string seqInserterQuery = "INSERT INTO " + getTableName(SeqTable) + " VALUES (?,
      ?)";
00149                 dbInserter->setSeqInserter(connection->prepareStatement(seqInserterQuery));
00150
00151                 std::string seqElInserterQuery = "INSERT INTO " + getTableName(SeqElTable) + " VALUES
      (?, ?, ?)";
00152                 dbInserter->setSeqElInserter(connection->prepareStatement(seqElInserterQuery));
00153         }
00154
00155
00156         void Connection::DirectDBInserter::insertHeadBodyElems(TableKind kind, const ItemSet& elems,
      size_t counter) {
00157           MRLogStartMeasuring("Head/Body Insertion time:");
00158
00159           assert( !elems.empty() );
00160           mrdb::PreparedStatement* state;
00161           switch(kind) {
00162                   case BodiesTable:
00163                           state = bodyInserter;
00164                           break;
00165                   case HeadsTable:
00166                           state = headInserter;
00167                           break;
00168                   default:
00169                           throw MineruleException(MR_ERROR_INTERNAL, "bad call to insertHeadBodyElems.
      This is a bug! Please report it!");
00170           }
00171
00172           ItemSet::const_iterator it = elems.begin();
00173           for(; it!=elems.end() ; it++ ) {
00174             state->setLong(1,counter);
00175                 it->setPreparedStatementParameters(state,2);
00176             state->execute();
00177           }
00178
00179           MRLogStopMeasuring("Head/Body Insertion time:");
00180         }
00181
00182
00183         void Connection::CachedDBInserter::insertHeadBodyElems(TableKind kind, const ItemSet& elems,
      size_t id) {
00184         //  assert( !elems.empty() );
00185             MRLogStartMeasuring("Head/Body Insertion time:");
00186
00187                 std::ofstream* outStream = NULL;
00188                 switch(kind) {
00189                         case RulesTable:
00190                                 outStream = &outR;
00191                                 break;
00192                         case HeadsTable:
00193                                 outStream = &outH;
00194                                 break;
00195                         case BodiesTable:
00196                                 outStream = &outB;
00197                                 break;
00198                         default:
00199                                 throw MineruleException( MR_ERROR_INTERNAL, "Unexpected TableKind
      value." );
00200                 }
00201
00202                 ItemSet::const_iterator it = elems.begin();
00203                 std::string str;
00204                 for(; it!=elems.end() ; it++ ) {
00205                         (*outStream) « id « "\t" « it->getElement().asString("\t") « std::endl;
00206                 }
00207
00208             MRLogStopMeasuring("Head/Body Insertion time:");
00209         }
00210
00211         void Connection::CachedDBInserter::init() {
00212           MRLogStartMeasuring("CachedDBInserter init");
00213
00214           char tmpFName[30];
00215           strcpy(tmpFName,"/tmp/cacheDBInserterXXXXXX");
00216           if(mkstemp(tmpFName)==-1) {
00217                   throw MineruleException(MR_ERROR_OUTPUT_FILE_PROBLEM, std::string("Cannot create
      temporary file:")+tmpFName);
```

```
00218           }
00219
00220           filename = tmpFName;
00221
00222           MRLog() « "CachedDBInserter will use filestem: " « filename « std::endl;
00223
00224           std::string temp = filename + ".r";
00225           outR.open(temp.c_str());
00226
00227           temp = filename + ".h";
00228           outH.open(temp.c_str());
00229
00230           temp = filename + ".b";
00231           outB.open(temp.c_str());
00232           MRLogStopMeasuring("CachedDBInserter init");
00233         }
00234
00235     void Connection::CachedDBInserter::finalize() {
00236           MRLogStartMeasuring("CachedDBInserter finalize");
00237
00238           outR.close();
00239           outH.close();
00240           outB.close();
00241
00242           std::string loadstr1 = filename + ".r";
00243           if(chmod(loadstr1.c_str(),S_IRUSR|S_IRGRP|S_IROTH)==-1) {
00244                   throw MineruleException( MR_ERROR_INTERNAL,
00245                           std::string("Cannot change permissions on file ")+loadstr1 +
00246                           " reason is:" + strerror(errno));
00247           }
00248
00249           std::string loadstr2 = filename + ".h";
00250           if(chmod(loadstr2.c_str(),S_IRUSR|S_IRGRP|S_IROTH)==-1) {
00251                   throw MineruleException( MR_ERROR_INTERNAL,
00252                           std::string("Cannot change permissions on file ")+loadstr2 +
00253                           " reason is:" + strerror(errno));
00254           }
00255
00256           std::string loadstr3 = filename + ".b";
00257           if(chmod(loadstr3.c_str(),S_IRUSR|S_IRGRP|S_IROTH)==-1) {
00258                   throw MineruleException( MR_ERROR_INTERNAL,
00259                           std::string("Cannot change permissions on file ")+loadstr3 +
00260                           " reason is:" + strerror(errno));
00261           }
00262
00263           const std::string& dbms = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00264           if(  dbms == "mysql" ) {
00265                   loadstr1 = "LOAD DATA INFILE '" + filename + ".r' INTO TABLE " +
    connection.getTableName(RulesTable);
00266                   loadstr2 = "LOAD DATA INFILE '" + filename + ".h' INTO TABLE " +
    connection.getTableName(HeadsTable);
00267                   loadstr3 = "LOAD DATA INFILE '" + filename + ".b' INTO TABLE " +
    connection.getTableName(BodiesTable);
00268           } else if( dbms == "postgres" ) {
00269                   loadstr1 = "COPY " + connection.getTableName(RulesTable)  +" FROM '" + filename +
    ".r'";
00270                   loadstr2 = "COPY " + connection.getTableName(HeadsTable)  +" FROM '" + filename +
    ".h'";
00271                   loadstr3 = "COPY " + connection.getTableName(BodiesTable) +" FROM '" + filename +
    ".b'";
00272           } else  {
00273                   throw MineruleException( MR_ERROR_OPTION_CONFIGURATION,
00274                           "Option for key mrdb::dbms is not set properly, it is set to "+dbms+
00275                           ", but only 'mysql' or 'postgres' are supported." );
00276           }
00277
00278           mrdb::Statement* state = connection.getMRDBConnection()->createStatement();
00279           state->execute(loadstr1.c_str());
00280           state->execute(loadstr2.c_str());
00281           state->execute(loadstr3.c_str());
00282
00283           delete state;
00284
00285           loadstr1 = filename + ".r";
00286           unlink(loadstr1.c_str());
00287
00288           loadstr2 = filename + ".h";
00289           unlink(loadstr2.c_str());
00290
00291           loadstr2 = filename + ".b";
00292       unlink(loadstr2.c_str());
00293
00294           MRLogStopMeasuring("CachedDBInserter finalize");
00295         }
00296
00297         //overload for sequences
00298    void Connection::CachedDBInserter::finalize(bool seq) {
```

```
00299     MRLogStartMeasuring("CachedDBInserter finalize");
00300     if(seq) std::cout«"Saving sequences on db..."«std::endl;
00301     outH.close();
00302     outB.close();
00303 
00304     std::string loadstr2 = filename + ".h";
00305     if(chmod(loadstr2.c_str(),S_IRUSR|S_IRGRP|S_IROTH)==-1) {
00306         throw MineruleException( MR_ERROR_INTERNAL,
00307             std::string("Cannot change permissions on file ")+loadstr2 +
00308             " reason is:" + strerror(errno));
00309     }
00310 
00311     std::string loadstr3 = filename + ".b";
00312     if(chmod(loadstr3.c_str(),S_IRUSR|S_IRGRP|S_IROTH)==-1) {
00313         throw MineruleException( MR_ERROR_INTERNAL,
00314             std::string("Cannot change permissions on file ")+loadstr3 +
00315             " reason is:" + strerror(errno));
00316     }
00317 
00318     const std::string& dbms = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00319     if(  dbms == "mysql" ) {
00320         loadstr2 = "LOAD DATA INFILE '" + filename + ".h' INTO TABLE " +
    connection.getTableName(SeqTable);
00321         loadstr3 = "LOAD DATA INFILE '" + filename + ".b' INTO TABLE " +
    connection.getTableName(SeqElTable);
00322     } else if( dbms == "postgres" ) {
00323         loadstr2 = "COPY " + connection.getTableName(SeqTable)  +" FROM '" + filename + ".h'";
00324         loadstr3 = "COPY " + connection.getTableName(SeqElTable) +" FROM '" + filename + ".b'";
00325     } else  {
00326         throw MineruleException( MR_ERROR_OPTION_CONFIGURATION,
00327             "Option for key mrdb::dbms is not set properly, it is set to "+dbms+
00328             ", but only 'mysql' or 'postgres' are supported." );
00329     }
00330 
00331     mrdb::Statement* state = connection.getMRDBConnection()->createStatement();
00332     state->execute(loadstr2.c_str());
00333     state->execute(loadstr3.c_str());
00334 
00335     delete state;
00336 
00337     loadstr2 = filename + ".h";
00338     unlink(loadstr2.c_str());
00339 
00340     loadstr3 = filename + ".b";
00341     unlink(loadstr3.c_str());
00342 
00343     MRLogStopMeasuring("CachedDBInserter finalize");
00344   }
00345 
00346         void Connection::CachedDBInserter::insert(const ItemSet& body,
00347                                   const ItemSet& head,
00348                                   double support,
00349                                   double confidence,
00350                                   bool saveBody) {
00351           // if either body or head elements are too many or too few with respect to
00352           // the allowed cardinalities, we simply return back to the caller.
00353           if( !connection.bodyCard.validate(body.size()) ||
    !connection.headCard.validate(head.size()))
00354             return;
00355 
00356           MRLogStartMeasuring("Rule Insertion");
00357 
00358           static size_t counter=0;
00359 
00360           static size_t bodyId = counter;
00361           if (saveBody) {
00362                 bodyId = counter++;
00363                 insertHeadBodyElems(BodiesTable, body, bodyId);
00364           }
00365 
00366           size_t headId = counter++;
00367           insertHeadBodyElems(HeadsTable, head, headId);
00368 
00369           outR « bodyId « "\t" « headId « "\t" « support « "\t" « confidence « "\t" « body.size() «
    "\t" « head.size() « std::endl;
00370 
00371           MRLogStopMeasuring("Rule Insertion");
00372         }
00373 
00374         void Connection::DirectDBInserter::insert(const ItemSet& body,
00375                                   const ItemSet& head,
00376                                   double support,
00377                                   double confidence,
00378                         bool saveBody) {
00379 
00380           // if either body or head, is too big, too low with respect to
00381           // the allowed cardinalities we simply return back to the caller.
```

```
00382           if( !connection.bodyCard.validate(body.size()) ||
        !connection.headCard.validate(head.size()))
00383             return;
00384
00385           MRLogStartMeasuring("Rule Insertion");
00386
00387           static size_t counter=0;
00388
00389           static mrdb::Statement* state = connection.connection->createStatement();
00390
00391           static size_t bodyId = counter;
00392           if (saveBody) {
00393                 bodyId = counter++;
00394                 insertHeadBodyElems(BodiesTable, body, bodyId);
00395           }
00396
00397           size_t headId = counter++;
00398           insertHeadBodyElems(HeadsTable, head, headId);
00399
00400           std::stringstream query;
00401           query « "INSERT INTO " « connection.getTableName(RulesTable)
00402                       « " VALUES (" « bodyId « "," « headId « ","
00403                       « support « "," « confidence « "," « body.size() « "," « head.size() « ")" ;
00404
00405           state->execute(query.str());
00406
00407           // delete state;
00408           MRLogStopMeasuring("Rule Insertion");
00409         }
00410
00411         void Connection::DirectDBInserter::insert(int seqId, int seqEl, double support, int pos, bool
        saveSeqId) {
00412
00413     MRLogStartMeasuring("Sequences Insertion time:");
00414
00415     mrdb::PreparedStatement* state;
00416     if(saveSeqId) {
00417                         state = seqInserter;
00418       state->setInt(1,seqId);
00419       state->setDouble(2,support);
00420       state->execute();
00421     }
00422     state = seqElInserter;
00423     state->setInt(1,seqId);
00424     state->setInt(2,seqEl);
00425     state->setInt(3,pos);
00426     state->execute();
00427
00428                 MRLogStopMeasuring("Sequences Insertion time:");
00429
00430   }
00431
00432   void Connection::CachedDBInserter::insert(int seqId, int seqEl, double support, int pos, bool
        saveSeqId) {
00433     MRLogStartMeasuring("Sequences Insertion time:");
00434
00435     std::ofstream* outStream = NULL;
00436     if(saveSeqId) {
00437       outStream = &outH;
00438       (*outStream) « seqId « "\t"« support « std::endl;
00439     }
00440     outStream = &outB;
00441
00442     (*outStream) « seqId « "\t" « seqEl « "\t" « pos « std::endl;
00443
00444     MRLogStopMeasuring("Sequences Insertion time:");
00445   }
00446
00447         void Connection::insert(const char * what){
00448             static mrdb::Statement* statement=connection->createStatement();
00449             statement->execute(what);
00450             delete statement;
00451          }
00452
00453
00454         void Connection::create_tmp_db(int sintax,
00455                                        const SourceRowAttrCollectionDescriptor& body,
00456                                        const SourceRowAttrCollectionDescriptor& head)
00457          {
00458             mrdb::Statement* statement=connection->createStatement();
00459             std::string create;
00460
00461             delete_tmp_db();
00462
00463             switch (sintax)
00464             {
00465                 case 0 : create="CREATE TABLE tmp_Rule_Base(id int AUTO_INCREMENT PRIMARY KEY,level
```

```
      int, "
00466                           + body.getSQLDataDefinition() +");";
00467                         break;
00468                 case 1 : create=
00469                           "CREATE TABLE tmp_Rule_Ext(id int AUTO_INCREMENT PRIMARY KEY,level int,"
00470                           + body.getSQLDataDefinition() + ", id_head char(13));";
00471                         break;
00472             }
00473
00474             statement->execute(create);
00475
00476             if (sintax==1)
00477             {
00478               create="CREATE TABLE tmp_Rule_Head_Ext(id int AUTO_INCREMENT PRIMARY KEY,id_head
      char(13),level int," + head.getSQLDataDefinition()+");";
00479               statement->execute(create);
00480             }
00481             delete statement;
00482          }
00483
00484         void Connection::delete_tmp_db()
00485          {
00486
00487           deleteTable("tmp_Rule_Base");
00488           deleteTable("tmp_Rule_Ext");
00489           deleteTable("tmp_Rule_Head_Ext");
00490
00491          }
00492
00493 }
```

7.247 /Users/esposito/Software/minerule/src/Database/PrepareData↩ Utils.cpp File Reference

```
#include "minerule/Database/PrepareDataUtils.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include <string>
#include <memory>
```

Namespaces

- namespace minerule

Macros

- #define ADDCOLALIAS (addColAlias?(" AS " + alias+∗it):"")
- #define ADDCOLALIAS (addColAlias?(alias+∗it):"")

7.247.1 Macro Definition Documentation

7.247.1.1 ADDCOLALIAS [1/2]

```
#define ADDCOLALIAS (addColAlias?(" AS " + alias+*it):"")
```

7.247.1.2 ADDCOLALIAS [2/2]

```
#define ADDCOLALIAS (addColAlias?(alias+*it):"")
```

7.248 PrepareDataUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/PrepareDataUtils.hpp"
00017 #include "minerule/Utils/MineruleOptions.hpp"
00018
00019 #include <string>
00020 #include <memory>
00021
00022
00023
00024 namespace minerule {
00025
00026         std::string PrepareDataUtils::buildAttrListDescription( const ParsedMinerule::AttrVector&
      attrs, const std::string& alias, bool addColAlias)  {
00027
00028                 #define ADDCOLALIAS (addColAlias?(" AS " + alias+*it):"")
00029
00030                 ParsedMinerule::AttrVector::const_iterator it;
00031                 std::string result;
00032                 it=attrs.begin();
00033
00034     // the first element should not have a leading ","
00035                 assert(it!=attrs.end());
00036                 if(alias.length()!=0)
00037                         result=alias+"."+*it + ADDCOLALIAS;
00038                 else
00039                         result=*it;
00040
00041                 it++;
00042
00043                 for( ;it!=attrs.end(); it++ ) {
00044                         if(alias.length()!=0)
00045                                 result += ","+alias+"."+*it + ADDCOLALIAS;
00046                         else
00047                                 result+=","+*it;
00048                 }
00049
00050                 #undef ADDCOLALIAS
00051
00052                 return result;
00053         }
00054
00055         std::string
00056         PrepareDataUtils::buildAttrListAlias( const ParsedMinerule::AttrVector& attrs,        const
      std::string& alias, bool addColAlias)  {
00057                 #define ADDCOLALIAS (addColAlias?(alias+*it):"")
00058
00059                 ParsedMinerule::AttrVector::const_iterator it;
00060                 std::string result;
00061                 it=attrs.begin();
00062
00063     // the first element should not have a leading ","
00064                 assert(it!=attrs.end());
00065                 if(alias.length()!=0)
00066                         result=ADDCOLALIAS;
00067                 else
00068                         result=*it;
00069
00070                 it++;
```

```
00071
00072                 for( ;it!=attrs.end(); it++ ) {
00073                         if(alias.length()!=0)
00074                                 result += ","+ADDCOLALIAS;
00075                         else
00076                                 result+=","+*it;
00077                 }
00078
00079                 #undef ADDCOLALIAS
00080
00081                 return result;
00082         }
00083
00084         std::string PrepareDataUtils::buildAttrListEquiJoin( const std::string& alias1, const
      std::string& alias2) const {
00085
00086                 ParsedMinerule::AttrVector::const_iterator it;
00087                 std::string result;
00088                 it=mr.ga.begin();
00089
00090     // *** Filtering out rows not having the same gid ***
00091     // the first element should not have a leading ","
00092                 assert(it!=mr.ga.end());
00093                 result=alias1+"."+*it+"="+alias2+"."+*it;
00094                 it++;
00095
00096                 for( ;it!=mr.ga.end(); it++ ) {
00097                         result+=" AND "+ alias1+"."+*it+"="+alias2+"."+*it;
00098                 }
00099
00100     // *** Filtering out rows having the same cid and the same item
00101
00102     // *** if body!=head there will be no tautologies and hence we
00103     // *** cannot actually filter out anything
00104                 if( !mr.body_coincident_head ) {
00105                         MRLog() << "filtering query:" << result << std::endl;
00106                         return result;
00107                 }
00108
00109                 result+=" AND NOT (";
00110                 if( !mr.ca.empty() ) {
00111                         it=mr.ca.begin();
00112                         assert(it!=mr.ca.end());
00113                         result+=alias1+"."+*it+"="+alias2+"."+*it;
00114                         it++;
00115
00116                         for( ;it!=mr.ca.end(); it++ ) {
00117                                 result+=" AND "+ alias1+"."+*it+"="+alias2+"."+*it;
00118                         }
00119
00120                         result+=" AND ";
00121                 }
00122
00123                 it=mr.ba.begin();
00124                 assert(it!=mr.ba.end());
00125                 result+=alias1+"."+*it+"="+alias2+"."+*it;
00126                 it++;
00127
00128                 for( ;it!=mr.ba.end(); it++ ) {
00129                         result+=" AND "+ alias1+"."+*it+"="+alias2+"."+*it;
00130                 }
00131
00132                 result +=")";
00133
00134                 return result;
00135         }
00136
00137
00138
00139
00140         std::string
00141         PrepareDataUtils::buildBodyTableQuery(SourceRowColumnIds& rowDes, const std::string&
      body_mining_condition) const {
00142                 std::string queryText;
00143
00144                 queryText = "SELECT ";
00145                 queryText += buildAttrListDescription(mr.ga);
00146                 queryText += "," + buildAttrListDescription(mr.ba);
00147                 queryText += " FROM "+mr.tab_source;
00148                 if(!body_mining_condition.empty())
00149                         queryText += " WHERE "+body_mining_condition;
00150
00151                 if( sourceTableRequirements.sortedGids() )
00152                         queryText += " ORDER BY "+buildAttrListDescription(mr.ga);
00153
00154                 unsigned int lastElem;
00155                 lastElem= rowDes.setGroupElems(1,mr.ga.size());
```

```
00156                 rowDes.setBodyElems(lastElem+1,mr.ba.size());
00157
00158                 return queryText;
00159         }
00160
00161         std::string
00162         PrepareDataUtils::buildHeadTableQuery(SourceRowColumnIds& rowDes, const std::string&
      head_mining_condition) const {
00163                 std::string queryText;
00164
00165                 queryText = "SELECT ";
00166                 queryText += buildAttrListDescription(mr.ga);
00167                 queryText += "," + buildAttrListDescription(mr.ha);
00168                 queryText += " FROM "+mr.tab_source;
00169
00170                 if(!head_mining_condition.empty())
00171                         queryText += " WHERE "+head_mining_condition;
00172
00173                 if( sourceTableRequirements.sortedGids() )
00174                         queryText += " ORDER BY "+buildAttrListDescription(mr.ga);
00175
00176                 unsigned int lastElem;
00177                 lastElem= rowDes.setGroupElems(1,mr.ga.size());
00178                 rowDes.setHeadElems(lastElem+1,mr.ha.size());
00179
00180                 return queryText;
00181         }
00182
00183         std::string PrepareDataUtils::buildAndList(const list_AND_node* l) const {
00184                 const list_AND_node* current = l;
00185                 std::string result;
00186
00187                 assert( current!=NULL );
00188                 assert( current->sp!=NULL );
00189
00190                 result = std::string(current->sp->val1) + std::string(current->sp->op) +
      std::string(current->sp->val2);
00191                 current = current->next;
00192
00193                 while( current!=NULL ) {
00194                         assert(current->sp!=NULL );
00195                         result += " AND "+ std::string(current->sp->val1) +
      std::string(current->sp->op) + std::string(current->sp->val2);
00196                         current = current->next;
00197                 }
00198
00199                 return result;
00200         }
00201
00202         std::string
00203         PrepareDataUtils::buildConditionFilter(const list_OR_node* current) const {
00204                 std::string result;
00205
00206                 if( current==NULL )
00207                         return "";
00208
00209                 result = "("+ buildAndList(current->l_and)+")";
00210                 current = current->next;
00211
00212                 while( current!=NULL ) {
00213                         result += " OR (" + buildAndList(current->l_and) + ")";
00214                         current = current->next;
00215                 }
00216
00217                 return result;
00218         }
00219
00220         void
00221         PrepareDataUtils::dropTableIfExists(mrdb::Connection* conn, const std::string& tname)  {
00222                 std::auto_ptr<mrdb::Statement> state(conn->createStatement());
00223                 state->execute("DROP TABLE IF EXISTS "+tname);
00224         }
00225
00226         std::string
00227         PrepareDataUtils::createSourceTable() const {
00228                 std::string aliasA, aliasB;
00229                 std::string queryText;
00230
00231                 assert( !(mr.ga.empty() || mr.ba.empty() || mr.ha.empty()) );
00232
00233                 aliasA = "BODY";
00234                 aliasB = "HEAD";
00235
00236                 std::string groupAttrList =  buildAttrListDescription(mr.ga, aliasA,true);
00237
00238                 queryText = "SELECT ";
00239                 queryText += groupAttrList;
```

```
00240                 if( !mr.ca.empty() )
00241                         queryText += "," + buildAttrListDescription(mr.ca, aliasA,true);
00242                 queryText += "," + buildAttrListDescription(mr.ba, aliasA,true);
00243                 if( !mr.ca.empty() )
00244                         queryText += "," + buildAttrListDescription(mr.ca, aliasB,true);
00245                 queryText += "," + buildAttrListDescription(mr.ha, aliasB,true);
00246 
00247                 queryText += " FROM "+mr.tab_source+" AS "+aliasA;
00248                 queryText += ","+mr.tab_source+" AS "+aliasB;
00249 
00250                 queryText += " WHERE " + buildAttrListEquiJoin( aliasA, aliasB);
00251 
00252                 std::string miningCond = buildConditionFilter(mr.mc);
00253                 if(miningCond!="") {
00254                         queryText += " AND (" + miningCond +")";
00255                 }
00256 
00257                 std::string clusterCond = buildConditionFilter(mr.cc);
00258                 if(clusterCond!="") {
00259                         queryText += " AND (" + clusterCond +")";
00260                 }
00261 
00262                 if( sourceTableRequirements.sortedGids() )
00263                         queryText += " ORDER BY "+buildAttrListDescription(mr.ga, aliasA,false);
00264 
00265                 std::string clusteredTable       = mr.tab_result+"_tmpSource";
00266                 std::string createQuery          = "CREATE TABLE  "+clusteredTable+ " AS "+queryText +
    "; ";
00267                 std::string createIndexQuery = "CREATE INDEX "+clusteredTable+"_index " + " ON " +
    clusteredTable + " ("+buildAttrListAlias(mr.ga,aliasA,true)+");";
00268 
00269                 mrdb::Connection* conn =
    MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00270 
00271                 dropTableIfExists(conn, clusteredTable);
00272 
00273                 mrdb::Statement* state = conn->createStatement();
00274 
00275                 MRLog() << "Creating clustered table... " << std::endl;
00276                 MRDebug("Creating mining table, the query is:" + createQuery);
00277 
00278                 state->execute(createQuery);
00279                 state->execute(createIndexQuery);
00280 
00281                 MRLog() << "DONE..." << std::endl;
00282 
00283                 delete state;
00284 
00285                 return  clusteredTable;
00286         }
00287 
00288 
00289         std::string PrepareDataUtils::buildExtendedSourceTableQuery(SourceRowColumnIds& rowDes) const
    {
00290                 std::string tableName = createSourceTable();
00291 
00292                 std::string queryText = "SELECT * FROM "+tableName;
00293 
00294                 unsigned int lastElem;
00295                 lastElem= rowDes.setGroupElems(1,mr.ga.size());
00296                 lastElem= rowDes.setClusterBodyElems(lastElem+1,mr.ca.size());
00297                 lastElem= rowDes.setBodyElems(lastElem+1,mr.ba.size());
00298                 lastElem= rowDes.setClusterHeadElems(lastElem+1,mr.ca.size());
00299                 rowDes.setHeadElems(lastElem+1, mr.ha.size());
00300 
00301                 return queryText;
00302         }
00303 
00304 
00305         std::string PrepareDataUtils::buildSourceTableQuery(SourceRowColumnIds& rowDes) const {
00306                 if(sourceTableRequirements.crossProduct() ) {
00307                         return buildExtendedSourceTableQuery(rowDes);
00308                 } else {
00309                         throw MineruleException(MR_ERROR_INTERNAL, "Check for consistency with
    previous implementation");
00310                         return buildBodyTableQuery(rowDes,"");
00311                 }
00312         }
00313 
00314 
00315         size_t PrepareDataUtils::evaluateTotGroups(const ParsedMinerule& pmr) {
00316                 MRLogPush("Evaluating Number of groups...");
00317                 std::string qry=
00318                         "SELECT count(distinct " + buildAttrListDescription(pmr.ga) +") "
00319                                 "FROM " + pmr.tab_source;
00320 
00321                 mrdb::Connection* conn =
```

```
00322                         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00323
00324                 std::auto_ptr<mrdb::Statement> state(conn->createStatement());
00325                 std::auto_ptr<mrdb::ResultSet> rs(state->executeQuery(qry));
00326
00327                 MRLogPop();
00328
00329                 if(!rs->next())
00330                         throw MineruleException(MR_ERROR_DATABASE_ERROR, "Cannot count the number of
      groups in the source table");
00331
00332                 return rs->getInt(1);
00333         }
00334
00335 } // namespace
```

7.249 /Users/esposito/Software/minerule/src/Database/SourceRow.cpp File Reference

```
#include "minerule/Database/SourceRow.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
```

Namespaces

- namespace minerule

Macros

- #define CREATE(a) (((rhs.a)!=NULL)?(a)=rhs.a->copy():(a)=NULL)
- #define DESTROY(a) if((a)!=NULL) delete a;
- #define DESTROY(a) if((a)!=NULL) delete a;

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const SourceRow &sr)

7.249.1 Macro Definition Documentation

7.249.1.1 CREATE

```
#define CREATE(
              a ) (((rhs.a)!=NULL)?(a)=rhs.a->copy():(a)=NULL)
```

7.249.1.2 DESTROY [1/2]

```
#define DESTROY(
              a ) if((a)!=NULL) delete a;
```

7.249.1.3 DESTROY [2/2]

```
#define DESTROY(
              a ) if((a)!=NULL) delete a;
```

7.250 SourceRow.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/SourceRow.hpp"
00017 #include "minerule/Utils/MineruleLogs.hpp"
00018
00019 namespace minerule {
00020
00021   const SourceRowEmptyElement SourceRow::emptyElement;
00022
00023 /* ************ */
00024 /* SourceRow methods */
00025 /* ************ */
00026
00027   SourceRow::SourceRow(mrdb::ResultSet* resultSet,
00028                                       const SourceRowColumnIds& srd) {
00029     mrdb::ResultSetMetaData* rsmd = resultSet->getMetaData();
00030
00031     group =    SourceRowElement::createElement(rsmd,resultSet,srd.groupElems);
00032     clusterBody =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterBodyElems);
00033     body =         SourceRowElement::createElement(rsmd,resultSet,srd.bodyElems);
00034     clusterHead =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterHeadElems);
00035     head =         SourceRowElement::createElement(rsmd,resultSet,srd.headElems);
00036   }
00037
00038   SourceRow::SourceRow(const SourceRow& rhs) {
00039 #define CREATE(a) (((rhs.a)!=NULL)?(a)=rhs.a->copy():(a)=NULL)
00040
00041     CREATE(group);
00042     CREATE(clusterBody);
00043     CREATE(body);
00044     CREATE(clusterHead);
00045     CREATE(head);
00046
00047 #undef CREATE
00048   }
00049
00050
00051
00052 SourceRow::~SourceRow() {
00053 #define DESTROY(a) if((a)!=NULL) delete a;
00054
00055   DESTROY(group);
00056   DESTROY(clusterBody);
00057   DESTROY(body);
00058   DESTROY(clusterHead);
```

```
00059   DESTROY(head);
00060
00061 #undef DESTROY
00062 }
00063
00064 void
00065 SourceRow::init(mrdb::ResultSet* resultSet,
00066                         const SourceRowColumnIds& srd) {
00067
00068 #define DESTROY(a) if((a)!=NULL) delete a;
00069
00070   DESTROY(group);
00071   DESTROY(clusterBody);
00072   DESTROY(body);
00073   DESTROY(clusterHead);
00074   DESTROY(head);
00075
00076   mrdb::ResultSetMetaData* rsmd = resultSet->getMetaData();
00077
00078   group =    SourceRowElement::createElement(rsmd,resultSet,srd.groupElems);
00079   clusterBody =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterBodyElems);
00080   body =         SourceRowElement::createElement(rsmd,resultSet,srd.bodyElems);
00081   clusterHead =  SourceRowElement::createElement(rsmd,resultSet,srd.clusterHeadElems);
00082   head =         SourceRowElement::createElement(rsmd,resultSet,srd.headElems);
00083
00084 #undef DESTROY
00085 }
00086
00087
00088
00089 /* ***************** *
00090  * printing routines *
00091  * ***************** */
00092
00093 std::ostream&
00094 operator«(std::ostream& os, const SourceRow& sr) {
00095   os « "group:" « sr.getGroup() « std::endl;
00096   os « " clusterBody:" « sr.getClusterBody() « std::endl;
00097   os « " body:" « sr.getBody() « std::endl;
00098   os « " clusterHead:" « sr.getClusterHead() « std::endl;
00099   os « " head:" « sr.getHead() « std::endl;
00100
00101   return os;
00102 }
00103
00104 } // end namespace
```

7.251 /Users/esposito/Software/minerule/src/Database/SourceRow↩ Attribute.cpp File Reference

```
#include "minerule/Database/SourceRowAttribute.hpp"
#include <sstream>
#include "minerule/Utils/Converter.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
```

Namespaces

- namespace minerule

7.252 SourceRowAttribute.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
```

```
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/SourceRowAttribute.hpp"
00017 #include <sstream>
00018 #include "minerule/Utils/Converter.hpp"
00019
00020 #include "minerule/mrdb/ResultSetMetaData.hpp"
00021
00022 //#define DEBUG
00023
00024 namespace minerule {
00025
00026   /********* static members *********/
00027
00028         size_t MemDebugGenericSourceRowAttribute::instanceCount = 0;
00029
00033                 SourceRowAttribute*
00034                         SourceRowAttribute::createAttribute(mrdb::ResultSetMetaData* rsmd,
00035                                 mrdb::ResultSet* rs,
00036
00037         int elem) {
00038                 mrdb::Types::SQLType colType = (mrdb::Types::SQLType) rsmd->getColumnType(elem);
00039 #ifdef DEBUG
00040 #warning MemDebugGeneric...
00041                 return new MemDebugGenericSourceRowAttribute(rs,elem,colType);
00042 #else
00043                 if(colType==mrdb::Types::INTEGER || colType==mrdb::Types::BIGINT)
00044                         return new NumericSourceRowAttribute(rs,elem);
00045                 else
00046                         return new GenericSourceRowAttribute(rs,elem,colType);
00047 #endif
00048         }
00049
00050
00051   /*
00052         * GENERIC SOURCE ROW ATTRIBUTE
00053                 * Implementation of methods inherited from SourceRowAttribute
00054    */
00055
00056                         void
00057         GenericSourceRowAttribute::setValue( mrdb::ResultSet* rs, int col) {
00058                 assert(rs!=NULL);
00059                 assert(col<=rs->getMetaData()->getColumnCount());
00060
00061                 type = (mrdb::Types::SQLType) rs->getMetaData()->getColumnType(col);
00062                 value = rs->getString(col);
00063         }
00064
00065         void
00066         GenericSourceRowAttribute::setValue( const std::string& inStr ) {
00067
00068                 type = (mrdb::Types::SQLType) 4;
00069                 value = inStr;
00070         }
00071
00072
00073         int
00074         GenericSourceRowAttribute::compareTo(const SourceRowAttribute& rhs) const {
00075                 return value.compare(dynamic_cast<const GenericSourceRowAttribute&>(rhs).value);
00076         }
00077
00078         mrdb::Types::SQLType
00079         GenericSourceRowAttribute::getType() const {
00080                 return type;
00081         }
00082
00083         std::string
00084         GenericSourceRowAttribute::asString(const std::string& sep) const {
00085                 if(value == "")
00086                         return "(not set)";
00087                 else
00088                         return value;
00089         }
00090
00091         void
00092         GenericSourceRowAttribute::serialize(std::ostream& os) const {
00093                 os « " @[ " « value « "@] ";
00094         }
00095
```

```
00096         void
00097         GenericSourceRowAttribute::deserialize(std::istream& is) {
00098     // In order to avoid missing spaces etc. we resort to use istream::get method
00099     // instead of » operator. This means that we need to read one char at a time
00100     // and implement a simple automata in order to parse the regular expression / @[ .* @] /
00101     // Pro: quite general
00102     // Cons: Nobody ensures that the strings @[ and @] does not appear in the value itself
00103     //       When this happens everything break
00104     // Solution: We should escape any '@' character in thestd::string before inserting it
00105     // into the database. (Cons: "The insertion process becomes more costly...")
00106
00107
00108                 typedef enum {
00109                         AS_NORMAL, // normal automa state
00110                                 AS_EXITING // The state in which the automa will be when ? has been
      read as the last char
00111                                         } AutomaState;
00112
00113                         std::string buf;
00114                         char ch;
00115
00116                         assert(is);
00117                         is » buf;
00118                         assert(is);
00119
00120                         if(buf!="@[")
00121                                 throw MineruleException(MR_ERROR_INTERNAL, "Expected `@[', but:
      `"+buf+"' found!");
00122
00123                         is.get(); // throwing away spurious space.
00124
00125                         bool finished = false;
00126                         value = "";
00127
00128                         AutomaState state = AS_NORMAL;
00129                         while(is.good() && !finished) {
00130                                 ch=is.get();
00131
00132                                 if( ch==']' && state==AS_EXITING )
00133                                         finished=true;
00134                                 else {
00135                                         switch( ch ) {
00136                                                 case '@':
00137                                                         state = AS_EXITING;
00138                                                         break;
00139
00140                                                 default:
00141                                                         if(state==AS_EXITING)
00142                                                                 value += '@';
00143
00144                                                         value += ch;
00145                                                         break;
00146                                         }
00147                                 }
00148                         }
00149
00150                         if(!is)
00151                                 throw MineruleException(MR_ERROR_INTERNAL,
00152                                         "Unfinished Generic value, value read 'till now:"+value);
00153                 }
00154
00155                 std::string
00156                 GenericSourceRowAttribute::getSQLData() const {
00157                         return std::string("'")+value+"'";
00158                 }
00159
00160  /* NUMERIC SOURCE ROW ATTRIBUTE */
00161
00162
00163                 void
00164                 NumericSourceRowAttribute::setValue( mrdb::ResultSet* rs, int col) {
00165                         assert(rs!=NULL);
00166                         assert(col<=rs->getMetaData()->getColumnCount());
00167                         assert( getType() == (mrdb::Types::SQLType)
      rs->getMetaData()->getColumnType(col) );
00168
00169                         value = rs->getLong(col);
00170                 }
00171
00172                 void
00173                 NumericSourceRowAttribute::setValue( const std::string& inStr ) {
00174                         value = (long int) strtod(inStr.c_str(), NULL);
00175                 }
00176
00177                 int
00178                 NumericSourceRowAttribute::compareTo(const SourceRowAttribute& rhs) const {
00179                         long int res= value - dynamic_cast<const
```

```
      NumericSourceRowAttribute&>(rhs).value;
00180                         return res;
00181                 }
00182
00183                 std::string
00184                 NumericSourceRowAttribute::asString(const std::string& sep) const {
00185                         std::stringstream ss;
00186                         ss « value;
00187                         return ss.str();
00188                 }
00189
00190                 std::string
00191                 NumericSourceRowAttribute::getSQLData() const {
00192                         std::stringstream ss;
00193                         ss « "'" « value « "'";
00194                         return ss.str();
00195                 }
00196
00197
00202                         void
00203                 NumericSourceRowAttribute::serialize(std::ostream& os) const {
00204                         os « " " « value « " ";
00205                 }
00206
00207                 void NumericSourceRowAttribute::deserialize(std::istream& is) {
00208                         std::string val;
00209                         is » val;
00210                         try {
00211                                 value = Converter(val).toLong();
00212                         } catch (MineruleException& e) {
00213                                 throw MineruleException( MR_ERROR_INTERNAL,
00214                                         "Cannot deserialize the requested NumericSourceRowAttribute,"
00215                                                 "the reason is:" + std::string(e.what()) );
00216                         }
00217                 }
00218
00219         }
```

7.253 /Users/esposito/Software/minerule/src/Database/SourceRow↩ AttributeCollection.cpp File Reference

```
#include <typeinfo>
#include "minerule/Database/SourceRowAttributeCollection.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
```

Namespaces

- namespace minerule

7.254 SourceRowAttributeCollection.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
00016 #include <typeinfo>
00017 #include "minerule/Database/SourceRowAttributeCollection.hpp"
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019
00020 /* ****************************** */
00021 /* SourceRowAttributeCollection methods */
00022 /* ****************************** */
00023
00024 namespace minerule {
00025
00026   // public members
00027
00028   SourceRowAttributeCollection::SourceRowAttributeCollection(
00029                                     mrdb::ResultSetMetaData* rsmd,
00030                                     mrdb::ResultSet* rs,
00031                                     std::vector<int> elems) {
00032     assert(!elems.empty());
00033
00034     std::vector<int>::const_iterator it = elems.begin();
00035     for(; it!=elems.end(); it++) {
00036       SourceRowAttribute* attr = SourceRowAttribute::createAttribute(rsmd,rs,*it);
00037       attributes.push_back(attr);
00038     }
00039   }
00040
00041
00042   // copy constructor
00043   SourceRowAttributeCollection::SourceRowAttributeCollection(
00044                               const SourceRowAttributeCollection& rhs) {
00045     CollectionType::const_iterator it1 = rhs.attributes.begin();
00046
00047     while( it1!=rhs.attributes.end() ) {
00048       attributes.push_back((SourceRowAttribute*)(*it1)->copy());
00049       it1++;
00050     }
00051   }
00052
00053   SourceRowElement&
00054   SourceRowAttributeCollection::operator=(
00055                               const SourceRowElement& _rhs){
00056     try {
00057       const SourceRowAttributeCollection& rhs =
00058         dynamic_cast<const SourceRowAttributeCollection&>(_rhs);
00059
00060       freeCollection();
00061       CollectionType::const_iterator it1 = rhs.attributes.begin();
00062
00063       while( it1!=rhs.attributes.end() ) {
00064         attributes.push_back((SourceRowAttribute*)(*it1)->copy());
00065         it1++;
00066       }
00067     } catch (std::bad_cast& bc) {
00068       MRErr() << "Fatal error (SourceRowAttributeCollection::operator=)" << std::endl;
00069       throw;
00070     }
00071
00072       return *this;
00073   }
00074
00075   void
00076   SourceRowAttributeCollection::freeCollection() {
00077     //  std::cout << "freeCollection..." << std::endl;
00078     CollectionType::const_iterator it1 = attributes.begin();
00079     while( it1!=attributes.end() ) {
00080       delete *it1;
00081       it1++;
00082     }
00083
00084     if(attributes.size()>0)
00085       attributes.clear();
00086   }
00087
00090   std::string
00091   SourceRowAttributeCollection::asString(const std::string& sep) const {
00092     CollectionType::const_iterator it;
00093
00094     std::string attrString;
00095
00096     attrString = "";
00097
00098     //  std::cerr << "in sourcerowattrcollection::c_str collection size:"<< attributes->size() <<
    std::endl;
00099
00100     it=attributes.begin();
00101     if(it!=attributes.end()) {
00102       attrString+=(*it)->asString(sep);
00103       it++;
```

```
00104     }
00105
00106     for(; it!=attributes.end();it++) {
00107       attrString+=sep;
00108       attrString+=(*it)->asString(sep);
00109     }
00110
00111     return attrString;
00112   }
00113
00114
00115
00116   void SourceRowAttributeCollection::setPreparedStatementParameters(mrdb::PreparedStatement* state,
      size_t start_index) const {
00117           size_t count = start_index;
00118           for( CollectionType::const_iterator it = attributes.begin(); it!=attributes.end(); ++it ) {
00119                   (*it)->setPreparedStatementParameters(state, count++);
00120           }
00121   }
00122
00123
00124
00125   std::string
00126   SourceRowAttributeCollection::getSQLData() const {
00127     CollectionType::const_iterator it = attributes.begin();
00128    std::string result = "";
00129
00130     if(it!=attributes.end()) {
00131       result+= (*it)->getSQLData();
00132
00133       it++;
00134     }
00135
00136     for(; it!=attributes.end(); it++ ) {
00137       result+="," + (*it)->getSQLData();
00138     }
00139
00140     return result;
00141   }
00142
00143   std::string SourceRowAttributeCollection::getFullElementType() const {
00144           char chstr[2] = { getElementType(), '\0' };
00145           std::string result( chstr );
00146
00147           CollectionType::const_iterator it = attributes.begin();
00148           for(; it!=attributes.end(); ++it ) {
00149                   // nested attributes collections cannot be created, it suffices
00150                   // to iterate over the attributes and collect types with getElementType
00151                   // (instead of getFullElementType).
00152                   result+=(*it)->getElementType();
00153           }
00154
00155           return result;
00156   }
00157
00158   // operators
00159
00160   bool
00161   SourceRowAttributeCollection::operator==(const SourceRowElement& e2) const {
00162     const SourceRowAttributeCollection& s1 = *this;
00163     const SourceRowAttributeCollection& s2 =
00164       dynamic_cast<const SourceRowAttributeCollection&>(e2);
00165
00166     CollectionType::const_iterator it1 = s1.attributes.begin();
00167     CollectionType::const_iterator it2 = s2.attributes.begin();
00168
00169
00170     for(; it1!=s1.attributes.end() &&
00171           it2!=s2.attributes.end() &&
00172           (**it1)==(**it2); it1++, it2++ );
00173
00174     return (it1==s1.attributes.end() && it2==s2.attributes.end());
00175   }
00176   // Returns true if s1<s2
00177
00178   bool
00179   SourceRowAttributeCollection::operator()(const SourceRowElement& e1,const SourceRowElement& e2)
      const  {
00180     const SourceRowAttributeCollection& s1 =
00181       dynamic_cast<const SourceRowAttributeCollection&>(e1);
00182     const SourceRowAttributeCollection& s2 =
00183       dynamic_cast<const SourceRowAttributeCollection&>(e2);
00184
00185     CollectionType::const_iterator it1 = s1.attributes.begin();
00186     CollectionType::const_iterator it2 = s2.attributes.begin();
00187
00188
```

```
00189     for(; it1!=s1.attributes.end() &&
00190           it2!=s2.attributes.end() &&
00191           (**it1)==(**it2); it1++, it2++ );
00192
00193     // here: it1 and it2 counted the same number of elements from the start of the respective
00194     //   containers, i.e. pos(it1)==pos(it2)==J for a given J.
00195     // moreover: forall i<J s1[i]==s2[i] AND
00196     // (
00197     //  it1==it2 == end() (==> s1 == s2) OR
00198     //  it1==end() && it2!=end() (==> s1<s2)
00199     //  it1!=end() && it2!=end() && it1!=it2 (==> s1<s2 <=> (*it1)->compareTo(**it2)<0) )
00200     // )
00201
00202     if(it1==s1.attributes.end() && it2==s2.attributes.end()) // s1==s2
00203       return false;
00204
00205     if(it1==s1.attributes.end())  // s1<s2
00206       return true;
00207
00208     if(it2==s2.attributes.end()) // s1>s2
00209       return false;
00210
00211
00212     return (**it1)<(**it2);
00213   }
00214
00215
00216   std::ostream& SourceRowAttributeCollection::operator«(std::ostream& os) const {
00217     if(attributes.empty()) {
00218       os « "(attr_collection_empty)";
00219       return os;
00220     }
00221
00222     SourceRowAttributeCollection::CollectionType::const_iterator it;
00223     it = attributes.begin();
00224     os « "attr_collection:(";
00225
00226     os « **it;
00227     it++;
00228
00229     while(it!=attributes.end()) {
00230       os « "," « **it;
00231       it++;
00232     }
00233
00234     os «")";
00235
00236     return os;
00237   }
00238
00239
00240   void
00241   SourceRowAttributeCollection::serialize(std::ostream& os) const  {
00242     SourceRowAttributeCollection::CollectionType::const_iterator it;
00243     for(it=attributes.begin(); it!=attributes.end(); it++) {
00244       os « " " « (*it)->getElementType() « " ";  // more elements to come, specifying type
00245       (*it)->serialize(os);
00246     }
00247
00248     os « " S "; // stop!
00249   }
00250
00251   void
00252   SourceRowAttributeCollection::deserialize(std::istream& is) {
00253    std::string buf;
00254     is » buf;
00255     while(buf!="S" && is) {
00256       if( buf.size()!=1 )
00257         throw MineruleException(MR_ERROR_INTERNAL,
00258                                 "Expecting a single letter element class id, but found:"
00259                                 +buf);
00260
00261       SourceRowAttribute* element =
00262         dynamic_cast<SourceRowAttribute*>(SourceRowElement::createElementFromType(buf[0]));
00263
00264       if(element==NULL)
00265         throw MineruleException(MR_ERROR_INTERNAL,
00266                                 "Cannot create a source row element identified by:" +
00267                                 std::string(buf));
00268       element->deserialize(is);
00269
00270       attributes.push_back(element);
00271       is » buf;
00272     }
00273
00274     if(buf!="S")
00275       throw MineruleException(MR_ERROR_INTERNAL,
```

```
00276                               "Expecting a source row attribute, but the stream ended!");
00277   }
00278
00279
00280 } // namespace minerule
```

7.255 /Users/esposito/Software/minerule/src/Database/SourceRowColumnIds.cpp File Reference

```
#include "minerule/Database/SourceRowColumnIds.hpp"
```

Namespaces

- namespace minerule

7.256 SourceRowColumnIds.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/SourceRowColumnIds.hpp"
00017
00018 namespace minerule {
00019
00020         unsigned int
00021         SourceRowColumnIds::setElems(std::vector<int>& elems, unsigned int start, unsigned int
       numCols) const {
00022                 elems.clear();
00023                 for( unsigned int i=0; i<numCols; i++ ) {
00024                         elems.push_back(start+i);
00025                 }
00026
00027                 return start+numCols-1;
00028         }
00029
00030
00031         unsigned int
00032         SourceRowColumnIds::setGroupElems(unsigned int start,unsigned int numCols) {
00033                 return setElems(groupElems, start, numCols);
00034         }
00035
00036         unsigned int
00037         SourceRowColumnIds::setClusterBodyElems(unsigned int start, unsigned int numCols) {
00038                 return setElems(clusterBodyElems, start, numCols);
00039         }
00040
00041         unsigned int
00042         SourceRowColumnIds::setBodyElems(unsigned int start,unsigned int numCols) {
00043                 return setElems(bodyElems, start, numCols);
00044         }
00045
00046         unsigned int
00047         SourceRowColumnIds::setClusterHeadElems(unsigned int start, unsigned int numCols) {
00048                 return  setElems(clusterHeadElems, start, numCols);
00049         }
00050
00051         unsigned int
00052         SourceRowColumnIds::setHeadElems(unsigned int start, unsigned int numCols) {
00053                 return setElems(headElems, start, numCols);
00054         }
00055 }
```

7.257 /Users/esposito/Software/minerule/src/Database/SourceRowElement.cpp File Reference

```
#include "minerule/Database/SourceRowElement.hpp"
#include "minerule/Database/SourceRowAttribute.hpp"
#include "minerule/Database/SourceRowAttributeCollection.hpp"
```

Namespaces

- namespace minerule

7.258 SourceRowElement.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/SourceRowElement.hpp"
00017 #include "minerule/Database/SourceRowAttribute.hpp"
00018 #include "minerule/Database/SourceRowAttributeCollection.hpp"
00019
00020 namespace minerule {
00021
00022         SourceRowElement*
00023                 SourceRowElement::createElement(
00024                         mrdb::ResultSetMetaData* rsmd,
00025                                 mrdb::ResultSet* rs,
00026         const std::vector<int>& srd) {
00027                 if(srd.empty())
00028                         return NULL;
00029
00030                 if(srd.size()==1) {
00031                         return SourceRowAttribute::createAttribute(rsmd,rs,srd[0]);
00032                 }
00033                 else
00034                         return new SourceRowAttributeCollection(rsmd,rs,srd);
00035         }
00036
00037         SourceRowElement*
00038         SourceRowElement::createElementFromType(const ElementType el) {
00039                 if( SourceRowEmptyElement().getElementType()==el )
00040                         return new SourceRowEmptyElement();
00041
00042                 if( GenericSourceRowAttribute().getElementType()==el )
00043                         return new GenericSourceRowAttribute();
00044
00045                 if( MemDebugGenericSourceRowAttribute().getElementType()==el )
00046                         return new MemDebugGenericSourceRowAttribute();
00047
00048                 if( NumericSourceRowAttribute().getElementType()==el )
00049                         return new NumericSourceRowAttribute();
00050
00051                 if( SourceRowAttributeCollection().getElementType()==el)
00052                         return new SourceRowAttributeCollection();
00053
00054                 throw MineruleException(MR_ERROR_INTERNAL,
00055                         "SourceRowElement::createElementFromType: "
00056                                 "Cannot recognize the specified element type!");
00057         }
```

```
00058
00059         void SourceRowElement::serializeElementToString(const SourceRowElement& elem, std::string&
      strRepr) {
00060                 std::ostringstream sstream;
00061                 sstream « elem.getElementType();
00062                 elem.serialize(sstream);
00063                 strRepr = sstream.str();
00064         }
00065
00066         SourceRowElement*
00067         SourceRowElement::deserializeElementFromResultSet(mrdb::ResultSet* rs, size_t start_index)  {
00068                 mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00069                 size_t numColumns = rsmd->getColumnCount();
00070                 std::vector<int> descr;
00071
00072                 for(size_t i=start_index; i<=numColumns; ++i)
00073                         descr.push_back(i);
00074
00075                 SourceRowElement* elem = SourceRowElement::createElement(rsmd, rs, descr);
00076                 return elem;
00077         }
00078
00079
00080         SourceRowElement*
00081         SourceRowElement::deserializeElementFromString(const std::string& strRepr)  {
00082                 std::istringstream sstream(strRepr);
00083                 assert(sstream);
00084                 ElementType elType;
00085                 sstream»elType;
00086                 assert(sstream);
00087
00088                 SourceRowElement* elem = createElementFromType(elType);
00089                 elem->deserialize(sstream);
00090                 return elem;
00091         }
00092
00093 } // nameapace
```

7.259 /Users/esposito/Software/minerule/src/Database/SourceRowMeta↩ Info.cpp File Reference

```
#include <iostream>
#include <string.h>
#include "minerule/Database/SourceRowMetaInfo.hpp"
#include "minerule/Database/SourceRowColumnIds.hpp"
#include "minerule/mrdb/Types.hpp"
```

Data Structures

- class AttributesUtil

7.260 SourceRowMetaInfo.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
```

```
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <string.h>
00018
00019 #include "minerule/Database/SourceRowMetaInfo.hpp"
00020 #include "minerule/Database/SourceRowColumnIds.hpp"
00021 #include "minerule/mrdb/Types.hpp"
00022
00023
00024 using namespace minerule;
00025
00026
00027
00028 /* *********************************
00029  * SourceRowMetaInfo methods
00030  * *********************************/
00031
00032 void
00033 SourceRowAttrCollectionDescriptor::
00034 setColumnNames(mrdb::ResultSet* rs, const std::vector<int>& collectionElems) {
00035   mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00036
00037   std::vector<int>::const_iterator it = collectionElems.begin();
00038   columnNames = "";
00039
00040   if(it!=collectionElems.end()) {
00041     columnNames += rsmd->getColumnName(*it);
00042     it++;
00043   }
00044
00045   for(; it!=collectionElems.end(); it++ ) {
00046     columnNames += "," + rsmd->getColumnName(*it);
00047   }
00048 }
00049
00050 std::string
00051 SourceRowAttrCollectionDescriptor::dataDefinitionForElem(mrdb::ResultSet* rs, int elem) {
00052   mrdb::ResultSetMetaData* rsmd = rs->getMetaData();
00053
00054   std::string result = rsmd->getColumnName(elem) + " " + rsmd->getColumnTypeName(elem);
00055
00056   MRDebug() << result << std::endl;
00057
00058
00059   return result;
00060 }
00061
00062 std::string SourceRowAttrCollectionDescriptor::questionMarks() const {
00063         std::string result;
00064         for( size_t i=0; i<columnsCount; ++i) {
00065                 if( i!=0 )
00066                         result += ",?";
00067                 else
00068                         result += "?";
00069         }
00070
00071         return result;
00072 }
00073
00074 void
00075 SourceRowAttrCollectionDescriptor::setDataDefinition(mrdb::ResultSet* rs, const std::vector<int>&
      collectionElems) {
00076   std::vector<int>::const_iterator it = collectionElems.begin();
00077   dataDefinition = "";
00078
00079   if(it!=collectionElems.end()) {
00080     dataDefinition += dataDefinitionForElem(rs, *it);
00081     it++;
00082   }
00083
00084   for(; it!=collectionElems.end(); it++ ) {
00085     dataDefinition += "," + dataDefinitionForElem(rs,*it);
00086   }
00087 }
00088
00089 void SourceRowAttrCollectionDescriptor::init(mrdb::ResultSet* rs, const std::vector<int>&
      collectionElems ) {
00090     setColumnNames(rs,collectionElems);
00091     setDataDefinition(rs,collectionElems);
00092     columnsCount = collectionElems.size();
00093 }
00094
00095
00096
00097 SourceRowAttrCollectionDescriptor::SourceRowAttrCollectionDescriptor(mrdb::ResultSet* rs, const
```

```
      std::vector<int>& collectionElems) {
00098         init(rs, collectionElems);
00099 }
00100
00101 const std::string&
00102 SourceRowAttrCollectionDescriptor::getSQLDataDefinition() const {
00103   return dataDefinition;
00104 }
00105
00106 const std::string&
00107 SourceRowAttrCollectionDescriptor::getSQLColumnNames() const {
00108   return columnNames;
00109 }
00110
00111
00112 /* *********************************
00113  * SourceRowMetaInfo methods
00114  * *********************************/
00115
00116 SourceRowMetaInfo::SourceRowMetaInfo(mrdb::ResultSet* rs, const SourceRowColumnIds& rowDes)
00117          : group(rs,rowDes.groupElems),
00118            clusterBody(rs,rowDes.clusterBodyElems),
00119            body(rs,rowDes.bodyElems),
00120            clusterHead(rs,rowDes.clusterHeadElems),
00121            head(rs,rowDes.headElems) { }
00122
00123
00124  class AttributesUtil {
00125          int curr_attr_pos;
00126  public:
00127          AttributesUtil() : curr_attr_pos(1) {}
00128
00129          std::vector<int> generatePositions(const ParsedMinerule::AttrVector& attrs) {
00130                  std::vector<int> result;
00131                  for( ParsedMinerule::AttrVector::const_iterator it = attrs.begin(); it!=attrs.end();
      ++it ) {
00132                          result.push_back(curr_attr_pos++);
00133                  }
00134                  return result;
00135          }
00136
00137          static std::string names_to_string(const ParsedMinerule::AttrVector& attrs) {
00138                  std::string result;
00139                  for( ParsedMinerule::AttrVector::const_iterator it = attrs.begin(); it!=attrs.end();
      ++it ) {
00140                          if( it != attrs.begin() ) result += ",";
00141                          result += *it;
00142                  }
00143
00144                  return result;
00145          }
00146  };
00147
00148  SourceRowMetaInfo::SourceRowMetaInfo(mrdb::Connection* mrdb_connection, const ParsedMinerule&
      minerule) {
00149          ParsedMinerule::AttrVector attr_list;
00150          attr_list.insert(attr_list.end(), minerule.ga.begin(), minerule.ga.end());
00151          attr_list.insert(attr_list.end(), minerule.ca.begin(), minerule.ca.end());
00152      attr_list.insert(attr_list.end(), minerule.ba.begin(), minerule.ba.end());
00153      attr_list.insert(attr_list.end(), minerule.ca.begin(), minerule.ca.end());
00154          attr_list.insert(attr_list.end(), minerule.ha.begin(), minerule.ha.end());
00155
00156          std::string query =
00157                  "SELECT " + AttributesUtil::names_to_string(attr_list) +
00158                  " FROM " + minerule.tab_source + " LIMIT 1";
00159
00160          mrdb::Statement* state = mrdb_connection->createStatement();
00161          mrdb::ResultSet* rs = state->executeQuery(query);
00162
00163          AttributesUtil attrUtils;
00164          group.init(rs, attrUtils.generatePositions(minerule.ga) );
00165          clusterBody.init(rs, attrUtils.generatePositions(minerule.ca) );
00166          body.init(rs, attrUtils.generatePositions(minerule.ba));
00167          clusterHead.init(rs, attrUtils.generatePositions(minerule.ca));
00168          head.init(rs,attrUtils.generatePositions(minerule.ha));
00169
00170          delete rs;
00171          delete state;
00172  }
```

7.261 /Users/esposito/Software/minerule/src/Database/SourceTable.cpp File Reference

```
#include "minerule/Database/SourceTable.hpp"
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/mrdb/ResultSet.hpp"
```

Namespaces

- namespace minerule

7.262 SourceTable.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Database/SourceTable.hpp"
00017 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00018 #include "minerule/mrdb/ResultSet.hpp"
00019
00020 namespace minerule {
00021
00022         SourceTable::~SourceTable() {
00023                 for(std::vector<mrdb::ResultSet*>::const_iterator it = _managedResults.begin();
       it!=_managedResults.end(); ++it) {
00024                         delete *it;
00025                 }
00026         }
00027
00028         bool SourceTable::Iterator::next() {
00029                 ++_rowCounter;
00030                 assert( _sourceRow != NULL && _resultSet != NULL );
00031
00032                 if( _resultSet->next() ) {
00033                         _sourceRow->init(_resultSet, _columnIds);
00034                         return true;
00035                 } else {
00036                         return false;
00037                 }
00038         }
00039
00040         bool SourceTable::Iterator::isAfterLast() const {
00041                 assert( _resultSet!=NULL );
00042                 return _resultSet->isAfterLast();
00043         }
00044
00045         SourceTable::Iterator SourceTable::newIterator(SourceTable::IteratorKind kind)  {
00046                 switch(kind) {
00047                         case BodyIterator:
00048                                 {
00049                                         _managedResults.push_back(_bodyStatement->executeQuery());
00050                                         SourceRowColumnIds bodyCols = _columnIds;
00051                                         bodyCols.headElems.clear();
00052
00053                                         return Iterator(_managedResults.back(), bodyCols);
00054                                 }
00055                         case HeadIterator:
00056                                 {
```

```
00057                                           _managedResults.push_back(_headStatement->executeQuery());
00058                                           SourceRowColumnIds headCols = _columnIds;
00059                                           headCols.bodyElems.clear();
00060 
00061                                           return Iterator(_managedResults.back(), headCols);
00062                                   }
00063                           case FullIterator:
00064                                   {
00065                                           _managedResults.push_back(_fullStatement->executeQuery());
00066                                           return Iterator(_managedResults.back(), _columnIds);
00067                                   }
00068                   }
00069 
00070                   throw MineruleException(MR_ERROR_INTERNAL, "Uknown IteratorKind. This is a bug, please
      report it!");
00071           }
00072 
00073           size_t SourceTable::getTotGroups() {
00074                   return _pdu.evaluateTotGroups();
00075           }
00076 
00077           void SourceTable::initBodyHeadResultSets() {
00078                   mrdb::Connection* connection =
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00079 
00080                   std::string bodyCondition, headCondition;
00081 
00082                   list_AND_node* miningCondition = (_minerule.mc!=NULL ? _minerule.mc->l_and : NULL);
00083                   HeadBodyPredicatesSeparator::separate(miningCondition, bodyCondition, headCondition);
00084 
00085                   std::string bodyQuery = _pdu.buildBodyTableQuery(_columnIds, bodyCondition);
00086                   std::string headQuery = _pdu.buildHeadTableQuery(_columnIds, headCondition);
00087 
00088                   _bodyStatement = connection->prepareStatement( bodyQuery );
00089                   _headStatement = connection->prepareStatement( headQuery );
00090 
00091                   MRLog("Body Query:"+bodyQuery);
00092                   MRLog("Head Query:"+headQuery);
00093           }
00094 
00095           void SourceTable::initFullResultSet() {
00096                   _usesCrossProduct = true;
00097                   mrdb::Connection* connection =
      MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00098 
00099                   std::string query = _pdu.buildExtendedSourceTableQuery(_columnIds);
00100 
00101                   MRLog("Source Table Query:"+query);
00102                   _fullStatement = connection->prepareStatement( query );
00103           }
00104 
00105 
00106           void SourceTable::init() {
00107                   if( _sourceTableRequirements.crossProduct() )
00108                           initFullResultSet();
00109                   else
00110                           initBodyHeadResultSets();
00111           }
00112 
00113 } // namespace
```

7.263 /Users/esposito/Software/minerule/src/Optimizer/CatalogueInstaller.cpp File Reference

```
#include "minerule/Optimizer/CatalogueInstaller.hpp"
#include "minerule/Optimizer/Installers/MySqlCatalogueInstaller.hpp"
#include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
```

Namespaces

- namespace minerule

7.264 CatalogueInstaller.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Optimizer/CatalogueInstaller.hpp"
00017 #include "minerule/Optimizer/Installers/MySqlCatalogueInstaller.hpp"
00018 #include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
00019 #include "minerule/Utils/MineruleOptions.hpp"
00020
00021 namespace minerule {
00022
00023         CatalogueInstaller::CatalogueInstaller() : _connection(
       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection() ), _statement(
       _connection->createStatement() ) {}
00024
00025         CatalogueInstaller* CatalogueInstaller::newInstaller(SupportedDbms dbms) {
00026                 switch(dbms) {
00027                 case MySql:
00028                         return new MySqlCatalogueInstaller();
00029                 case Postgres:
00030                         return new PostgresCatalogueInstaller();
00031                 }
00032         }
00033
00034         CatalogueInstaller* CatalogueInstaller::newInstaller() {
00035                 std::string dbmsStr = MineruleOptions::getSharedOptions().getMRDB().getDBMS();
00036
00037                 if(dbmsStr == "mysql") return newInstaller(MySql);
00038                 if(dbmsStr == "postgres") return newInstaller(Postgres);
00039
00040                 throw MineruleException(MR_ERROR_OPTION_CONFIGURATION,std::string("Option mrdb::dbms
       is set to an unsupported value.") +
00041                         " Current value is:"+dbmsStr+ ". Supported values are 'postgres' and
       'mysql'.");
00042         }
00043
00044
00045 }
```

7.265 /Users/esposito/Software/minerule/src/Optimizer/GAQuery↩ Combinator.cpp File Reference

```
#include "minerule/Optimizer/GAQueryCombinator.hpp"
#include "minerule/PredicateUtils/PredicateUtils.hpp"
#include <iterator>
#include <cmath>
#include <algorithm>
```

Namespaces

- namespace minerule

Macros

- #define bit_vector std::vector<bool>

Variables

- const int minerule::HAM_LOSS = 10
- const int minerule::DISJ_LOSS = 1
- const int minerule::QUERY_LOSS = 1

7.265.1 Macro Definition Documentation

7.265.1.1 bit_vector

```
#define bit_vector std::vector<bool>
```

Definition at line 23 of file GAQueryCombinator.cpp.

7.266 GAQueryCombinator.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Optimizer/GAQueryCombinator.hpp"
00017 #include "minerule/PredicateUtils/PredicateUtils.hpp"
00018 #include <iterator>
00019 #include <cmath>
00020 #include <algorithm>
00021
00022 #ifndef bit_vector
00023         #define bit_vector std::vector<bool>
00024 #endif
00025
00026 namespace minerule {
00027
00028         const int HAM_LOSS   = 10;
00029         const int DISJ_LOSS  = 1;
00030         const int QUERY_LOSS = 1;
00031
00032         void GAQueryCombinator::buildAssociatedPredicate(const GAGenome& g, Predicate& p,
00033                                 size_t& numUsedDisjuncts, size_t& numUsedQueries, size_t&
       numAssertedBits) const {
00034                 size_t last_used_disj = 1<<10;
00035
00036                 const GA1DBinaryStringGenome& genome = dynamic_cast<const GA1DBinaryStringGenome&>(g);
00037                 bit_vector usedQueries(mr_candidates.size());
00038
00039                 for(size_t i=0; i<maxDisjuncts; i++) {
00040                         Predicate innerPred;
00041
00042                         for(size_t j=0; j<mr_candidates.size(); j++) {
00043                                 assert(i*mr_candidates.size()+j<size_t(genome.length()));
00044
00045                                 int QjAsserted = genome.gene(i*mr_candidates.size()+j);
00046
00047                                 if(QjAsserted) {
00048                                         numAssertedBits++;
00049                                         usedQueries[j]=1;
```

```
00050                                           if(last_used_disj!=i) {
00051                                                   last_used_disj=i;
00052                                                   numUsedDisjuncts++;
00053                                           }
00054
00055                                           if(innerPred.empty()) {
00056                                                   innerPred |= mr_candidates[j];
00057                                           } else {
00058                                                   innerPred &= mr_candidates[j];
00059                                           }
00060                                   }
00061                           }
00062
00063                           p |= innerPred;
00064                   }
00065
00066                   for(size_t i=0; i<usedQueries.size(); i++) {
00067                           numUsedQueries+=usedQueries[i];
00068                   }
00069
00070           }
00071
00072
00073
00074           float GAQueryCombinator::evaluator(GAGenome& g) {
00075                   GAQueryCombinator* qc = (GAQueryCombinator*) g.userData();
00076                   assert(qc!=NULL);
00077
00078                   if((clock()-qc->startingTime)/CLOCKS_PER_SEC > qc->timeOut )
00079                           throw TimeOutException();
00080
00081                   size_t numUsedDisjuncts=0;
00082                   size_t numUsedQueries=0;
00083                   size_t numAssertedBits=0;
00084
00085                   Predicate p2;
00086                   qc->buildAssociatedPredicate(g,p2, numUsedDisjuncts, numUsedQueries, numAssertedBits);
00087
00088                   if(numUsedDisjuncts > qc->maxDisjuncts ||       numUsedQueries > qc->maxQueries)
00089                                   return 0;
00090
00091                   Predicate& p1=qc->mr_target;
00092
00093       // The following two lines assign unique variables id
00094       // to elements in p1,p2. It do so setting the variable id
00095       // for each element of the union (through ci) to the position
00096       // in which the predicate appear in the union itself. Note
00097       // that this work only because two equivalent predicates are
00098       // indeed two references to the same object (look at SimplePredicate
00099       // class to understand why this is true).
00100                   size_t counter=0;
00101                   list_AND_node* allPreds=NULL;
00102
00103                   CountingIterator ci(counter,allPreds);
00104                   std::set<SimplePredicate*, PtrSimplePredComp>& sp1 = p1.getPredicateList();
00105                   std::set<SimplePredicate*, PtrSimplePredComp>& sp2 = p2.getPredicateList();
00106
00107                   std::set_union( sp1.begin(), sp1.end(),
00108                           sp2.begin(), sp2.end(), ci );
00109
00110
00111                   p1.setNumVariables(counter);
00112                   p2.setNumVariables(counter);
00113
00114                   InvalidConfigurationFilter filter( qc->tab_source, allPreds );
00115
00116                   ExpressionNFCoder e1(filter);
00117                   ExpressionNFCoder e2(filter);
00118
00119                   float ham_distance=EncodedNF::computeHammingDistance(e1.encode(p1),e2.encode(p2));
00120
00121                   CountingIterator::delete_list_AND_node(allPreds);
00122
00123                   if( ham_distance==0 )
00124                           return HUGE_VAL;
00125
00126                   return ( (1 << qc->maxDistinctPredicates)* HAM_LOSS
00127                           + qc->mr_candidates.size()*qc->maxDisjuncts - ham_distance*HAM_LOSS -
      numAssertedBits);
00128
00129           }
00130
00131
00132           GABoolean
00133           GAQueryCombinator::TerminationCriterion(GAGeneticAlgorithm & ga) {
00134                   if(ga.statistics().current( GAStatistics::Maximum )==HUGE_VAL)
00135                           return gaTrue;
```

```
00136                 else
00137                         return (ga.generation() < ga.nGenerations() ? gaFalse : gaTrue);
00138         }
00139
00140
00141
00142         void GAQueryCombinator::evolve() {
00143     // create a genome
00144                 GA1DBinaryStringGenome genome(mr_candidates.size()*maxDisjuncts,
      GAQueryCombinator::evaluator, this);
00145     // create the genetic algorithm
00146                 GASimpleGA ga(genome);
00147                 ga.populationSize(10);
00148                 ga.nGenerations(100);
00149                 ga.pCrossover(0.8);
00150                 ga.pMutation(0.05);
00151                 ga.terminator(TerminationCriterion);
00152     // do the evolution
00153
00154                 startingTime = clock();
00155                 ga.evolve();
00156
00157                 buildResult(ga);
00158         }
00159
00160
00161         void GAQueryCombinator::buildResult(const GAGeneticAlgorithm& ga) {
00162                 result.clear();
00163
00164                 if(ga.statistics().current(GAStatistics::Maximum)!=HUGE_VAL)
00165                         return;
00166
00167                 size_t conjSize = mr_candidates.size();
00168
00169                 const GA1DBinaryStringGenome& solution = dynamic_cast<const
      GA1DBinaryStringGenome&>(ga.population().best());
00170
00171                 for(size_t i=0; i<maxDisjuncts; i++) {
00172                         QueryAndList currAnd;
00173
00174                         for(size_t j=0; j<conjSize; j++) {
00175                                 if( solution.gene(i*conjSize+j) )
00176                                         currAnd.push_back(j);
00177                         }
00178
00179                         if(!currAnd.empty())
00180                                 result.push_back(currAnd);
00181                 }
00182         }
00183
00184         const GAQueryCombinator::QueryOrList& GAQueryCombinator::getResult() const {
00185                 return result;
00186         }
00187
00188 }
00189
00190
00191
00192
00193
00194
```

7.267 /Users/esposito/Software/minerule/src/Optimizer/Installers/↩ PostgresCatalogueInstaller.cpp File Reference

```
#include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
#include "minerule/mrdb/Statement.hpp"
```

Namespaces

- namespace minerule

7.268 PostgresCatalogueInstaller.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Optimizer/Installers/PostgresCatalogueInstaller.hpp"
00017 #include "minerule/mrdb/Statement.hpp"
00018
00019 namespace minerule {
00020         void PostgresCatalogueInstaller::dropMRQuery()                    {
00021                 _statement->execute("DROP TABLE IF EXISTS mr_query");
00022         }
00023
00024         void PostgresCatalogueInstaller::dropMRAttList()                {
00025                 _statement->execute("DROP TABLE IF EXISTS mr_att_lists");
00026         }
00027
00028         void PostgresCatalogueInstaller::dropMREqKeys()                 {
00029                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys");
00030         }
00031
00032         void PostgresCatalogueInstaller::dropMREqKeysCol()           {
00033                 _statement->execute("DROP TABLE IF EXISTS mr_eq_keys_col");
00034         }
00035
00036         void PostgresCatalogueInstaller::dropMRDepFun()                 {
00037                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun");
00038         }
00039
00040         void PostgresCatalogueInstaller::dropMRDepFunCol()           {
00041                 _statement->execute("DROP TABLE IF EXISTS mr_dep_fun_col");
00042         }
00043
00044         void PostgresCatalogueInstaller::dropMRAutoincrement()    {
00045                 _statement->execute("DROP TABLE IF EXISTS mr_autoincrement");
00046         }
00047
00048
00049
00050         void PostgresCatalogueInstaller::installMRQuery() {
00051                 _statement->execute("CREATE TABLE mr_query (query_id  integer NOT NULL PRIMARY KEY,"
00052                                                         "query_text  text NOT NULL,"
00053                                                         "query_name  varchar(128) NOT NULL,     "
00054                                                         "tab_results_name varchar(128) NOT NULL,"
00055                                                         "source_tab_name varchar(128) NOT NULL,"
00056                                                         "gal  integer NOT NULL,"
00057                                                         "ral  integer NOT NULL,"
00058                                                                 "cal  integer)");
00059                 _statement->execute("CREATE INDEX mr_query_tab_results_name_index ON
      mr_query(tab_results_name)");
00060                 _statement->execute("CREATE INDEX mr_query_query_name_index ON mr_query
      (query_name)");
00061                 _statement->execute("CREATE INDEX mr_query_source_tab_name_index ON mr_query
      (source_tab_name)");
00062         }
00063
00064
00065
00066         void PostgresCatalogueInstaller::installMRAttList() {
00067                 _statement->execute("CREATE TABLE mr_att_lists (att_list_id  integer NOT NULL,col_name
      varchar(128) NOT NULL)");
00068                 _statement->execute("CREATE INDEX mr_att_lists_att_list_id_index ON
      mr_att_lists(att_list_id)");
00069         }
00070
00071
00072
00073         void PostgresCatalogueInstaller::installMREqKeys() {
00074                 _statement->execute("CREATE TABLE mr_eq_keys (tab_name  varchar(128) NOT NULL,"
00075                                                                 "key_id  integer NOT NULL,"
00076                                                                 "ref_key_id  integer NOT NULL,"
00077                                                                 "order_type char(1),"
```

```
00078                                                   "CONSTRAINT mr_eq_keys_primary PRIMARY
      KEY(tab_name,key_id,ref_key_id))");
00079         }
00080
00081
00082
00083         void PostgresCatalogueInstaller::installMREqKeysCol() {
00084                 _statement->execute("CREATE TABLE mr_eq_keys_col (key_id  integer NOT NULL,"
00085                                                         "col_name  varchar(128) NOT NULL)");
00086                 _statement->execute("CREATE INDEX mr_eq_keys_col_key_id_index ON
      mr_eq_keys_col(key_id)");
00087         }
00088
00089
00090
00091         void PostgresCatalogueInstaller::installMRDepFun() {
00092                 _statement->execute("CREATE TABLE mr_dep_fun (lhs_tab_name  varchar(128) NOT NULL,"
00093                                                         "lhs_att_list_id  integer NOT NULL,"
00094                                                         "rhs_tab_name  varchar(128) NOT NULL,"
00095                                                         "rhs_att_list_id  integer NOT NULL,"
00096                                                         "order_type char(1),"
00097                                                         "CONSTRAINT mr_dep_fun_primary PRIMARY
      KEY(lhs_tab_name,lhs_att_list_id,rhs_tab_name, rhs_att_list_id))");
00098         }
00099
00100
00101
00102         void PostgresCatalogueInstaller::installMRDepFunCol() {
00103                 _statement->execute("CREATE TABLE mr_dep_fun_col (col_id  integer NOT NULL, col_name
      varchar(128) NOT NULL)");
00104                 _statement->execute("CREATE INDEX mr_dep_fun_col_col_id_index ON
      mr_dep_fun_col(col_id)");
00105         }
00106
00107
00108
00109         void PostgresCatalogueInstaller::installMRAutoincrement() {
00110                 _statement->execute("CREATE TABLE mr_autoincrement (table_name varchar(128) NOT NULL,"
00111                                                         "current_id integer NOT NULL,"
00112                                                         "CONSTRAINT mr_autoincrement_primary PRIMARY
      KEY (table_name))");
00113
00114         }
00115
00116
00117
00118         void PostgresCatalogueInstaller::initializeAutoincrement() {
00119                 _statement->execute("INSERT INTO mr_autoincrement (table_name, current_id) VALUES
      ('mr_att_lists',0),('mr_eq_keys',0),('mr_query',0)");
00120         }
00121
00122
00123
00124
00125
00126 }
```

7.269 /Users/esposito/Software/minerule/src/Optimizer/Optimized↩ Minerule.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>
#include <cstdlib>
#include <list>
#include <algorithm>
#include "minerule/Optimizer/OptimizedMinerule.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Statement.hpp"
```

```
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Optimizer/QueryNormalizer.hpp"
#include "minerule/PredicateUtils/PredicateUtils.hpp"
```

Namespaces

- namespace minerule

7.270 OptimizedMinerule.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include<stdio.h>
00017 #include<stdlib.h>
00018 #include<string>
00019 #include<iostream>
00020 #include<sstream>
00021 #include<fstream>
00022 #include<cstdlib>
00023 #include<list>
00024 #include<algorithm>
00025
00026 /*
00027   #ifndef String
00028   #  include<wxstrgnu.h>
00029 typedef wxString String;
00030   #endif
00031 */
00032
00033 #include"minerule/Optimizer/OptimizedMinerule.hpp"
00034 #include"minerule/Parsers/ParsedMinerule.hpp"
00035
00036 #include "minerule/mrdb/Connection.hpp"
00037 #include "minerule/mrdb/ResultSet.hpp"
00038 #include "minerule/mrdb/Statement.hpp"
00039 #include "minerule/mrdb/ResultSetMetaData.hpp"
00040
00041 #include "minerule/Utils/MineruleOptions.hpp"
00042 #include "minerule/Optimizer/QueryNormalizer.hpp"
00043 #include "minerule/PredicateUtils/PredicateUtils.hpp"
00044
00045
00046 namespace minerule {
00047
00048         void OptimizedMinerule::attributesInPredicate( const list_OR_node* l, std::set<std::string>&
      result ) {
00049                 while( l!=NULL ) {
00050                         list_AND_node* l_and = l->l_and;
00051                         while( l_and!=NULL ) {
00052                                 if( !Converter(l_and->sp->val1).isNumber() )
00053                                         result.insert(l_and->sp->val1);
00054
00055                                 if( !Converter(l_and->sp->val2).isNumber() )
00056                                         result.insert(l_and->sp->val2);
00057
00058                                 l_and=l_and->next;
00059                         }
00060
00061                         l=l->next;
```

```
00062                 }
00063         }
00064
00065         bool OptimizedMinerule::predicateAttributesAreIncluded( const ParsedMinerule& mr1,
00066         const ParsedMinerule& mr2 ) {
00067                 std::set<std::string> mr1Attrs;
00068                 std::set<std::string> mr2Attrs;
00069
00070                 attributesInPredicate( mr1.mc, mr1Attrs );
00071                 attributesInPredicate( mr2.mc, mr2Attrs );
00072
00073                 return includes(mr2Attrs.begin(), mr2Attrs.end(), mr1Attrs.begin(), mr1Attrs.end());
00074
00075         }
00076
00077         // mr1: candidate query
00078         // mr2: target query
00079         bool OptimizedMinerule::isACandidateQuery( const ParsedMinerule& mr1, const ParsedMinerule&
      mr2  ) {
00080                 return mr1.ga==mr2.ga &&
00081                         mr1.ca==mr2.ca &&
00082                         mr1.ra==mr2.ra &&
00083                         mr1.ba==mr2.ba &&
00084                         mr1.ha==mr2.ha &&
00085                         mr1.c_aggr_list == mr2.c_aggr_list &&
00086                         mr1.sup<=mr2.sup &&
00087                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00088                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00089                         mr1.tab_source==mr2.tab_source &&
00090               // above are the standard checking
00091               // if in addition both the queries are item dependent (mr1 is
00092               // item dependent by hypothesis).
00093                         OptimizerCatalogue::hasIDConstraints(mr1) &&
00094           // and the all the attributes on which mr1.mc predicates are included
00095           // in those on which mr2 predicates
00096                         predicateAttributesAreIncluded(mr1,mr2);
00097                   // then mr1 is a candidate query
00098         }
00099
00100         void OptimizedMinerule::checkForCombinedQueries(  ) {
00101                 MRLogPush("Searching for equivalences due to query combinations...");
00102                 std::vector<Predicate> candidates;
00103
00104                 std::vector<ParsedMinerule>::const_iterator it;
00105
00106     // disabling the logging of the Parser since it would confuse
00107     // the user.
00108                 MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00109
00110                 std::ostream& log = MRLog() « "Candidates are: ";
00111
00112                 for(it=optInfo.minerulesToCombine.begin(); it!=optInfo.minerulesToCombine.end(); it++)
      {
00113                         if(it!=optInfo.minerulesToCombine.begin()) log«", ";
00114
00115                         log « it->tab_result;
00116                         candidates.push_back(Predicate(it->mc));
00117                 }
00118                 log « std::endl;
00119
00120     // re-enabling the log
00121                 MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00122
00123                 Predicate target(minerule.mc);
00124
00125         // FIXME: The third parameters ("Sales") seems to be obsoleted and not used any more
00126                 GAQueryCombinator qCombinator(target, candidates);
00127                 MineruleOptions::Optimizations::Combinator& combopt =
00128                         MineruleOptions::getSharedOptions().getOptimizations().getCombinator();
00129
00130                 qCombinator.setMaxDisjuncts(combopt.getMaxDisjuncts());
00131                 qCombinator.setMaxQueries(combopt.getMaxQueries());
00132                 qCombinator.setMaxDistinctPredicates(combopt.getMaxDistinctPredicates());
00133                 qCombinator.setTimeOutThreshold(combopt.getTimeOutThreshold());
00134
00135                 try {
00136                         qCombinator.evolve();
00137                 } catch( TimeOutException& e ) {
00138                         MRLog() « "The algorithm timed out." « std::endl;
00139                 }
00140
00141
00142         // FIXME: all of the following is about output. It should be moved into an appropriate method
00143                 if( qCombinator.getResult().empty() ) {
00144                         MRLog() « "No valid combination found!" « std::endl;
00145                 } else {
00146                         std::string qryString;
```

```
00147
00148                         GAQueryCombinator::QueryOrList::const_iterator itOr;
00149                         for(itOr=qCombinator.getResult().begin(); itOr!=qCombinator.getResult().end();
       itOr++) {
00150                                 if(itOr!=qCombinator.getResult().begin())
00151                                         qryString += " OR ";
00152
00153                                 GAQueryCombinator::QueryAndList::const_iterator itAnd;
00154                                 for(itAnd=itOr->begin(); itAnd!=itOr->end(); itAnd++) {
00155                                         if( itAnd!=itOr->begin() )
00156                                                 qryString += " AND ";
00157
00158                                         qryString += optInfo.minerulesToCombine[*itAnd].tab_result;
00159                                 }
00160                         }
00161
00162
00163                         MRLog()     « "A combination of queries equivalent to the given one"
00164                                 « " has been found. The found query follows:" « std::endl;
00165                         MRLog()     « qryString « std::endl;
00166
00167                         optInfo.combinationFormula = qCombinator.getResult();
00168                         optInfo.relationship = Combination;
00169                 }
00170
00171                 MRLogPop();
00172         }
00173
00174
00175         void OptimizedMinerule::optimize()
00176         {
00177                 if(minerule.miningTask!=MTMineRules) {
00178                         MRLog() « "Skipping optimization, " « miningTaskToString(minerule.miningTask)
       « " task is still unsupported by the optimizer" « std::endl;
00179                         MRDebug() « "Skipping optimization, " «
       miningTaskToString(minerule.miningTask) « " task is still unsupported by the optimizer" « std::endl;
00180                         return;
00181                 }
00182
00183                 std::list<ParsedMinerule> lPreviousMr;
00184                 if(!MineruleOptions::getSharedOptions().getOptimizations().getTryOptimizations())
00185                         return;
00186
00187                 bool avoidCombinationDetection =
00188
       MineruleOptions::getSharedOptions().getOptimizations().getAvoidCombinationDetection()
00189                                 || !hasIDConstraints();
00190
00191                 OptimizerCatalogue& catalogue =
00192                         MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00193
00194                 QueryNormalizer normalizer( catalogue, minerule );
00195                 normalizer.normalize();
00196                 optInfo.minerule = minerule;
00197                 optInfo.relationship = None;
00198                 MRDebug("Minerule text after normalization:" + minerule.getText());
00199
00200                 std::vector<CatalogueInfo> catInfos;
00201                 OptimizerCatalogue::getMRQueryInfos(catInfos);
00202
00203     //lettura delle minerules precedenti
00204                 try     {
00205                         MRLogPusher _("Determining if a relationship exists with previous
       minerules...");
00206
00207                         MRLog() « "Reading past minerules" « std::endl;
00208                         for(std::vector<CatalogueInfo>::const_iterator it=catInfos.begin();
       it!=catInfos.end(); ++it) {
00209                                 ParsedMinerule mr;
00210                                 mr.init(it->qryText);
00211                                 MRDebug("Optimize: trying minerule:" + mr.getText());
00212
00213                                 if( !avoidCombinationDetection && isACandidateQuery(mr,minerule) ) {
00214                                         optInfo.minerulesToCombine.push_back(mr);
00215                                 }
00216
00217                                 MineruleRelationship currentRel =
       getMineruleRelationship(mr,minerule);
00218
00219                                 if( currentRel==Equivalence &&
       MineruleOptions::getSharedOptions().getOptimizations().getAvoidEquivalenceDetection() )
00220                                                 currentRel=Dominance;
00221
00222                                 switch( currentRel ) {
00223                                         case Equivalence:
00224                                                 MRLog() « "A past minerule found which includes the
       current one!" « std::endl;
```

```
00225                                                   MRLog() « "Name of the found minerule:" « it->qryName
      « std::endl;
00226                                                   MRLog() « "(The algorithm will store this information
      in the catalogue" « std::endl;
00227                                                   MRLog() « " and exit)" « std::endl;
00228                                                   optInfo.relationship = Equivalence;
00229                                                   optInfo.minerule = mr;
00230                                                   break;
00231                                           case Dominance:
00232                                                   {
00233                                                           MRLog() « "Minerule named `" « mr.tab_result
      «"" dominates the current one." «std::endl;
00234                                                   if(optInfo.relationship==None) {
00235                           // since relationship==None, there is not yet any information
00236                           // about other Dominances. We simply store the found minerule
00237                                                           optInfo.minerule = mr;
00238                                                           MRLog() « "Minerule " « "`" «
      mr.tab_result «"" is now the best promising"
00239                                                                   « " dominant minerule." «
      std::endl;
00240                                                   } else {
00241                                                           // we must keep the current minerule
      only if its result set
00242                                                           // is smaller than the one of the past
      Dominant query found
00243                                                           CatalogueInfo oldQryInfo;
00244                                                           OptimizerCatalogue::getMRQueryInfo(
      optInfo.minerule.tab_result, oldQryInfo );
00245                                                           if( it->resSize<oldQryInfo.resSize) {
00246                                                                   optInfo.minerule = mr;
00247                                                                   MRLog() « "Minerule " « "`" «
      mr.tab_result «"" is now the best promising"
00248                                                                                   « " dominant
      minerule." « std::endl;
00249                                                           } else {
00250                                                                   MRLog() « "Keeping the
      previously found dominant minerule" « std::endl;
00251                                                           }
00252                                                   }
00253
00254                                                   optInfo.relationship = Dominance;
00255                                           }
00256                                           break;
00257                                   default:
00258                                           // nothing to do
00259                                           break;
00260                                   };
00261
00262                             // if the found relationship is Equivalence, then we cannot do better
00263                             // and exit immediately from the loop, otherwise we check if we can
00264                             // find some minerule which is a better candidate for the incremental
00265                             // algorithms.
00266                                   if(optInfo.relationship==Equivalence)
00267                                           break;
00268                           }
00269
00270                           if(optInfo.relationship==None)
00271                                   MRLog() « "No interesting past minerules found." « std::endl;
00272
00273                           if(optInfo.relationship==Dominance)
00274                                   MRLog() « "The best promising dominant minerule found is `"
00275                                           « optInfo.minerule.tab_result « "`" « std::endl;
00276                   } catch (mrdb::SQLException& e) {
00277                           MRErr() « e.what() « std::endl;
00278                           throw;
00279                   }
00280
00281                   if(optInfo.relationship==None && !avoidCombinationDetection) {
00282                           checkForCombinedQueries();
00283                           if(optInfo.relationship==None)
00284                                   optInfo.minerulesToCombine.clear();
00285                   }
00286           }
00287
00288
00289
00290
00291           bool OptimizedMinerule::firstMineruleIncludesSecondMinerule(const ParsedMinerule& mr1,const
      ParsedMinerule& mr2) {
00292                   Predicate mr1pmc(mr1.mc);
00293                   Predicate mr2pmc(mr2.mc);
00294                   Predicate mr1pgc(mr1.gc);
00295                   Predicate mr2pgc(mr2.gc);
00296                   Predicate mr1pcc(mr1.cc);
00297                   Predicate mr2pcc(mr2.cc);
00298
00299                   return
```

```
00300                         mr1.ga==mr2.ga &&
00301                         mr1.ca==mr2.ca &&
00302                         mr1.ra==mr2.ra &&
00303                         mr1.ba==mr2.ba &&
00304                         mr1.ha==mr2.ha &&
00305                         mr1.c_aggr_list == mr2.c_aggr_list &&
00306                         mr1.sup<=mr2.sup &&
00307                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00308                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00309                         mr1.tab_source==mr2.tab_source &&
00310                         PredicateUtils::predicatesAreEquivalent( mr1pmc, mr2pmc, mr1.tab_source ) &&
00311                         PredicateUtils::predicatesAreEquivalent( mr1pgc, mr2pgc, mr1.tab_source ) &&
00312                         PredicateUtils::predicatesAreEquivalent( mr1pcc, mr2pcc, mr1.tab_source );
00313         }
00314
00315
00316         bool OptimizedMinerule::firstMineruleDominatesSecondMinerule(const ParsedMinerule& mr1,const
       ParsedMinerule& mr2) {
00317
00318
00319                 Predicate mr1pmc(mr1.mc);
00320                 Predicate mr2pmc(mr2.mc);
00321                 Predicate mr1pgc(mr1.gc);
00322                 Predicate mr2pgc(mr2.gc);
00323                 Predicate mr1pcc(mr1.cc);
00324                 Predicate mr2pcc(mr2.cc);
00325
00326                 PredicateUtils::PredicateRelationship mcrel =
00327                         PredicateUtils::getPredicateRelationship( mr1pmc, mr2pmc, mr1.tab_source );
00328                 PredicateUtils::PredicateRelationship gcrel =
00329                         PredicateUtils::getPredicateRelationship( mr1pgc, mr2pgc, mr1.tab_source );
00330                 PredicateUtils::PredicateRelationship ccrel =
00331                         PredicateUtils::getPredicateRelationship( mr1pcc, mr2pcc, mr1.tab_source );
00332
00333
00334                 return
00335                         mr1.ga==mr2.ga &&
00336                         mr1.ca==mr2.ca &&
00337                         mr1.ra==mr2.ra &&
00338                         mr1.ba==mr2.ba &&
00339                         mr1.ha==mr2.ha &&
00340                         mr1.c_aggr_list == mr2.c_aggr_list &&
00341                         mr1.sup<=mr2.sup &&
00342                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00343                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00344                         mr1.tab_source==mr2.tab_source &&
00345                         (mcrel==EncodedNF::FirstMoreGeneral || mcrel==EncodedNF::Equivalent)&&
00346                         (gcrel==EncodedNF::FirstMoreGeneral || gcrel==EncodedNF::Equivalent)&&
00347                         (ccrel==EncodedNF::FirstMoreGeneral || ccrel==EncodedNF::Equivalent);
00348         }
00349
00350         OptimizedMinerule::MineruleRelationship
00351         OptimizedMinerule::getMineruleRelationship(const ParsedMinerule& mr1, const ParsedMinerule&
       mr2) {
00352                 if(
       MineruleOptions::getSharedOptions().getOptimizations().getAvoidDominanceDetection() )
00353                         return None;
00354
00355                 if(!(mr1.ga==mr2.ga &&
00356                         mr1.ca==mr2.ca &&
00357                         mr1.ra==mr2.ra &&
00358                         mr1.ba==mr2.ba &&
00359                         mr1.ha==mr2.ha &&
00360                         mr1.c_aggr_list == mr2.c_aggr_list &&
00361                         mr1.sup<=mr2.sup &&
00362                         mr1.bodyCardinalities.contains(mr2.bodyCardinalities) &&
00363                         mr1.headCardinalities.contains(mr2.headCardinalities) &&
00364                         mr1.tab_source==mr2.tab_source))
00365                                 return None;
00366
00367                 Predicate mr1pmc(mr1.mc);
00368                 Predicate mr2pmc(mr2.mc);
00369                 Predicate mr1pgc(mr1.gc);
00370                 Predicate mr2pgc(mr2.gc);
00371                 Predicate mr1pcc(mr1.cc);
00372                 Predicate mr2pcc(mr2.cc);
00373
00374                 PredicateUtils::PredicateRelationship relations[3];
00375
00376                 relations[0] = PredicateUtils::getPredicateRelationship( mr1pmc, mr2pmc,
       mr1.tab_source );
00377                 relations[1] = PredicateUtils::getPredicateRelationship( mr1pgc, mr2pgc,
       mr1.tab_source );
00378                 relations[2] = PredicateUtils::getPredicateRelationship( mr1pcc, mr2pcc,
       mr1.tab_source );
00379
00380                 MineruleRelationship current = Equivalence;
```

```
00381
00382                 for(size_t i=0; i<3; i++) {
00383                         switch( relations[i] ) {
00384                                 case EncodedNF::FirstMoreGeneral:
00385                                         current=Dominance;
00386                                         break;
00387                                 case EncodedNF::Equivalent:
00388                                         // Nothing to do, just avoid to select the default
00389                                         // label
00390                                         break;
00391                                 default:
00392                                         return None;
00393                         };
00394                 }
00395
00396                 return current;
00397         }
00398
00399
00400         bool OptimizedMinerule::hasIDConstraints() const {
00401                 return OptimizerCatalogue::hasIDConstraints(minerule);
00402         }
00403
00404
00405 } // end namespace minerule
```

7.271 /Users/esposito/Software/minerule/src/Optimizer/OptimizerCatalogue.cpp File Reference

```
#include <string.h>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Utils/SQLUtils.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Database/Connection.hpp"
#include <memory>
#include <iterator>
#include <algorithm>
#include <iomanip>
```

Namespaces

- namespace minerule

7.272 OptimizerCatalogue.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
00016 #include <string.h>
00017 #include "minerule/mrdb/ResultSet.hpp"
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019 #include "minerule/Utils/SQLUtils.hpp"
00020 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00021 #include "minerule/Parsers/ParsedMinerule.hpp"
00022 #include "minerule/Database/Connection.hpp"
00023 #include <memory>
00024 #include <iterator>
00025 #include <algorithm>
00026 #include <iomanip>
00027
00028 #include <memory>
00029
00030 namespace minerule {
00031
00032 bool OptimizerCatalogue::existsMinerule(const std::string &mrname) {
00033   mrdb::Connection *conn =
00034       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00035
00036   std::string query = (std::string) "SELECT * FROM mr_query WHERE query_name=" +
00037                       SQLUtils::quote(mrname);
00038
00039   std::auto_ptr<mrdb::Statement> stat(conn->createStatement());
00040   std::auto_ptr<mrdb::ResultSet> rs(stat->executeQuery(query));
00041
00042   return rs->next();
00043 }
00044
00045 void OptimizerCatalogue::deleteMinerule(const std::string &mrname) {
00046   std::vector<std::string> qryNames;
00047
00048   MRLogPusher _("Deleting past minerules...");
00049
00050   Connection conn;
00051   conn.useMRDBConnection(
00052       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection());
00053   conn.setOutTableName(mrname);
00054
00055   std::string query = (std::string) "SELECT query_id "
00056                                     "FROM mr_query "
00057                                     "WHERE query_name=" +
00058                       SQLUtils::quote(mrname) + " OR tab_results_name=" +
00059                       SQLUtils::quote(mrname);
00060
00061   std::auto_ptr<mrdb::Statement> stat(
00062       conn.getMRDBConnection()->createStatement());
00063   mrdb::ResultSet *rs = stat->executeQuery(query);
00064
00065   while (rs->next()) {
00066     MRDebug("Qry to delete:" + rs->getString(1));
00067     qryNames.push_back(rs->getString(1));
00068   }
00069
00070   delete rs;
00071
00072   MRLog() << "I found " << qryNames.size()
00073           << " minerules that need to be deleted" << std::endl;
00074
00075   assert(qryNames.size() > 0);
00076   if (qryNames.size() > 1 &&
00077       !MineruleOptions::getSharedOptions()
00078            .getSafety()
00079            .getAllowCascadeDeletes()) {
00080     throw MineruleException(MR_ERROR_SAFETY_PROBLEM,
00081                             "Requested the delete of"
00082                             " a minerule having dependent minerules, but option"
00083                             " safety::allowsCascadeDeletes is not set");
00084   }
00085
00086   std::vector<std::string>::const_iterator it;
00087   for (it = qryNames.begin(); it != qryNames.end(); it++) {
00088     MRLog() << "Deleting query: '" << *it << "'" << std::endl;
00089
00090     std::string delQry =
00091         "DELETE FROM mr_query WHERE query_id=" + SQLUtils::quote(*it);
00092
00093     try {
00094       stat->execute(delQry);
00095     } catch (mrdb::SQLException& e) {
00096       throw MineruleException(MR_ERROR_INTERNAL, "Cannot delete query " + *it + " reason:" +
    e.what());
00097     }
00098   }
00099
00100   MRLog() << "Dropping tables connected to query named:" << mrname << std::endl;
00101   conn.deleteDestTables();
```

```
00102 }
00103
00104 void OptimizerCatalogue::Catalogue::insertMapping(
00105     const std::string &table, const KeyCols &cols, int refKey,
00106     OrderType orderType) {
00107   mrdb::Connection *connection =
00108       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00109
00110   try {
00111     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00112
00113     std::string getKeyQuery =
00114         std::string("SELECT " + getSchemaInfo("tab_cols_col_name") + " " +
00115                     "FROM " + getSchemaInfo("tab_cols") + " " + "WHERE " +
00116                     getSchemaInfo("tab_cols_key_id") + "=") +
00117         Converter((long)refKey).toString();
00118
00119     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getKeyQuery));
00120     KeyCols refKeyCols;
00121
00122     while (rs->next()) {
00123       refKeyCols.insert(rs->getString(1));
00124     }
00125
00126     //      OrderType orderType = getOrderType(origKey,refKey);
00127     // (*this)[table][cols] = pair<KeyCols,OrderType>(refKeyCols,orderType);
00128     std::pair< KeyCols, std::pair<KeyCols, OrderType> > insElem(
00129         cols, std::pair<KeyCols, OrderType>(refKeyCols, orderType));
00130     (*this)[table].insert(insElem);
00131
00132   } catch (mrdb::SQLException &e) {
00133     throw MineruleException(
00134         MR_ERROR_DATABASE_ERROR,
00135         std::string("Cannot access the db while "
00136                     "initializing a member of the OptimizerCatalogue,"
00137                     "the reason is:") +
00138             e.what());
00139   }
00140 }
00141
00142 OptimizerCatalogue::OrderType
00143 OptimizerCatalogue::Catalogue::stringToOrder(const std::string &str) {
00144   if (str == "r")
00145     return Reversed;
00146
00147   if (str == "e")
00148     return Equal;
00149
00150   return None;
00151 }
00152
00153 void OptimizerCatalogue::Catalogue::initialize() {
00154   mrdb::Connection *connection =
00155       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00156
00157   try {
00158     std::auto_ptr<mrdb::Statement> stat1(connection->createStatement());
00159     std::string getCatalogueQuery =
00160         (std::string) "SELECT A." + getSchemaInfo("tab_main_tab_name") + ",B." +
00161         getSchemaInfo("tab_cols_col_name") + ",A." +
00162         getSchemaInfo("tab_main_rhs_key_id") + ",A." +
00163         getSchemaInfo("tab_main_order_type") + " " + "FROM " +
00164         getSchemaInfo("tab_main") + " AS A," + getSchemaInfo("tab_cols") +
00165         " AS B " + "WHERE A." + getSchemaInfo("tab_main_lhs_key_id") + "=B." +
00166         getSchemaInfo("tab_cols_key_id");
00167
00168     std::auto_ptr<mrdb::ResultSet> rs(stat1->executeQuery(getCatalogueQuery));
00169     KeyCols origKey;
00170     unsigned long refKeyId = 0;
00171     std::string curTable;
00172     OrderType orderType;
00173
00174     // We start building the first of the two col l lists.
00175     // we accumulate the result in the set origKey.
00176
00177     if (rs->next()) {
00178       curTable = rs->getString(1);
00179       origKey.insert(rs->getString(2));
00180       refKeyId = rs->getInt(3);
00181       orderType = stringToOrder(rs->getString(4));
00182     } else {
00183       return;
00184     }
00185
00186     while (rs->next()) {
00187       if (curTable != rs->getString(1) ||
00188           refKeyId != (unsigned long)rs->getInt(3)) {
```

```
00189         // here the current rule changed, in fact either the table
00190         // or the refKeyId are different
00191         insertMapping(curTable, origKey, refKeyId, orderType);
00192         origKey.clear();
00193
00194         curTable = rs->getString(1);
00195         origKey.insert(rs->getString(2));
00196         refKeyId = rs->getInt(3);
00197         orderType = stringToOrder(rs->getString(4));
00198       } else {
00199         // here we continue to accumulate the columns in origKey
00200         origKey.insert(rs->getString(2));
00201       }
00202     } // end while
00203
00204     assert(!origKey.empty());
00205     insertMapping(curTable, origKey, refKeyId, orderType);
00206   } catch (mrdb::SQLException &e) {
00207     throw MineruleException(
00208         MR_ERROR_DATABASE_ERROR,
00209         std::string(
00210             "Cannot access the db while "
00211             "initializing a member of the OptimizerCatalogue, the reason is:") +
00212             e.what());
00213   }
00214 }
00215
00216 std::string OptimizerCatalogue::getNewAutoincrementValue(
00217     const std::string &tableName) {
00218   mrdb::Connection *connection =
00219       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00220
00221   std::string idstr;
00222
00223   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00224   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(
00225       "SELECT current_id FROM mr_autoincrement WHERE table_name=" +
00226       SQLUtils::quote(tableName)));
00227
00228   if (!rs->next())
00229     throw MineruleException(
00230         MR_ERROR_INSTALLATION_PROBLEM,
00231         "Cannot find the autoincrement value for table " + tableName +
00232             " in Optimizer catalogue (i.e. in table:`mr_autoincrement'),"
00233             " please check your installation");
00234
00235   size_t id = rs->getInt(1);
00236   idstr = Converter((long)++id).toString();
00237
00238   std::auto_ptr<mrdb::Statement> updateStatement(connection->createStatement());
00239   updateStatement->execute("UPDATE mr_autoincrement SET current_id=" + idstr +
00240                            " WHERE table_name=" + SQLUtils::quote(tableName));
00241
00242   return idstr;
00243 }
00244
00245 std::string OptimizerCatalogue::addMineruleAttributeList(
00246     const ParsedMinerule::AttrVector &l) {
00247   if (l.empty())
00248     return "NULL";
00249
00250   std::string id = getNewAutoincrementValue("mr_att_lists");
00251   mrdb::Connection *connection =
00252       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00253
00254   std::string query =
00255       "INSERT INTO mr_att_lists (att_list_id, col_name) VALUES ";
00256
00257   ParsedMinerule::AttrVector::const_iterator it;
00258   for (it = l.begin(); it != l.end(); it++) {
00259     if (it != l.begin())
00260       query += ",";
00261
00262     query += "(" + SQLUtils::quote(id) + "," + SQLUtils::quote(*it) + ")";
00263   }
00264
00265   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00266   statement->execute(query);
00267
00268   return id;
00269 }
00270
00271 bool OptimizerCatalogue::checkInstallation() {
00272   mrdb::Connection *connection =
00273       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00274   mrdb::DatabaseMetaData *meta = connection->getMetaData();
00275   mrdb::ResultSet *rs = meta->getTables("mr_%");
```

```
00276
00277   typedef std::pair<std::string, bool> TableInfo;
00278   std::vector<TableInfo> tables;
00279   tables.push_back(TableInfo("mr_query", false));
00280   tables.push_back(TableInfo("mr_att_lists", false));
00281   tables.push_back(TableInfo("mr_eq_keys", false));
00282   tables.push_back(TableInfo("mr_eq_keys_col", false));
00283   tables.push_back(TableInfo("mr_dep_fun", false));
00284   tables.push_back(TableInfo("mr_dep_fun_col", false));
00285   tables.push_back(TableInfo("mr_autoincrement", false));
00286
00287   rs->next();
00288   while (!rs->isAfterLast()) {
00289     std::string currentTable = rs->getString(1);
00290     std::vector<TableInfo>::iterator it =
00291         find(tables.begin(), tables.end(),
00292              std::pair<std::string, bool>(currentTable, false));
00293
00294     if (it != tables.end()) {
00295       MRLog() « "Table: " « std::left « std::setw(50) « currentTable
00296               « StringUtils::toGreen(" OK") « std::endl;
00297       it->second = true;
00298     }
00299     rs->next();
00300   }
00301
00302   delete rs;
00303
00304   bool allTablePresents = true;
00305   for (std::vector<TableInfo>::iterator it = tables.begin(); it != tables.end();
00306        ++it) {
00307     allTablePresents &= it->second;
00308     if (!it->second) {
00309       MRLog() « "Table: " « std::left « std::setw(50) « it->first
00310               « StringUtils::toRed(" MISSING") « std::endl;
00311     }
00312   }
00313
00314   return allTablePresents;
00315 }
00316
00317 void OptimizerCatalogue::install() {
00318   CatalogueInstaller *installer = CatalogueInstaller::newInstaller();
00319   installer->install();
00320   delete installer;
00321 }
00322
00323 void OptimizerCatalogue::uninstall() {
00324   CatalogueInstaller *installer = CatalogueInstaller::newInstaller();
00325   installer->uninstall();
00326   delete installer;
00327 }
00328
00329 void OptimizerCatalogue::addMineruleResult(
00330     const OptimizerCatalogue::MineruleResultInfo &
00331         mri) {
00332
00333   mrdb::Connection *connection =
00334       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00335
00336   std::string galid = addMineruleAttributeList(mri.ga);
00337   std::string ralid = addMineruleAttributeList(mri.ra);
00338   std::string calid = addMineruleAttributeList(mri.ca);
00339   std::string qryid = getNewAutoincrementValue("mr_query");
00340   // std::cout«"PIPPO"«std::endl;
00341   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00342   // std::cout«"PLUTO"«std::endl;
00343   statement->execute(
00344       std::string("INSERT INTO mr_query ") +
00345       "(query_id, query_text, query_name, tab_results_name, source_tab_name," +
00346       "gal,ral,cal) VALUES (" + qryid + "," + SQLUtils::quote(mri.getText()) +
00347       "," + SQLUtils::quote(mri.tab_result) + "," +
00348       SQLUtils::quote(mri.resultset) + "," + SQLUtils::quote(mri.tab_source) +
00349       "," + galid + "," + ralid + "," + calid + ")");
00350 }
00351
00352 void OptimizerCatalogue::addDerivedResult(
00353     const std::string &original,
00354     const std::string derived) {
00355   minerule::CatalogueInfo originalInfo;
00356   bool derivedQueryAlreadyPresent = false;
00357
00358   try {
00359     minerule::OptimizerCatalogue::getMRQueryInfo(original, originalInfo, false);
00360   } catch (minerule::MineruleException &e) {
00361     throw MineruleException(
00362         MR_ERROR_CATALOGUE_ERROR,
```

```
00363         std::string(
00364             "Cannot retrieve the original minerule from the catalogue") +
00365             "The reason is:" + e.what());
00366   }
00367
00368   try {
00369     // checking the derived results do not already exist.
00370     minerule::CatalogueInfo derivedInfo;
00371     minerule::OptimizerCatalogue::getMRQueryInfo(derived, derivedInfo, false);
00372     derivedQueryAlreadyPresent = true;
00373   } catch (minerule::MineruleException &e) {
00374     // do nothing, we are "rooting" for this
00375   }
00376
00377   if (derivedQueryAlreadyPresent) {
00378     throw MineruleException(
00379         MR_ERROR_CATALOGUE_ERROR,
00380         std::string("The derived query name already exists in the catalogue. "
00381                     "Bailing out") +
00382             " so to avoid possible overwriting of useful results.");
00383   }
00384
00385   minerule::ParsedMinerule mr(originalInfo.qryText);
00386   mr.tab_result = derived;
00387   minerule::OptimizerCatalogue::addMineruleResult(
00388       minerule::OptimizerCatalogue::MineruleResultInfo(mr));
00389 }
00390
00391 std::string OptimizerCatalogue::getResultsetName(
00392     const std::string &queryname) {
00393   mrdb::Connection *connection =
00394       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00395
00396   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00397   std::auto_ptr<mrdb::ResultSet> rs(
00398       statement->executeQuery("SELECT tab_results_name "
00399                               "FROM mr_query "
00400                               "WHERE query_name=" +
00401                               SQLUtils::quote(queryname)));
00402
00403   if (!rs->next())
00404     throw MineruleException(
00405         MR_ERROR_CATALOGUE_ERROR,
00406         "Cannot find the query named:" + queryname +
00407             " "
00408             "either there is some problem with the optimizer catalogue "
00409             "or a wrong query name has been specified");
00410
00411   return rs->getString(1);
00412 }
00413
00414 std::string
00415 OptimizerCatalogue::getMRQueryName(size_t i) {
00416   //  -- This implementation is much more efficient but does not ensure that the
00417   //  result retrieved
00418   //             are in the same order of the ones returned by getMRQueryInfos
00419   //
00420   // std::string queryNumber = Converter(i-1).toString();
00421   // mrdb::Connection* connection =
00422   // MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00423   //
00424   // std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00425   // std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery("SELECT
00426   // query_name FROM mr_query LIMIT 1 OFFSET "+queryNumber));
00427   //
00428   // if( rs->next() && !rs->isAfterLast() ) {
00429   //    return rs->getString(1);
00430   // } else {
00431   //    throw MineruleException(MR_ERROR_CATALOGUE_ERROR, "Query number
00432   // "+Converter(i).toString()+" not found");
00433   // }
00434   //  -- NEW IMPLEMENTATION
00435
00436   std::vector<CatalogueInfo> catInfoVec;
00437   getMRQueryInfos(catInfoVec);
00438   if (i - 1 < catInfoVec.size()) {
00439     return catInfoVec[i - 1].resName;
00440   } else {
00441     throw MineruleException(MR_ERROR_CATALOGUE_ERROR,
00442                             "Query number " + Converter(i).toString() +
00443                                 " not found");
00444   }
00445 }
00446
00447 void OptimizerCatalogue::getMRQueryNames(
00448     std::vector<std::string> &nameVec) {
00449   mrdb::Connection *connection =
```

```
00450       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00451
00452   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00453   std::auto_ptr<mrdb::ResultSet> rs(
00454       statement->executeQuery("SELECT query_name FROM mr_query"));
00455
00456   while (rs->next()) {
00457     nameVec.push_back(rs->getString(1));
00458   }
00459 }
00460
00461 void OptimizerCatalogue::setMRQueryInfoFromResultSet(mrdb::ResultSet *rs,
00462                                                      CatalogueInfo &info,
00463                                                      bool includeResultSize) {
00464   Connection connection;
00465
00466   info.qryName = rs->getString(1);
00467   info.qryText = rs->getString(2);
00468   info.resName = rs->getString(3);
00469   connection.setOutTableName(info.resName);
00470
00471   info.resTables.push_back(connection.getTableName(Connection::RulesTable));
00472   info.resTables.push_back(connection.getTableName(Connection::BodiesTable));
00473   info.resTables.push_back(connection.getTableName(Connection::HeadsTable));
00474
00475   if (includeResultSize)
00476     info.updateQrySize();
00477 }
00478
00479 void OptimizerCatalogue::getMRQueryInfos(
00480     std::vector<CatalogueInfo> &catInfoVec,
00481     bool includeResultSize) {
00482   mrdb::Connection *connection =
00483       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00484
00485   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00486   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(
00487       "SELECT query_name,query_text, tab_results_name, source_tab_name "
00488       "FROM mr_query"));
00489
00490   while (rs->next()) {
00491     CatalogueInfo info;
00492     setMRQueryInfoFromResultSet(rs.get(), info, includeResultSize);
00493
00494     catInfoVec.push_back(info);
00495   }
00496 }
00497
00498 void OptimizerCatalogue::getMRQueryInfo(
00499     const std::string &qryName, CatalogueInfo &info,
00500     bool includeResultSize) {
00501   mrdb::Connection *mrdb_connection =
00502       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00503
00504   std::auto_ptr<mrdb::Statement> statement(mrdb_connection->createStatement());
00505   std::auto_ptr<mrdb::ResultSet> rs(
00506       statement->executeQuery("SELECT query_name,query_text, "
00507                               "  tab_results_name, source_tab_name "
00508                               "FROM mr_query "
00509                               "WHERE query_name='" +
00510                               qryName + "'"));
00511
00512   if (rs->next()) {
00513     setMRQueryInfoFromResultSet(rs.get(), info, includeResultSize);
00514   } else {
00515     throw MineruleException(
00516         MR_ERROR_CATALOGUE_ERROR,
00517         "Cannot find the query named:" + qryName +
00518             " "
00519             "Either there is some problem with the optimizer catalogue "
00520             "or a wrong query name has been specified");
00521   }
00522 }
00523
00524 void OptimizerCatalogue::getMRQueryResultIterator(
00525     const std::string &qryName, QueryResult::Iterator &it, double supp,
00526     double conf) {
00527   CatalogueInfo catInfo;
00528   getMRQueryInfo(qryName, catInfo);
00529
00530   ParsedMinerule p;
00531   MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00532   p.init(catInfo.qryText);
00533   MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00534   if (supp < 0)
00535     supp = p.sup;
00536   if (conf < 0)
```

```
00537     conf = p.conf;
00538
00539   it.init(catInfo.resName, supp, conf);
00540 }
00541
00542 void CatalogueInfo::updateQrySize() {
00543   ParsedMinerule p;
00544   MineruleOptions::getSharedOptions().getLogStreamObj().disable();
00545   p.init(qryText);
00546   MineruleOptions::getSharedOptions().getLogStreamObj().enable();
00547
00548   std::string qry = "SELECT count(*) "
00549                     "FROM " +
00550                     resName + " " + "WHERE supp>=" +
00551                     Converter(p.sup).toString() + " AND con>=" +
00552                     Converter(p.conf).toString();
00553
00554   mrdb::Connection *connection =
00555       MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00556   std::auto_ptr<mrdb::Statement> statement(connection->createStatement());
00557   std::auto_ptr<mrdb::ResultSet> rs(statement->executeQuery(qry));
00558
00559   MRDebug() « "updateQrySize, the query is:" « qry « std::endl;
00560
00561   if (rs->next()) {
00562     resSize = rs->getInt(1);
00563   } else {
00564     throw MineruleException(MR_ERROR_CATALOGUE_ERROR,
00565                             "CatalogueInfo::updateQrySize - cannot"
00566                             " retrieve the size of the result set");
00567   }
00568 }
00569
00570 bool OptimizerCatalogue::isIDAttribute(
00571     const std::string &table, const ParsedMinerule::AttrVector &itemCols,
00572     const std::string &attribute) const {
00573   std::set<std::string> itemColsSet;
00574   copy(itemCols.begin(), itemCols.end(),
00575        std::insert_iterator< std::set<std::string> >(itemColsSet,
00576                                                      itemColsSet.begin()));
00577
00578   if (itemColsSet.find(attribute) != itemColsSet.end()) {
00579     // if I am checking over the attributes that composes the attrlist then
00580     // the attribute is trivially item dependent.
00581     return true;
00582   }
00583
00584   Catalogue::const_iterator cat_it = depFunCatalogue.find(table);
00585   if (cat_it == depFunCatalogue.end())
00586     return false;
00587
00588   const CatalogueEntry &ce = cat_it->second;
00589   CatalogueEntry::const_iterator ce_it;
00590   for (ce_it = ce.begin(); ce_it != ce.end(); ce_it++) {
00591     // if I can find the attribute in the rhs of the fd
00592     // and the item cols includes the lhs of the fd
00593     if (ce_it->second.first.find(attribute) != ce_it->second.first.end() &&
00594         includes(itemColsSet.begin(), itemColsSet.end(), ce_it->first.begin(),
00595                  ce_it->first.end()))
00596       // then the attribute is item dependent
00597       return true;
00598   }
00599
00600   return false;
00601 }
00602
00610 bool OptimizerCatalogue::hasIDConstraints(const ParsedMinerule &minerule)  {
00611   OptimizerCatalogue &optcat =
00612       MineruleOptions::getSharedOptions().getOptimizations().getCatalogue();
00613
00614   list_OR_node *it_or = minerule.mc;
00615   for (; it_or != NULL; it_or = it_or->next) {
00616     list_AND_node *it_and = it_or->l_and;
00617     for (; it_and != NULL; it_and = it_and->next) {
00618       bool attr1 = SQLUtils::isAttribute(it_and->sp->val1);
00619       bool attr2 = SQLUtils::isAttribute(it_and->sp->val2);
00620       const ParsedMinerule::AttrVector *attrList;
00621       std::string theAttr;
00622
00623       if (attr1) {
00624         theAttr = it_and->sp->val1;
00625       }
00626
00627       if (attr2) {
00628         theAttr = it_and->sp->val2;
00629       }
00630
```

```
00631       if (attr1 && attr2)
00632         return false;
00633
00634       if (theAttr.substr(0, 5) == "BODY.")
00635         attrList = &minerule.ba;
00636       else if (theAttr.substr(0, 5) == "HEAD.")
00637         attrList = &minerule.ha;
00638       else
00639         throw MineruleException(MR_ERROR_MINERULETEXT_PARSING,
00640                                 "Found a condition defined over an attribute,"
00641                                 " but neither HEAD nor BODY selector has been"
00642                                 " specified.");
00643
00644       SQLUtils::removeHeadBodyFromAttrName(theAttr);
00645
00646       if ((attr1 || attr2) &&
00647           !optcat.isIDAttribute(minerule.tab_source, *attrList, theAttr))
00648         return false;
00649     }
00650   }
00651
00652   return true;
00653 }
00654
00655 } // namespace
```

7.273 /Users/esposito/Software/minerule/src/Optimizer/QueryNormalizer.cpp File Reference

```
#include <string.h>
#include <algorithm>
#include <memory>
#include "minerule/Optimizer/QueryNormalizer.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include "minerule/PredicateUtils/SimplePredAnalyzer.hpp"
```

Namespaces

- namespace minerule

Macros

- #define HEAD(a) ((a).first)
- #define TAIL(a) ((a).second.first)

Functions

- void minerule::removeNode (list_AND_node **l)
- void minerule::removeAllNodes (list_AND_node *&l)

7.273.1 Macro Definition Documentation

7.273.1.1 HEAD

```
#define HEAD(
              a ) ((a).first)
```

7.273.1.2 TAIL

```
#define TAIL(
              a ) ((a).second.first)
```

7.274 QueryNormalizer.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <string.h>
00017 #include <algorithm>
00018 #include <memory>
00019 #include "minerule/Optimizer/QueryNormalizer.hpp"
00020 #include "minerule/Utils/Converter.hpp"
00021 #include "minerule/Utils/MineruleOptions.hpp"
00022 #include "minerule/mrdb/ResultSet.hpp"
00023 #include "minerule/mrdb/ResultSetMetaData.hpp"
00024 #include "minerule/PredicateUtils/SimplePredAnalyzer.hpp"
00025
00026 namespace minerule {
00027
00028     /*
00029      * *** PREDICATE NORMALIZATION SECTION ***
00030      * NOTE: The system now implements improvements !I1! and !I2!
00031      *       which are described below.
00032      *
00033      * (Note: numbers between '*' (e.g., *1*,*2*...), are marks used in
00034      * the implementation in order to specify which part of the
00035      * description any function implemetns).
00036      *
00037      * The following functions normalizes a predicate. Recall that the
00038      * predicate is in disjunctive normal form.  The normalization is
00039      * obtained by changing the condition in such a way that it
00040      * maintains the same semantics, but it uses the reference key
00041      * columns instead of non-reference ones.
00042      *
00043      * Let us consider a candidate substitution Ai->ai (meaning that we
00044      * have to substitute the set Ai of columns with the single column
00045      * ai) and a single disjunct having the form C0 && C1 && ... Cn
00046      * (each Cj is a predicate in the form a op b (where "a" and "b" are
00047      * attributes or values)).
00048      *
00049      * *4* The substitution is possible iff:
00050      *
00051      * [1] all elements of Ai appears in the conditions C0 ... Cn, let
00052      *   us call the predicates in which the elements Ai appears
00053      *   as G0..Gm
00054      *
00055      * [2] there are not a condition C= (a op b) in C0..Cn such that:
00056      *    * a and b are both attributes AND
00057      *    * at least one among a,b belongs to Ai
00058      *
```

```
00059      * [3] the operators are consistent with the substitution, more
00060      * precisely: let us call o1...om the operators in G0...Gm.
00061      * The substitution is possible
00062      * if the following conditions holds:
00063      *   * o1=o2..=on AND
00064      *   * if they are all <, or all >, the substitution is
00065      *     possible only if the order of the substitution is either EQUAL
00066      *     or REVERSED (as specified in the catalogue).
00067      *
00068      * *3* When the above conditions are met, the substitution is performed
00069      * as follows:
00070      *
00071      *    [1] If G0==(X0=x0), G1==(X1=x1)..Gn==(Xm=xm): let us call r a
00072      *    row in the source table such r(X0)=x0...r(Xm)=xm, the
00073      *    substitution will substitute to G0&G1&..Gn with the predicate
00074      *    ai=r(ai) (note that since Ai is included in {X0,..,Xm} and
00075      *    Ai->ai holds, then this predicate will be satisfied whenever
00076      *    G0&G1..&Gm is satisified).If a row r satisfying the given
00077      *    condition does not exist, then the substitution will be "1=0",
00078      *    which holds whenever G0&G1..&Gn hold (i.e.: never).
00079      *
00080      *    [2] if G0==(X0<x0), G1==(X1<x1),...,Gm==(Xm<xm):
00081      *    let us consider the set of R in the source table such that
00082      *      r(X0)<x0, r(X1)<x1, ..., r(Xm)<xm. Let us assume that R
00083      *    is ordered accordingly to ai and consider r=max(R) if the
00084      *    order between Ai and ai is EQUAL, r=min(R) otherwise).
00085      *    The substitution will substitute C0..Cn with
00086      *     -  ai<r(ai) if the order between Ai and ai is known to be EQUAL,
00087      *     -  ai>r(ai) otherwise
00088      *
00089      *    [3] if C0==(X0>x0)..Cn==(Xm>xm): the substitution is simmetric
00090      *    with respect to the * precedent case. r is chosen so that it
00091      *    holds r=min(R) if the ordering is EQUAL, or r=max(R)
00092      *    otherwise. The substitution will be:
00093      *     -  ai>r(ai), if the ordering is EQUAL
00094      *     -  ai<r(ai), otherwise.
00095      *
00096      *    [4] if G0==(X0<=X1),G1==(X1<=x1),..,Gm==(Xm<=xm): the
00097      *    substitution behaves as in [2], with the difference that <= is
00098      *    susbtituted everywhere to <
00099      *
00100      *    [5] if G0==(X0>=X1),G1==(X1>=x1),..,Gm==(Xm>=xm): the
00101      *    substitution behaves as in [2], with the difference that >= is
00102      *    susbtituted everywhere to >
00103      *
00104      * *1* If more than one conjunct is present in the predicate, the
00105      * procedure is repeated for each conjunct.
00106      *
00107      * *2* If more than one candidate substitution is present, all of them
00108      * are tried in the order in which they appear in the catalogue.
00109      *
00110      * ------------------------------------------------------------------------
00111      * IMPROVEMENTS
00112      * ------------------------------------------------------------------------
00113      * !I1! - improvement 1 (Done!)
00114      *
00115      * The algorithm for the substitution can be improved as follows:
00116      * For each conjunct C = X0 op x
00117      * find the predicate of ai that satisfies C (for instance: ai>q)
00118      * (take in account the order of the correspondence)
00119      * then put all the predicates in and.
00120      * A pictorial way to think to the process is the following:
00121      *
00122      *  X0>x0         X1>x1      X2>x2         q
00123      *  -----         -----      -----       -----
00124      *    |             *          |           |     q0 -> beg. of X1>x1 interv.
00125      *    *             *          |           |     q1 -> beg. of X0>x0 interv.
00126      *    *             *          |           |
00127      *    *             *          *           *     q2 -> beg. of X2>x2 interv.
00128      *    *             *          *           *     q3 -> end of X2>x2 interv.
00129      *    *             |          *           |
00130      *    *             |          *           |
00131      *    *             |          *           |     q4 -> end of X2>x2 interv.
00132      *    *             |          |           |
00133      *    *             |          |           |     q5 -> end of X0>x0 interv.
00134      *
00135      * the resulting predicate is: q>q2 && q<q3.
00136      * that is we need to find the max. beginning interval and the min ending one
00137      * ------------------------------------------------------------------------
00138      * !I2! - improvement 2 (Done!)
00139      *
00140      * A substitution is in the form body -> head
00141      *
00142      * At the present time the algorithm which substitute the predicates tries to
00143      * the substitutions one at a time. It would be better to implement the same
00144      * schema used for the attribute substitution (i.e. we find all possible
00145      * substitution and then we substitute the union of the head with the union
```

```
00146      * of the bodies). In order to do so, the algorithm must proceed as follows:
00147      * 1) Find all possible substitutions (with the relative ranges)
00148      * 2) Remove from the predicate list, all predicate which appear in at least
00149      *    one substitution body
00150      * 3) Add all the conditions in the head of the substitutions
00151      *
00152      * ------
00153      * Actually the improvements above will have the effect of relaxing condition
00154      * *4*[3]. The new condition will be:
00155      *   *4*[3]New : The operator are consistent with the substitution, let o1..om
00156      *               be the operators found in G0..Gm. The substitution is possible
00157      *              iff:
00158      *               - o1=o2=o3..=om== '=' OR
00159      *               - order == REVERSED OR order == EQUAL
00160      *         That is: if order==NONE we must check that all operators are '=',
00161      *               otherwise we can always perform the substitution.
00162      * The second part of the improvements, make the substitution more useful, since
00163      * it allows a greater number of equivalences to be found.
00164      * ------------------------------------------------------------------------
00165      * !I3! - improvements 3 - DEPRECATED - It were the hint of using pre
00166      *        built tables in order to make the query which select min and
00167      *        max in the algorithm described in !I1!. It is not a good idea,
00168      *        since it is much simpler and more effective to use indexes on
00169      *        all columns.
00170      * ------------------------------------------------------------------------
00171      * !I4! - improvement 4 (Done!)
00172      *        Do not add predicates which are not useful. In particular:
00173      *        in algorithm !I1!, we should not add the predicate A<=x if
00174      *        x is the minimal element of column A, similarly we should not
00175      *        add the predicate A>=x if x is the maximal element of column A.
00176      * ------------------------------------------------------------------------
00177      * !I5! - improvement 5 (Done! implemented in SimplePredAnalyzer.cpp and in
00178      *        this file)
00179      * Compact the predicates! After the substitutions it is possible that
00180      * a number of predicates became redundant. Those redundancies must be
00181      * removed for a number of good reasons:
00182      *  1) The user would find them unuseful
00183      *  2) They threaten the possibility of find useful equivalencies. In
00184      *     fact we will not be able any more to identify the two predicates:
00185      *     (1) A<=5 and A<=100       (2) A<=5
00186      *     as equivalent.
00187      *
00188      * In the following we try to tackle this problem in the simplified setting
00189      * of dealing only with conjunction of predicates.
00190      * The disjunction of conjunction is handled by removing redundancies in
00191      * each conjunction separately and independendtly from the other conjuncts.
00192      * The algorithm to deal with a conjunction is the following:
00193      * - Select the first conjunct (this is the FIRST predicate)
00194      * - compare it to each of the following ones taking one of the action
00195      *   described below (during the comparison such predicates are referred
00196      *   as the SECOND predicate)
00197      * - select the next conjunct and repeat the process.
00198      *
00199      * when we need to eliminate a predicate we always remove the SECOND
00200      * predicate. When the FIRST has to be removed we swap the two predicates
00201      * and then we remove the SECOND.
00202      *
00203      * Unfortunately to identify and remove redundancies is not too simple
00204      * for a number of techninal reasons:
00205      *  1) The ordering of values depends on the type of the column involved.
00206      *     -> we must access to the DB for obtaining such information
00207      *  2) Many different situations may arise depending on the combination of
00208      *     the kind of operators involved and the ordering of values involved.
00209      *     Depending on the situation we could take one of the following actions:
00210      *          -> : the FIRST predicate implies the second one, i.e. the SECOND
00211      *               one is redundant and we should eliminate it
00212      *          <- : the SECOND predicate implies the first one.
00213      *          <->: the two predicates are equivalent, we should eliminate one
00214      *               of the two
00215      *           / : the two predicates are unrelated, we cannot do anything
00216      *           ! : the two predicates form a contraddiction. The whole
00217      *               and list can be discarded
00218      *           = : the combination of the two predicates is equivalent to
00219      *               one which uses one the operator '='. '=' should be subst.
00220      *               in one of the two predicates and the unchanged predicate
00221      *               should be discarded.
00222      *           < : as for '=', but for operator '<'
00223      *           > : as for '=', but for operator '>'
00224      *
00225      * Let us call X the FIRST predicate and Y the SECOND one, which action
00226      * should be taken depends on the following things:
00227      *   a) the operator Xop of predicate X
00228      *   b) the operator Yop of predicate Y
00229      *   c) the relative ordering of values appearing in X and Y (we denote
00230      *      with - the situation in which X value < Y value, with 0 the
00231      *      situation in which X value == Y value, and with + the situation
00232      *      in which X value > Y value.
```

```
00233      *
00234      * The following table reports for each of the 98 combinations the right
00235      * action to be performed.
00236      *
00237      * +-------+---+---+---+---+---+---+
00238      * |Xop\Yop| < | <=| = | >=| > | <>|
00239      * +-------+---+---+---+---+---+---+
00240      * |-      | ->| ->| ->| / | / | / |
00241      * |0    < |<->|<- | ! | ! | ! |<- |
00242      * |+      |<- |<- | ! | ! | ! |<- |
00243      * +-------+---+---+---+---+---+---+
00244      * |-      | ->| ->| ->| / | / | / |
00245      * |0   <= | ->|<->| ->| = | ! | < |
00246      * |+      |<- |<- | ! | ! | ! |<- |
00247      * +-------+---+---+---+---+---+---+
00248      * |-      | ! | ! | ! |<- |<- |<- |
00249      * |0    = | ! |<- |<->|<- | ! | ! |
00250      * |+      |<- |<- | ! | ! | ! |<- |
00251      * +-------+---+---+---+---+---+---+
00252      * |-      | ! | ! | ! |<- |<- |<- |
00253      * |0   >= | ! | = | ->|<->| ->| > |
00254      * |+      | / | / | ->| ->| ->| / |
00255      * +-------+---+---+---+---+---+---+
00256      * |-      | ! | ! | ! |<- |<- |<- |
00257      * |0    > | ! | ! | ! |<- |<->|<- |
00258      * |+      | / | / | ->| ->| ->| / |
00259      * +-------+---+---+---+---+---+---+
00260      * |-      | ->| ->| ->| / | / | / |
00261      * |0   <> | ->| < | ! | > | ->|<->|
00262      * |+      | / | / | ->| ->| ->| / |
00263      * +-------+---+---+---+---+---+---+
00264      *
00265      * ------------------------------------------------------------------------
00266      * !I6! Improvement 6 - we should substituted before hand all '<' predicates
00267      *        with '<='. More formally: Given the predicate A<x we should
00268      *        substitute it with A<=y where y is the first value of column A
00269      *        which satisfies A<x. The same have to be done for predicates A>x.
00270      *
00271      */
00272
00273
00274
00275
00276     // Implements *1*
00277     void QueryNormalizer::substituteInPredicate(
00278                                                 list_OR_node* cond,
00279                                                 const OptimizerCatalogue::CatalogueEntry& catEntry )
       const {
00280         while( cond != NULL ) {
00281             substituteInAndList( cond->l_and, catEntry, "BODY." );
00282             substituteInAndList( cond->l_and, catEntry, "HEAD." );
00283             substituteInAndList( cond->l_and, catEntry, "" );
00284             compactPredicates( cond->l_and );
00285             cond = cond->next;
00286         }
00287     }
00288
00289
00290
00291     // Implements *2* *3* !I2!
00292     void QueryNormalizer::substituteInAndList( list_AND_node*& andList,
00293                                               const OptimizerCatalogue::CatalogueEntry& catEntry,
00294                                               const std::string& prefix) const {
00295         BodyInfo bodies;    // information needed to delete old predicates
00296         HeadInfo heads;     // will be used to build new predicates
00297
00298         // gathering substitution informations
00299         OptimizerCatalogue::CatalogueEntry::const_iterator it;
00300         for(it=catEntry.begin(); it!=catEntry.end(); it++) {
00301             findSubstitutions( andList,
00302                                it->first,
00303                                it->second.first,
00304                                it->second.second,
00305                                bodies,
00306                                heads,
00307                                prefix);
00308         }
00309
00310         removeAndNodes( bodies );
00311         addNewConditions( andList, heads, prefix );
00312     }
00313
00314
00315     std::string QueryNormalizer::reverseOperator(std::string op, bool doReverse) const {
00316         if(!doReverse)
00317             return op;
00318
```

```
00319         if( op == "<" )
00320             return ">";
00321 
00322         if( op == "<=" )
00323             return ">=";
00324 
00325         if( op == ">" )
00326             return "<";
00327 
00328         if( op == ">=" )
00329             return "<=";
00330 
00331         return op;
00332     }
00333 
00334     // Implements checking of conditions *4*[2] and *4*[3]. Building meanwhile
00335     // the pieces information needed for the substitution
00336 
00337     bool QueryNormalizer::findSubstitutions( list_AND_node*& andList,
00338                                              const OptimizerCatalogue::KeyCols& Ai,
00339                                              const OptimizerCatalogue::KeyCols& ai,
00340                                              const OptimizerCatalogue::OrderType order,
00341                                              BodyInfo& bodies,
00342                                              HeadInfo& heads,
00343                                              const std::string& prefix) const {
00344         if( andList == NULL )
00345             return false;
00346 
00347         list_AND_node** prevNode=&andList;
00348         list_AND_node* curNode=andList;
00349         size_t curPos = 0;
00350 
00351         OptimizerCatalogue::KeyCols foundCols;
00352         HeadInfo foundHeadEntries;
00353         BodyInfo foundBodyEntries;
00354 
00355 
00356 
00357         //
00358         // Note: in the following portion of code, the ill-famed "goto"
00359         // statement is used. I think that the code is much cleaner than
00360         // it would have been without it (in fact the "goto" statement is
00361         // used only to provide a sort of "continue" command, continue
00362         // cannot be use because it would skip the "curNode=curNode->next"
00363         // statement at the end of the while).
00364         while( curNode!=NULL ) {
00365             assert( curNode->sp != NULL );
00366 
00367             SubstEntryHead::Elem headElem;
00368             SubstEntryBody bodyEntry;
00369             headElem.order = order;
00370 
00371             if( SQLUtils::isAttribute(curNode->sp->val1) ) {
00372                 headElem.colName = curNode->sp->val1;
00373                 headElem.value = curNode->sp->val2;
00374                 headElem.op = curNode->sp->op;
00375             }
00376             else if( SQLUtils::isAttribute(curNode->sp->val2) ) {
00377                 headElem.colName = curNode->sp->val2;
00378                 headElem.value = curNode->sp->val1;
00379                 headElem.op = reverseOperator( curNode->sp->op );
00380             } else
00381                 goto nextNode; // if neither val1, or val2 are attributes than this node is not
00382             // of any interest
00383 
00384             if( order==OptimizerCatalogue::None &&
00385                headElem.op!="=" ) // condition *4*[3]New
00386                 goto nextNode;
00387 
00388             // We need to take in account that the mine rule may refer to single attributes
00389             // by prefixing them with HEAD and BODY. Since in the catalogue we do not have
00390             // such prefixes we need to delete them now. We need also to be sure to re-add
00391             // such prefixes when we add the substituted rules to the AND_list.
00392             size_t prefPos;
00393 
00394             if( prefix!="" ) {
00395                 if( (prefPos=headElem.colName.find(prefix))!=headElem.colName.npos ) {
00396                     headElem.colName.erase(prefPos,5);
00397                 } else goto nextNode;
00398             }
00399 
00400             if( Ai.find( headElem.colName )!=Ai.end() ) {
00401                 if( SQLUtils::isAttribute( headElem.value ) ) // condition *4*[2]
00402                     goto nextNode;
00403 
00404                 // All condition are met and an element of Ai found
00405                 // inserting the proper informations in sInfo and foundCols
```

```
00406                 bodyEntry.pred = prevNode;
00407                 bodyEntry.pos = curPos;
00408
00409                 foundCols.insert( headElem.colName );
00410                 foundBodyEntries.insert( bodyEntry );
00411
00412                 SubstEntryHead headEntry;
00413                 headEntry.refKey = ai;
00414                 HeadInfo::iterator it = foundHeadEntries.find( headEntry );
00415
00416                 if( it==foundHeadEntries.end() ) {
00417                     headEntry.elems.push_back(headElem);
00418                     foundHeadEntries.insert( headEntry );
00419                 }
00420                 else
00421                     const_cast< std::vector<SubstEntryHead::Elem>& >(it->elems).push_back(headElem);
00422             }
00423
00424         nextNode:
00425             prevNode=&curNode->next;
00426             curNode=curNode->next;
00427             curPos++;
00428         }
00429
00430
00431         if( foundCols == Ai ) {
00432             // all Ai elements are matched in the andList.
00433             // We can apply the substitution.
00434             bodies.insert( foundBodyEntries.begin(), foundBodyEntries.end() );
00435             heads.insert( foundHeadEntries.begin(), foundHeadEntries.end() );
00436
00437             return true;
00438         }
00439
00440         // everything ok
00441         return false;
00442     }
00443
00444
00445     void QueryNormalizer::removeAndNodes( BodyInfo& bodies ) const {
00446         BodyInfo::reverse_iterator rit;
00447         for( rit=bodies.rbegin(); rit!=bodies.rend(); rit++ ) {
00448             list_AND_node* curr = *rit->pred;
00449             assert( curr!=NULL );
00450             *rit->pred = curr->next;
00451
00452             delete curr->sp->val1;
00453             delete curr->sp->val2;
00454             delete curr->sp->op;
00455             delete curr->sp;
00456             delete curr;
00457         }
00458     }
00459
00460     void QueryNormalizer::addNewConditions( list_AND_node*& andList,
00461                                             const HeadInfo& heads,
00462                                             const std::string& prefix) const {
00463         HeadInfo::const_iterator it;
00464         for( it=heads.begin(); it!=heads.end(); it++ ) {
00465             assert( it->refKey.size()==1 );
00466             std::string ai = *(it->refKey.begin());
00467
00468             // note: the following is NOT a recursive call to this function (note
00469             //  that this function ends with an 's' while the following one does
00470             //  not).
00471             addNewCondition(andList, ai, it->elems, prefix);
00472         }
00473     }
00474
00475
00476
00477     void QueryNormalizer::setSimplePred(simple_pred* pred,
00478                                         std::string colName,
00479                                         std::string op,
00480                                         SQLUtils::Type type,
00481                                         const std::string& value) const {
00482         std::string quote;
00483
00484         if( type==SQLUtils::String )
00485             quote = "'";
00486         else
00487             quote = "";
00488
00489         std::string val2=quote+ value + quote;
00490
00491         pred->val1 = new char[colName.length()+1];
00492         pred->val2 = new char[val2.length()+1];
```

```
00493         pred->op = new char[op.length()+1];
00494
00495         strcpy(pred->val1, colName.c_str() );
00496         strcpy(pred->op, op.c_str());
00497         strcpy(pred->val2, val2.c_str() );
00498     }
00499
00500     void QueryNormalizer::addAndNode(list_AND_node*& andList,
00501                                      simple_pred* pred) const {
00502         list_AND_node* andNode = new list_AND_node;
00503         andNode->sp = pred;
00504         andNode->next = andList;
00505         andList = andNode;
00506     }
00507
00508
00509
00510     void QueryNormalizer::addNewCondition(   list_AND_node*& andList,
00511                                         const std::string& aiVal,
00512                                         const std::vector<SubstEntryHead::Elem>& elems,
00513                                         const std::string& prefix) const {
00514         std::string queryCondition;
00515         std::vector<SubstEntryHead::Elem>::const_iterator it=elems.begin();
00516         assert(it!=elems.end());
00517
00518         queryCondition = it->colName + it->op + it->value;
00519         it++;
00520
00521         for(; it!=elems.end(); it++) {
00522             queryCondition += " AND " + it->colName + it->op + it->value;
00523         }
00524
00525         std::string query = (std::string)
00526         "SELECT " +   " min(" + aiVal + "), max(" + aiVal +")"
00527         " FROM " + mr->tab_source +
00528         " WHERE " + queryCondition; // +
00529         // " ORDER BY " +aiVal;
00530
00531         mrdb::Connection* con =
00532         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00533         mrdb::Statement* state=NULL;
00534         mrdb::ResultSet* rs=NULL;
00535
00536         simple_pred* lower_bound  = NULL;
00537         simple_pred* upper_bound  = NULL;
00538
00539         try {
00540             state = con->createStatement();
00541             rs = state->executeQuery(query);
00542
00543             if( ! rs->next() ) {
00544                 lower_bound = new simple_pred;
00545                 lower_bound->val1 = new char[2];
00546                 lower_bound->val2 = new char[2];
00547                 lower_bound->op = new char[2];
00548
00549                 strcpy( lower_bound->val1, "0" );
00550                 strcpy( lower_bound->val2, "1" );
00551                 strcpy( lower_bound->op, "=");
00552             } else if( rs->getString(1)==rs->getString(2) ) {
00553                 lower_bound = new simple_pred;
00554                 std::string val = rs->getString(1);
00555                 setSimplePred( lower_bound, prefix+aiVal, "=", SQLUtils::getType(rs, 1), val);
00556             } else {
00557                 std::string minVal=rs->getString(1);
00558                 std::string maxVal=rs->getString(2);
00559
00560                 delete rs;
00561                 delete state;
00562                 rs=NULL;
00563                 state=NULL;
00564
00565                 query = (std::string)
00566                 "SELECT " +   " min(" + aiVal + "), max(" + aiVal +")"
00567                 " FROM " + mr->tab_source;
00568
00569                 state = con->createStatement();
00570                 rs = state->executeQuery(query);
00571
00572                 assert( rs->next() );
00573
00574                 std::string new_aiVal = prefix+aiVal;
00575
00576                 if( rs->getString(1)!=minVal ) {
00577                     lower_bound = new simple_pred;
00578                     setSimplePred( lower_bound, new_aiVal, ">=", SQLUtils::getType(rs,1), minVal );
00579                 }
```

```
00580
00581                 if( rs->getString(2)!=maxVal ) {
00582                     upper_bound = new simple_pred;
00583                     setSimplePred( upper_bound, new_aiVal, "<=", SQLUtils::getType(rs,2), maxVal);
00584                 }
00585             }
00586
00587             if( lower_bound!=NULL )
00588                 addAndNode( andList, lower_bound );
00589
00590             if( upper_bound!=NULL )
00591                 addAndNode( andList, upper_bound );
00592         } catch (...) {
00593             if( rs != NULL ) delete rs;
00594             if( state!=NULL ) delete state;
00595             if( lower_bound!=NULL ) delete lower_bound;
00596             if( upper_bound!=NULL ) delete upper_bound;
00597             throw;
00598         }
00599
00600         delete rs;
00601         delete state;
00602     }
00603
00604
00605     /*
00606      *  *** END (PREDICATE NORMALIZATION SECTION) ***
00607      */
00608
00609
00610
00621     void QueryNormalizer::substituteInAttrList( ParsedMinerule::AttrVector& l,
00622                                                 const OptimizerCatalogue::CatalogueEntry& catEntry
00623                                                 ) const {
00624         // each element of catEntry is a relation A->B, we call A the head and B the tail
00625 #define HEAD(a) ((a).first)
00626 #define TAIL(a) ((a).second.first)
00627
00628         // just to be sure that l is sorted...
00629         sort(l.begin(),l.end());
00630
00631         std::set<std::string> heads; // <- union of all heads elements
00632         std::set<std::string> tails; // <- union of all tails elements
00633
00634         OptimizerCatalogue::CatalogueEntry::const_iterator eit;
00635
00636         // tails  <- union of all tails in the current entry of the catalog which can be applied to l
00637         for(eit=catEntry.begin(); eit!=catEntry.end(); eit++ ) {
00638             if( TAIL(*eit).size() != 1 ) {
00639                 MRErr() « "Warning: Found a substitution A->B, whit |B|<>0, it will be
    ignored!"«std::endl;
00640                 continue;
00641             }
00642
00643             // -- looking if the current head is contained in l
00644             OptimizerCatalogue::KeyCols::const_iterator hit;
00645             for( hit=HEAD(*eit).begin();
00646                 hit!=HEAD(*eit).end()  && find(l.begin(),l.end(),*hit)!=l.end();
00647                 hit++ ) /* noop */ ;
00648
00649             if( hit!=HEAD(*eit).end() ) // if(not all head elems can be found in l)
00650                 continue;                 // skip; i.e., look for another head
00651
00652             // -- adding HEAD and TAIL to the set of found heads and tails elements
00653             // insert all head elements in "heads"
00654             copy( HEAD(*eit).begin(), HEAD(*eit).end(),
00655                   std::insert_iterator< std::set<std::string> >(heads, heads.begin()) );
00656
00657             // insert the single tail element in tails
00658             tails.insert( *TAIL(*eit).begin() );
00659         }
00660
00661
00662         // Now that we have all the relevant informations about all the rules that can
00663         // be applied, we can perform the substitution
00664
00665         std::vector<std::string> tmp;
00666         // tmp = l - heads;
00667         set_difference( l.begin(), l.end(),
00668                         heads.begin(), heads.end(),
00669                         std::insert_iterator< std::vector<std::string> >(tmp,tmp.begin()));
00670         // l = Union( tmp, tails)
00671         l.clear();
00672         set_union( tmp.begin(), tmp.end(),
00673                    tails.begin(), tails.end(),
00674                    std::insert_iterator< ParsedMinerule::AttrVector >( l, l.begin() ) );
00675
```

```
00676 #undef HEAD
00677 #undef TAIL
00678     }
00679
00680     // Substitutes < with <= and > with >=
00681     void QueryNormalizer::relaxOperators(list_OR_node* cond)
00682     {
00683         while( cond!=NULL ) {
00684             relaxOperatorsInAndList(cond->l_and);
00685             cond=cond->next;
00686         }
00687     }
00688
00689     std::string QueryNormalizer::getRelaxedValue(const std::string& tabSource,
00690                                                  const std::string& attr,
00691                                                  const std::string& op,
00692                                                  const std::string& value)
00693     {
00694         assert(op=="<" || op==">");
00695
00696         mrdb::Connection* conn =
00697         MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00698         std::auto_ptr<mrdb::Statement> state(conn->createStatement());
00699
00700         std::string sqlfun;
00701         if( op=="<" )
00702             sqlfun="max";
00703         if( op==">" )
00704             sqlfun="min";
00705
00706         std::string query =
00707         "SELECT "+sqlfun+"("+attr+") "
00708         "FROM "+tabSource+" "
00709         "WHERE "+attr+op+value;
00710
00711         std::auto_ptr<mrdb::ResultSet> result(state->executeQuery(query));
00712         if(!result->next())
00713             throw MineruleException(MR_ERROR_DATABASE_ERROR,
00714                                     "Failed to select "+sqlfun+" of column "+attr+
00715                                     " of table "+tabSource);
00716
00717         return SQLUtils::quote(result->getString(1));
00718     }
00719
00720     void QueryNormalizer::relaxOperatorInPred(simple_pred* pred)
00721     {
00722         std::string attribute;
00723         std::string op;
00724         std::string value;
00725         std::string newvalue;
00726         std::string prefix;
00727         size_t prefPos;
00728
00729         if( SQLUtils::isAttribute(pred->val1) ) {
00730             attribute=pred->val1;
00731             value = pred->val2;
00732             op = pred->op;
00733         }
00734         else {
00735             attribute=pred->val2;
00736             value = pred->val1;
00737             op = reverseOperator(pred->op);
00738         }
00739
00740         assert(op=="<" || op==">");
00741
00742         // we do not handle cross predicates (i.e. A<B where both A and B are
00743         // attributes).
00744         if(SQLUtils::isAttribute(value))
00745             return;
00746
00747         if( (prefPos=attribute.find("HEAD."))!=attribute.npos ) {
00748             attribute.erase(prefPos,5);
00749             prefix="HEAD.";
00750         }
00751
00752         if( (prefPos=attribute.find("BODY."))!=attribute.npos ) {
00753             assert(prefix=="");
00754             attribute.erase(prefPos,5);
00755             prefix="BODY.";
00756         }
00757
00758         newvalue = getRelaxedValue(mr->tab_source, attribute, op, value);
00759
00760         // we now have all the info we can make the substitutions
00761
00762         // trashing the trashable
```

```
00763         delete pred->val1;
00764         delete pred->op;
00765         delete pred->val2;
00766 
00767         attribute=prefix+attribute;
00768 
00769         // setting val1 (i.e. attribute name)
00770         pred->val1=new char[attribute.size()+1];
00771         strcpy( pred->val1, attribute.c_str() );
00772         // setting op
00773         pred->op = new char[3];
00774         pred->op[0]=op[0];
00775         pred->op[1]='=';
00776         pred->op[2]='\0';
00777         // setting val2 (i.e. newname);
00778         pred->val2=new char[newvalue.size()+1];
00779         strcpy( pred->val2, newvalue.c_str() );
00780     }
00781 
00782     void QueryNormalizer::relaxOperatorsInAndList(list_AND_node* andlist)
00783     {
00784         while(andlist!=NULL) {
00785             char* op=andlist->sp->op;
00786             if(!strcmp(op,"<") || !strcmp(op,">"))
00787                 relaxOperatorInPred(andlist->sp);
00788 
00789             andlist=andlist->next;
00790         }
00791     }
00792 
00793     void QueryNormalizer::cleanPredicate(list_OR_node*& cur) const {
00794         if(cur==NULL)
00795             return;
00796 
00797         if( cur->l_and==NULL ) {
00798             list_OR_node* old_node=cur;
00799             cur=cur->next;
00800             delete old_node;
00801             cleanPredicate(cur);
00802         } else {
00803             cleanPredicate(cur->next);
00804         }
00805     }
00806 
00807     /*
00808      In this procedure will be contained all the logic for making safe
00809      substitutions. At the present time it only handles A = B mappings
00810      (with A and B being single attributes).
00811      */
00812     void QueryNormalizer::normalize() {
00813         OptimizerCatalogue::Catalogue::const_iterator it = catalogue.find( mr->tab_source );
00814         if( it==catalogue.end() )
00815             return;  // no mapping info for this table
00816 
00817         substituteInAttrList( mr->ga, it->second );
00818         substituteInAttrList( mr->ca, it->second );
00819         substituteInAttrList( mr->ra, it->second );
00820         substituteInAttrList( mr->ba, it->second );
00821         substituteInAttrList( mr->ha, it->second );
00822 
00823         substituteInPredicate( mr->mc, it->second );
00824         substituteInPredicate( mr->gc, it->second );
00825         substituteInPredicate( mr->cc, it->second );
00826         relaxOperators(mr->mc);
00827         relaxOperators(mr->gc);
00828         relaxOperators(mr->cc);
00829 
00830         // We may have removed a number of l_and lists
00831         // an l_or node having an empty l_and list is an
00832         // invalid node, we need to clear them!
00833         cleanPredicate(mr->mc);
00834         cleanPredicate(mr->gc);
00835         cleanPredicate(mr->cc);
00836     }
00837 
00838 
00839     //  ------------------------------------------------------------------------
00840     //  COMPACT PREDICATE AND HELPERS
00841     //  ------------------------------------------------------------------------
00842 
00843     void removeNode( list_AND_node** l ) {
00844         list_AND_node* node = *l;
00845         *l = node->next;
00846         delete node->sp->val1;
00847         delete node->sp->val2;
00848         delete node->sp->op;
00849         delete node->sp;
```

```
00850         delete node;
00851     }
00852
00853     void removeAllNodes( list_AND_node*& l ) {
00854         while( l!=NULL ) {
00855             removeNode(&l);
00856         }
00857     }
00858
00859
00860
00861     // it implements !I5!.
00862
00863     void QueryNormalizer::compactPredicates( list_AND_node*& l ) const {
00864         list_AND_node** Xptr;
00865
00866         for( Xptr=&l; *Xptr!=NULL; Xptr = &((*Xptr)->next) ) {
00867             char *Xattr,*Xvalue;
00868
00869             bool XreverseOp;
00870             if(!SimplePredAnalyzer::isAttrOpValuePredicate((*Xptr)->sp,Xattr,Xvalue,XreverseOp))
00871                 continue;
00872
00873             list_AND_node** Yptr;
00874             list_AND_node* oldCell;
00875
00876             // note the strange increment in the thir for argument.
00877             // We need to advance only when we did not delete the cell
00878             // pointed by Yptr. In fact, in this situation the pointer
00879             // must stay the same (althought the cell it points to has
00880             // been changed by the remove operator).
00881             for( Yptr=&(*Xptr)->next; *Yptr!=NULL; *Yptr==oldCell ? (Yptr=&(*Yptr)->next) : 0 ) {
00882                 oldCell=*Yptr;
00883
00884                 char *Yattr,*Yvalue;
00885                 bool YreverseOp;
00886                 if(!SimplePredAnalyzer::isAttrOpValuePredicate((*Yptr)->sp,
00887                                                                Yattr,Yvalue,
00888                                                                YreverseOp))
00889                     continue;
00890
00891                 if(strcmp(Xattr,Yattr))
00892                     continue;
00893
00894                 SQLUtils::Type type =
00895                 SQLUtils::getType( MineruleOptions::getSharedOptions()
00896                                    .getMRDB()
00897                                    .getMRDBConnection(),
00898                                    mr->tab_source, Xattr);
00899
00900                 char opRel;
00901                 switch( opRel=SimplePredAnalyzer::getRelation(
00902
    reverseOperator((*Xptr)->sp->op,XreverseOp).c_str(), Xvalue,
00903
    reverseOperator((*Yptr)->sp->op,YreverseOp).c_str(), Yvalue,
00904                                                                type) ) {
00905                     case '/': // do nothing simply skip to next node
00906                         break;
00907                     case '!': // clean the main list and return
00908                         removeAllNodes(l);
00909                         return;
00910                     case '=': // substitute opRel in X and delete Y
00911                         (*Xptr)->sp->op[0]='=';
00912                         (*Xptr)->sp->op[1]='\0';
00913                         removeNode(Yptr);
00914                         break;
00915                     case '<':
00916                     case '>':
00917                         char buf[2];
00918                         buf[0]=opRel;
00919                         buf[1]='\0';
00920                         if(!strcmp((*Xptr)->sp->op,"<>"))
00921                             (*Xptr)->sp->op[0]=reverseOperator(buf,YreverseOp)[0];
00922                         else if(!strcmp((*Yptr)->sp->op,"<>"))
00923                             (*Xptr)->sp->op[0]=reverseOperator(buf,XreverseOp)[0];
00924                         else throw MineruleException( MR_ERROR_INTERNAL, "The optimizer found a
    strange operators configuration..."
00925                                                       "this is a bug!");
00926                         (*Xptr)->sp->op[1]='\0';
00927                         removeNode(Yptr);
00928                         //        nodeToRemove = Yptr;
00929                         break;
00930                     case 'l': {// substitute Y to X, delete X
00931                         simple_pred* tmp = (*Yptr)->sp;
00932                         (*Yptr)->sp=(*Xptr)->sp;
00933                         (*Xptr)->sp=tmp;
```

```
00934                         removeNode(Yptr);
00935                         //        nodeToRemove = Yptr;
00936                     }
00937                         break;
00938                     case 'r': // delete Y
00939                     case 'b': // delete Y
00940                         removeNode(Yptr);
00941                         //        nodeToRemove = Yptr;
00942                         break;
00943                     default:
00944                         throw MineruleException(MR_ERROR_OPTIMIZER_ERROR,
00945                                                 "Unknown relationship between two operators"
00946                                                 " this is a BUG!");
00947                 }
00948             }
00949
00950         }
00951     }
00952
00953
00954
00955
00956 } // namespace
```

7.275 /Users/esposito/Software/minerule/src/Parsers/MineruleParser.ypp File Reference

7.276 MineruleParser.ypp

Go to the documentation of this file.

```
00001 %{
00002 #include <list>
00003 #include <set>
00004 #include <vector>
00005 #include <iostream>
00006 #include <algorithm>
00007
00008 #include "minerule/Utils/MinMaxPair.hpp"
00009 #include "minerule/Parsers/ParsedMinerule.hpp"
00010 #include "minerule/Utils/Converter.hpp"
00011 #include "minerule/Parsers/ParserLibrary.hpp"
00012 #include "minerule/Parsers/ParsedPredicate.hpp"
00013 #include "minerule/Utils/MineruleOptions.hpp"
00014 #include "minerule/Parsers/SupportMeasure.hpp"
00015
00016 #define PARSER_BUF_SIZE 4096
00017 extern int mrlex();
00018 extern void mrerror(char const* s);
00019 bool check_body_coincident_head(minerule::ParsedMinerule::AttrVector& ,
       minerule::ParsedMinerule::AttrVector&);
00020 int find(std::vector<minerule::Bem_cond*> vect,std::string type_cond);
00021
00022 using namespace minerule;
00023
00024 extern std::map<std::string,int> keywords_to_token;
00025 /* extern std::vector<ParsedSimplePredicate*>* CrossCond; */
00026 extern minerule::ParsedPredicate* hc_mining;
00027
00028 /*****************************************/
00029 bool first_start=true;
00030 bool first_end=true;
00031 /*****************************************/
00032
00033 %}
00034
00035 %union {
00036         minerule::ParsedSimplePredicate *TParsedSimplePredicate;
00037         minerule::ParsedPredConjunction* TParsedPredConjunction;
00038         minerule::ParsedPredicate* TParsedPredicate;
00039         minerule::ParsedMinerule::AttrVector* TListType;
00040         char* TString;
00041         minerule::MinMaxPair* TMinMaxPair;
00042         minerule::SupportMeasure* ev_m;
00043     minerule::ParsedMinerule::bem_c bem_condition;
00044         }
00045
```

```
00046 %token MINE RULE AS SEQUENCES SELECT BODY HEAD CLUSTER FROM ITEM ORDER GROUP BY HAVING OR AND NOT
       WHERE EXTRACTING RULES WITH DOTS ITEMSET SUPPORT CONFIDENCE
00047 %token SEQUENCE ITEMSETS GAP SINGLE_DOT BETWEEN BEGIN_C START END MID SQUARE LIKE IN
00048
00049 %token <TString> ATTR CARD PARTBODY QUOTED_VALUE PARTHEAD COMPARISON ID AGGREGATE DATE_MR TIME_
       DISTINCT
00050 %token <TString> INT_NUMBER DBL_NUMBER
00051 %type <TString>  value
00052 %type <TString> conj_cond seq_clause distinct_clause range constant cond bool_literal between_exp
       in_exp like_exp bool_cond base_predicate
00053 %type <TString> filter_cond filter_and_cond filter_or_cond constant_commalist filter_predicate
00054 %type <bem_condition> mid_conj_cond mid_cond bem_range be_cond
00055 %type <TListType> attr_list
00056 %type <TString> qualified_attribute
00057 %type <TParsedSimplePredicate> predicate
00058 %type <TParsedPredConjunction> and_cond
00059 %type <TParsedPredicate> or_cond
00060 %type <TMinMaxPair> card gaps_clause mid_card length_clause
00061 %type <TString> from_clause aggr_member
00062 %type <TParsedPredicate> mining_cond cluster_cond
00063 %type <TParsedPredicate> having_clause having_m_cond
00064 %type <TListType> group_clause cluster_clause order_clause
00065 //%type <ev_m> ev_measures
00066 %left OR
00067 %left AND
00068
00069 %nonassoc NOT
00070
00071 %error-verbose
00072 %locations
00073
00074 %%
00075
00076 query: MINE RULE ATTR AS SELECT
00077        DISTINCT card attr_list AS BODY ',' card attr_list AS HEAD ',' SUPPORT ',' CONFIDENCE
00078        mining_cond
00079        having_m_cond
00080        from_clause
00081        group_clause
00082        having_clause
00083        cluster_clause
00084        cluster_cond
00085        EXTRACTING RULES WITH SUPPORT ':' value ',' CONFIDENCE ':' value
00086                    {
00087
00088                /*      if ($25!=NULL){
00089                        switch ($25->getMeasureType()){
00090                                                        case EvaluationMeasure::MONOTONE:
00091                                                                cout«"LA MISURA SUL SUPP E
       MONOTONA"«endl;
00092                                                break;
00093                                                        case EvaluationMeasure::ANTIMONOTONE:
00094                                                                cout«"LA MISURA SUL SUPP E
       ANTIMONOTONA"«endl;
00095                                                break;
00096                                                        default: break;
00097                        }
00098                    } else cout«"SUPPORTO NON ESPRESSO"«endl;
00099                */
00100                   #warning GESTIRE HC_MINING
00101                   /* hc_mining=$17; */
00102
00103                    ParsedMinerule& mr = getParserOutputObj();
00104                    mr.miningTask = MTMineRules;
00105                    mr.bodyCardinalities = *$7;
00106                    mr.headCardinalities = *$12;
00107
00108                    if(
       !MineruleOptions::getSharedOptions().getParsers().getBodyCardinalities().contains(mr.bodyCardinalities)
       )
00109                        yyerror("\nTHE CARDINALITY OF THE BODY IS INCORRECT REGARDING THE OPTION\n");
00110
00111                    if(
       !MineruleOptions::getSharedOptions().getParsers().getHeadCardinalities().contains(mr.headCardinalities)
       )
00112                        yyerror("\nTHE CARDINALITY OF THE HEAD IS NOT CORRECTED REGARDING THE
       OPTION\n");
00113
00114                    mr.ba = *$8;
00115                    mr.ha = *$13;
00116
00117                    /* MANAGE RULE ATTRIBUTES in way to don't have duplicate attributes*/
00118                    minerule::ParsedMinerule::AttrVector::iterator it;
00119                    minerule::ParsedMinerule::AttrVector* list_ra= new
       minerule::ParsedMinerule::AttrVector(*$8);
00120                    for (it=$13->begin();it!=$13->end();++it){
00121                                if (find(list_ra->begin(),list_ra->end(), *it)==list_ra->end())
```

```
00122                                       list_ra->push_back(*it);
00123
00124                       }
00125
00126                    mr.ra=*list_ra;
00127                    /*FINISH TO CREATE THE LIST OR RULE ATTRIBUTE, IN THIS LIST THRE ISN'T ITEM
     DUPLICATES*/
00128
00129                    mr.tab_source=$22;
00130                    mr.tab_result=$3;
00131                    mr.body_coincident_head=check_body_coincident_head(mr.ba,mr.ha);
00132                    mr.mc = $20==NULL?NULL:$20->convert();
00133                    mr.ga = *$23;
00134                    mr.gc = $24==NULL?NULL:$24->convert();
00135
00136                    if ($25!=NULL)
00137                                mr.ca=*$25;
00138
00139                    mr.cc = $26==NULL?NULL:$26->convert();
00140                    mr.sup = Converter($32).toDouble();
00141                    mr.conf= Converter($36).toDouble();
00142
00143                    //ev measure e il token $25
00144
00145                    delete $7;
00146                    delete $12;
00147                    delete $8;
00148                    delete $13;
00149                    delete $23;
00150                    if ($24!=NULL) delete $24;
00151                    //if ($21!=NULL)  delete $21;
00152                    //if ($22!=NULL)  delete $22;
00153                    free($3);
00154                    free($22);
00155
00156                   }
00157                   |
00158                   MINE ITEMSET ATTR AS SELECT
00159                          DISTINCT card attr_list AS BODY ',' SUPPORT
00160                          mining_cond
00161                          having_m_cond
00162                          from_clause
00163                          group_clause
00164                          having_clause
00165                          EXTRACTING ITEMSETS WITH SUPPORT ':' value
00166                   {
00167                        std::cout«"EFFETTUATA UNA MINE ITEMSET"«std::endl«std::endl«std::endl;
00168                        /*
00169                        if ($16!=NULL){
00170                        switch ($16->getMeasureType()){
00171                                                        case EvaluationMeasure::MONOTONE:
00172                                                                cout«"LA MISURA SUL SUPP E
     MONOTONA"«endl;
00173                                                    break;
00174                                                        case EvaluationMeasure::ANTIMONOTONE:
00175                                                                cout«"LA MISURA SUL SUPP E
     ANTIMONOTONA"«endl;
00176                                                    break;
00177                                                        default: break;
00178                        }
00179                    } else cout«"SUPPORTO NON ESPRESSO"«endl;
00180                        */
00181                    #warning GESTIRE HC_MINING
00182                        /* hc_mining=$10;  */
00183                    ParsedMinerule& mr = getParserOutputObj();
00184
00185                    mr.miningTask = MTMineItemsets;
00186                    mr.bodyCardinalities = *$7;
00187
00188                   std::cout « mr.bodyCardinalities « std::endl;
00189
00190                    if(
     !MineruleOptions::getSharedOptions().getParsers().getBodyCardinalities().contains(mr.bodyCardinalities)
     )
00191                        yyerror("\nTHE CARDINALITY IS INCORRECT REGARDING THE OPTION\n");
00192                    mr.ba = *$8;
00193                    mr.ra=*$8;
00194                    /*FINISH TO CREATE THE LIST OR RULE ATTRIBUTE, IN THIS LIST THRE ISN'T ITEM
     DUPLICATES*/
00195
00196                    mr.tab_source=$15;
00197                    mr.tab_result=$3;
00198                       mr.body_coincident_head=0;
00199            mr.mc = $13==NULL?NULL:$13->convert();
00200                    mr.ga = *$16;
00201                    mr.gc = $17==NULL?NULL:$17->convert();
00202                    mr.sup = Converter($23).toDouble();
```

```
00203                     /*
00204                     delete $7;
00205
00206                     delete $8;
00207
00208                     delete $16;
00209                     if ($13!=NULL) delete $13;
00210
00211                     free($3);
00212                     free($15);
00213                     */
00214                    }
00215                   |
00216             MINE SEQUENCE length_clause ATTR AS SELECT distinct_clause attr_list ',' SUPPORT
00217                         gaps_clause
00218                         from_clause
00219              seq_clause
00220              filter_cond
00221                         group_clause
00222                         order_clause
00223                          EXTRACTING SEQUENCES WITH SUPPORT ':' value
00224           {
00225                         ParsedMinerule& mr = getParserOutputObj();
00226                         mr.miningTask = MTMineSequences;
00227              if ($3 != NULL)
00228                  mr.length = *$3;
00229              mr.tab_result=$4;
00230              if ($7 != NULL)
00231                  mr.distinct= true;
00232              else mr.distinct= false;
00233              mr.ba=*$8;
00234              if ($11 != NULL){
00235                 mr.sequenceAllowedGaps = *$11;
00236                      }
00237                      //mr.bodyCardinalities = $9 == NULL?NULL:*$9;
00238              mr.tab_source=$12;
00239              if($14 != NULL)
00240                  mr.filter_condition=$14;
00241              mr.ga=*$15;
00242              mr.oa=*$16;
00243              mr.sup = Converter($22).toDouble();
00244
00245              first_start=true;
00246              first_end=true;
00247             /* delete $16;
00248              delete $8;
00249              delete $15;
00250
00251              delete $11;
00252              delete $3;*/
00253                   };
00254
00255
00256 // ev_measures:    {$$=NULL;}
00257 //              |
00258 //              SUPPORT COMPARISON value
00259 //              {
00260 //      std::string comp=$2;
00261 //              if (comp=="="||comp=="<>")
00262 //                      mrerror("THE RELATIONAL OPERATOR USED FOR THE SUPPORT IS INCORRECT = or <>");
00263 //              if (comp==">")
00264 //                      $$=new SupportMeasure(EvaluationMeasure::Greater, Converter($3).toDouble());
00265 //              else if (comp==">=")
00266 //                         $$=new SupportMeasure(EvaluationMeasure::GreaterEqual,
       Converter($3).toDouble());
00267 //              else if (comp=="<")
00268 //                         $$=new SupportMeasure(EvaluationMeasure::Less, Converter($3).toDouble());
00269 //              else if (comp=="<=")
00270 //                         $$=new SupportMeasure(EvaluationMeasure::LessEqual,
       Converter($3).toDouble());
00271 //              else mrerror("AN INTERNAL ERROR OCCURS");
00272 //              };
00273
00274 distinct_clause: {$$=NULL;}
00275                 | DISTINCT {$$="DISTINCT";} ;
00276
00277 from_clause: FROM ATTR{
00278                         $$=$2;
00279                         };
00280
00281 group_clause: GROUP BY attr_list { $$=$3; } ;
00282
00283 order_clause: ORDER BY attr_list { $$=$3; } ;
00284
00285 cluster_clause: {$$=NULL;}
00286                 |
00287                 CLUSTER BY attr_list { $$=$3; };
```

```
00288
00289 cluster_cond:    {$$=NULL;}
00290             |
00291                 HAVING or_cond {
00292                                 $2->stamp();
00293                                 if      (!$2->areAllBorH())
00294                                                 mrerror("\nTHE CLUSTER CONDITION IS INCORRECT, LACK
      ONE OR MORE DECLARATION OF BODY|HEAD ATTRIBUTE\n");
00295
00296                                         $$=$2;
00297                 };
00298
00299 having_m_cond: {$$=NULL;}
00300               |
00301                 HAVING or_cond { if (!$2->areAllAggr_f())
00302                                         mrerror("\nTHE HAVING CONDITION OF MINING PREDICATE IS
      INCORRECT, COULD IS A MISTAKEN DECLARATION OF ATTRIBUTE\n");
00303                                  $$=$2;
00304                                 };
00305
00306
00307 mining_cond:    {$$=NULL;}
00308             |
00309                 WHERE or_cond {  if (!$2->areAllBorH())
00310                                         mrerror("\nTHE MINING CONDITION IS INCORRECT, LACK ONE OR MORE
      DECLARATION OF BODY|HEAD ATTRIBUTE\n");
00311                                 $$=$2;
00312                 };
00313
00314 having_clause:  { $$=NULL; }
00315                 | HAVING or_cond { if ($2->atLeastOneBorH())
00316                                         mrerror("\nTHE HAVING CONDITION IS INCORRECT, COULD IS A
      MISTAKEN DECLARATION OF ATTRIBUTE\n");
00317                                 $$=$2;
00318                                 };
00319
00320
00321 qualified_attribute: PARTBODY {
00322                                 /*
00323                                  * text_tUp serve per convertire il prefisso dell'attributo della
      mining condition
00324                                  * in maiuscolo in modo che dentro il MineRule la scritta BODY (HEAD)
      antecendente al vero
00325                                  * attributo sia maiuscola, mrtext+5 si riferisce alla parola
      contenuta in mrtext senza le prime 5
00326                                  * lettere (BODY. o HEAD.) in modo tale da poterlo ricopiare in con
      strcat in text_tUp
00327                                  * l'ultima istruzione serve per dire alla strcat di far ripartire il
      buffer dalla prima posizione
00328                                  */
00329
00330
00331                                 char text_Up[4096];
00332                                 strcpy(text_Up,"BODY.");
00333                                 strncat(text_Up,strdup($1+5),PARSER_BUF_SIZE);
00334                                 $$=strndup(text_Up,PARSER_BUF_SIZE);
00335                                 //$$=$1;
00336                         }
00337                     | PARTHEAD {
00338                                 /*
00339                                  * text_tUp serve per convertire il prefisso dell'attributo della
      mining condition
00340                                  * in maiuscolo in modo che dentro il MineRule la scritta BODY (HEAD)
      antecendente al vero
00341                                  * attributo sia maiuscola, mrtext+5 si riferisce alla parola
      contenuta in mrtext senza le prime 5
00342                                  * lettere (BODY. o HEAD.) in modo tale da poterlo ricopiare in con
      strcat in text_tUp
00343                                  * l'ultima istruzione serve per dire alla strcat di far ripartire il
      buffer dalla prima posizione
00344                                  */
00345
00346                                 char text_Up[4096];
00347                                 strcpy(text_Up,"HEAD.");
00348                                 strncat(text_Up,strdup($1+5),PARSER_BUF_SIZE);
00349                                 $$=strndup(text_Up,PARSER_BUF_SIZE);
00350                                 }       ;
00351
00352 attr_list: attr_list ',' ATTR {
00353                                 $1->push_back($3);
00354                                 $$ = $1;
00355                                 free($3);
00356                                 }
00357                         | ATTR {
00358                                 $$ = new ParsedMinerule::AttrVector();
00359                                 $$->push_back($1);
00360                                 free($1);
```

```
00361                                 };
00362
00363
00364
00365 card: INT_NUMBER DOTS INT_NUMBER
00366                         {
00367                          int n_min =Converter($1).toLong();
00368                          int n_max = Converter($3).toLong();
00369                          if (n_min>n_max) // se la cardinalita massima e maggiore di quella minima
00370                              mrerror("\nTHE MIN CARDINALITY IS GREATER THAN MAX CARDINALITY\n");
00371                          else
00372                              $$=new MinMaxPair(n_min,n_max);
00373                         };
00374 card: INT_NUMBER DOTS ATTR {
00375                         int n_min = Converter($1).toLong();
00376
00377                         if( strlen($3)!=1 || toupper($3[0])!='N' ) {
00378                             mrerror("Max cardinality must be either a number or a character in
       [n|N]\n");
00379                         }
00380
00381                         $$ = new MinMaxPair(n_min, n_min);
00382                         $$->setDefaultMax();
00383                     };
00384
00385 length_clause: mid_card
00386         {
00387             $$=$1;
00388             if($$->getMin()==0) {
00389                 $$->setMin(1);
00390                 std::cout«"Min length must be greater than zero! Resetting parameter with value:
       1\n"«std::endl;
00391              }
00392         }
00393         |
00394         {$$= NULL;}
00395         ;
00396
00397 gaps_clause: WITH GAP mid_card
00398             { $$=$3; }
00399         |
00400             {$$=NULL;}
00401         ;
00402
00403         /*WITH GAP INT_NUMBER DOTS INT_NUMBER
00404             {
00405                 int n_min =Converter($3).toLong();
00406                 int n_max = Converter($5).toLong();
00407                 if (n_min>n_max) // se la cardinalita massima e maggiore di quella minima
00408                     mrerror("\nTHE MIN CARDINALITY IS GREATER THAN MAX CARDINALITY\n");
00409                 else
00410                     $$=new MinMaxPair(n_min,n_max);
00411             }
00412         |{$$= NULL;}
00413         ;*/
00414
00415 /*seq_clause: WHERE
00416                 {$$= NULL;}
00417             |{$$= NULL;}
00418         ;*/
00419
00420 value: INT_NUMBER {$$=$1;}
00421         |
00422         DBL_NUMBER {$$=$1;};
00423
00424
00425
00426 aggr_member: AGGREGATE '(' qualified_attribute ')'
00427                 {
00428                  std::string temp= $1;
00429                  temp=temp+"("+$3+")";
00430                  $$=strndup(temp.c_str(),PARSER_BUF_SIZE);
00431                 };
00432
00433 or_cond: and_cond
00434                 { $$ = new ParsedPredicate();
00435                   $$->push_back($1);
00436                 }
00437         |
00438         or_cond OR or_cond
00439                 {
00440                   *$1 |= *$3;
00441                   $$ = $1;
00442                   delete $3;
00443                 }
00444         |
00445         or_cond AND or_cond
```

```
00446                 {
00447                       *$1 &= *$3;
00448                       $$ = $1;
00449                       delete $3;
00450                 }
00451         |
00452         NOT '(' or_cond ')'
00453                 {
00454
00455                         $$=&dynamic_cast<ParsedPredicate&>(!(*$3));
00456                         delete $3;
00457                 }
00458         | '(' or_cond ')'
00459                 {
00460                   $$=$2;
00461                 };
00462
00463
00464 and_cond:
00465     predicate
00466                 {
00467                 $$ = new ParsedPredConjunction();
00468                 $$->push_back($1);
00469                 }
00470         |
00471     and_cond AND predicate
00472                 {
00473
00474                           $1->push_back($3);
00475                           $$=$1;
00476
00477                 }
00478         ;
00479 predicate:
00480     ATTR COMPARISON value
00481             {
00482             /*ParsedSimplePredicate:
00483                 firt param: first term
00484                 second param: relational operator
00485                 third param: secondo term
00486                 fourth param: true if at least one of the two term was evaluated on body or head false
      otherwise
00487                 fifth param: true if at least one of the two term was an aggregation function false
      otherwise
00488             */
00489
00490                 $$=new ParsedSimplePredicate($1,$2,$3,false,false);
00491                 free($1);
00492                 free($2);
00493                 free($3);
00494                         }
00495             |
00496     ATTR COMPARISON QUOTED_VALUE
00497             {
00498                 $$=new ParsedSimplePredicate($1,$2,$3,false,false);
00499                 free($1);
00500                 free($2);
00501                 free($3);
00502
00503             }
00504             |
00505     ATTR COMPARISON ATTR
00506             {
00507                 $$=new ParsedSimplePredicate($1,$2,$3,false,false);
00508                 free($1);
00509                 free($2);
00510                 free($3);
00511              }
00512              |
00513     value COMPARISON value {
00514                 $$=new ParsedSimplePredicate($1,$2,$3,false,false);
00515                 free($1);
00516                 free($2);
00517                 free($3);
00518              }
00519             |
00520     value COMPARISON ATTR
00521             {
00522             std::string op=$2,nop;
00523                 if (op.compare("")==0) { nop="<";}
00524                 else if (op==">=") { nop="<=";}
00525                 else if (op=="<") { nop=">";}
00526                 else if (op=="<=") { nop=">=";}
00527                 $$=new ParsedSimplePredicate($3,nop,$1,false,false);
00528                 free($1);
00529                 free($2);
00530                 free($3);
```

```
00531             }
00532             |
00533     qualified_attribute COMPARISON  QUOTED_VALUE
00534                          {
00535                                  $$=new ParsedSimplePredicate($1,$2,$3,true,false);
00536                                  free($1);
00537                                  free($2);
00538                                  free($3);
00539                          }
00540                          |
00541         qualified_attribute COMPARISON value
00542                         {
00543                                  $$=new ParsedSimplePredicate($1,$2,$3,true,false);
00544                                  free($1);
00545                                  free($2);
00546                                  free($3);
00547                         }
00548                         |
00549         qualified_attribute COMPARISON qualified_attribute
00550                         {
00551                                  $$=new ParsedSimplePredicate($1,$2,$3,true,false);
00552                           /*    ParsedSimplePredicate *p = new
       ParsedSimplePredicate($1,$2,$3,true,false);
00553                                                                  CrossCond->push_back(p); */
00554                                  free($1);
00555                                  free($2);
00556                                  free($3);
00557                          }
00558                         |
00559         value COMPARISON qualified_attribute
00560                         {
00561                         std::string op=$2, nop;
00562                                  if (op.compare(">")==0) { nop="<";}
00563                                  else if (op==">=") { nop="<=";}
00564                                  else if (op=="<") { nop=">";}
00565                                  else if (op=="<=") { nop=">=";}
00566                                  $$=new ParsedSimplePredicate($3,nop,$1,true,false);
00567                                  free($1);
00568                                  free($2);
00569                                  free($3);
00570                          }
00571                          |
00572         aggr_member COMPARISON  value
00573                         {
00574                                  $$=new ParsedSimplePredicate($1,$2,$3,false,true);
00575                                  free($1);
00576                                  free($2);
00577                                  free($3);
00578                          }
00579                          |
00580         aggr_member COMPARISON  aggr_member
00581                         {
00582                                  $$=new ParsedSimplePredicate($1,$2,$3,false,true);
00583                                  free($1);
00584                                  free($2);
00585                                  free($3);
00586                          }
00587                          |
00588         aggr_member COMPARISON  qualified_attribute
00589                         {
00590                                  $$=new ParsedSimplePredicate($1,$2,$3,false,true);
00591                                  free($1);
00592                                  free($2);
00593                                  free($3);
00594                          }
00595                          ;
00596
00597     /*********************************************/
00598
00599     filter_cond: WHERE filter_or_cond {
00600                          $$=$2;
00601                 }
00602                 |
00603                 {$$=NULL;}
00604                 ;
00605
00606     filter_or_cond: filter_and_cond
00607             {
00608             $$= $1;
00609             }
00610         |
00611         filter_or_cond OR filter_or_cond
00612             {
00613             std::string tmp=$1;
00614             tmp=tmp+ " OR " + $3;
00615             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE); ;
00616             }
```

```
00617         |
00618         filter_or_cond AND filter_or_cond
00619             {
00620             std::string tmp=$1;
00621             tmp=tmp+ " AND " + $3;
00622             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00623             }
00624         |
00625         NOT '(' filter_or_cond ')'
00626             {
00627             std::string tmp="NOT (";
00628             tmp=tmp+ $3 + ")";
00629             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE); ;
00630
00631             }
00632         | '(' filter_or_cond ')'
00633             {
00634             std::string tmp="NOT (";
00635             tmp=tmp+ $2 + ")";
00636             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00637             }
00638         ;
00639
00640
00641     filter_and_cond:
00642         filter_predicate
00643             {
00644             $$ = $1;
00645             }
00646         |
00647         filter_and_cond AND filter_predicate
00648             {
00649             std::string tmp=$1;
00650             tmp=tmp+ " AND " + $3;
00651             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00652             }
00653         ;
00654
00655     filter_predicate:
00656         base_predicate
00657         |
00658         bool_literal
00659         |
00660         bool_cond
00661         ;
00662
00663     base_predicate:
00664         constant range
00665         {
00666             std::string tmp= $1;
00667             tmp += " ";
00668             tmp += $2;
00669             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00670         }
00671         ;
00672
00673     bool_cond:
00674         between_exp
00675         |
00676         like_exp
00677         |
00678         in_exp
00679         ;
00680
00681     between_exp:
00682         constant BETWEEN constant AND constant
00683         {
00684             std::string tmp=$1;
00685             tmp=tmp+ " BETWEEN " + $3 + " AND " + $5;
00686             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00687         }
00688         |
00689         constant NOT BETWEEN constant AND constant
00690         {
00691             std::string tmp=$1;
00692             tmp=tmp+ " NOT BETWEEN " + $4 + " AND " + $6;
00693             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00694         }
00695         ;
00696
00697     like_exp:
00698         constant LIKE constant
00699         {
00700             std::string tmp=$1;
00701             tmp=tmp+ " LIKE " + $3;
00702             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00703         }
```

```
00704         |
00705         constant NOT LIKE constant
00706         {
00707             std::string tmp=$1;
00708             tmp=tmp+ " NOT LIKE " + $4;
00709             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00710         }
00711         ;
00712
00713     in_exp:
00714         constant IN '(' constant_commalist ')'
00715         {
00716             std::string tmp=$1;
00717             tmp=tmp+ " IN (" + $4 + ")";
00718             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00719         }
00720         |
00721         constant NOT IN '(' constant_commalist ')'
00722         {
00723             std::string tmp=$1;
00724             tmp=tmp+ " NOT IN (" + $5 + ") ";
00725             $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00726         }
00727         ;
00728
00729     seq_clause:      HAVING conj_cond {$$ = $2;}
00730                     |
00731                      {$$ = NULL;}
00732                     ;
00733
00734
00735     conj_cond:      cond AND conj_cond
00736                     |
00737                     cond
00738                     ;
00739
00740     cond:           ATTR '(' attr_list ')' range
00741                     {
00742                         ParsedMinerule& mr= getParserOutputObj();
00743                         Dist_cond* dc = new Dist_cond();
00744                         dc->function= $1;
00745                         dc->attr= *$3;
00746                         dc->range= $5;
00747                         mr.seq_dist_vect.push_back(dc);
00748                     }
00749                     |
00750                     be_cond
00751                     {
00752                         ParsedMinerule& mr= getParserOutputObj();
00753                         if($1->type.compare("BEGIN")==0){
00754                            if(first_start)
00755                                 mr.seq_bem_vect.push_back($1);
00756                             else{
00757                                int start_index= find(mr.seq_bem_vect,"BEGIN");
00758                                mr.seq_bem_vect[start_index]->and_c= $1;
00759                             }
00760                            first_start=false;
00761                         }
00762                         else if($1->type.compare("END")==0){
00763                             if(first_end)
00764                                 mr.seq_bem_vect.push_back($1);
00765                             else{
00766                                 int end_index= find(mr.seq_bem_vect,"END");
00767                                 mr.seq_bem_vect[end_index]->and_c= $1;
00768                             }
00769                             first_end=false;
00770                         }
00771                     }
00772                     |
00773                     SQUARE mid_card mid_conj_cond SQUARE
00774                     {
00775                         ParsedMinerule& mr= getParserOutputObj();
00776                         if($2) {
00777                             $3->count_min= $2->getMin();
00778                             $3->count_max= $2->getMax();
00779                         }
00780                         mr.seq_bem_vect.push_back($3);
00781                     }
00782                     |
00783                     mid_card mid_cond
00784                     {
00785                         ParsedMinerule& mr= getParserOutputObj();
00786                         if($1) {
00787                             $2->count_min= $1->getMin();
00788                             $2->count_max= $1->getMax();
00789                         }
00790                         mr.seq_bem_vect.push_back($2);
```

```
00791                    }
00792                    ;
00793
00794     be_cond:  BEGIN_C SINGLE_DOT ATTR bem_range
00795                 {
00796                    $4->type+="BEGIN";
00797                    $4->attr+=$3;
00798                    $$=$4;
00799                  }
00800               |
00801                END SINGLE_DOT ATTR bem_range
00802                  {
00803                    $4->type+="END";
00804                    $4->attr+=$3;
00805                    $$=$4;
00806                  }
00807               ;
00808
00809     mid_conj_cond : mid_cond AND mid_conj_cond
00810                   {
00811                     $1->and_c=$3;
00812                     $$=$1;
00813                   }
00814                   |
00815                   mid_cond
00816                   {$$=$1;}
00817                   ;
00818
00819     mid_cond: MID SINGLE_DOT ATTR bem_range
00820                    {
00821                      $4->type+="MID";
00822                      $4->attr+=$3;
00823                      $$=$4;
00824                     }
00825                     ;
00826
00827     mid_card: INT_NUMBER DOTS INT_NUMBER
00828                   {
00829                      int n_min =Converter($1).toLong();
00830                      int n_max = Converter($3).toLong();
00831                      if (n_min>n_max) // se la cardinalita massima e maggiore di quella minima
00832                            mrerror("\nTHE MIN CARDINALITY IS GREATER THAN MAX CARDINALITY\n");
00833                      else
00834                          $$=new MinMaxPair(n_min,n_max);
00835                    }
00836                   |
00837                   INT_NUMBER DOTS ATTR
00838                     {
00839                        int n_min =Converter($1).toLong();
00840                        if( strlen($3)!=1 || toupper($3[0])!='N')
00841                            mrerror("Max cardinality must be either a number or a character in
      [n|N]\n");
00842                        else
00843                             $$=new MinMaxPair(n_min,std::numeric_limits<int>::max()-1);
00844
00845                     }
00846                   |
00847                    { $$=NULL; }
00848                   ;
00849
00850
00851     bem_range:  BETWEEN constant AND constant
00852                {
00853                  std::string tmp1 = $2;
00854                  tmp1 += " AND ";
00855                  tmp1 += $4;
00856                  Bem_cond* bc= new Bem_cond();
00857                  bc->op+="BETWEEN";
00858                  bc->val+=tmp1;
00859                  $$=bc;
00860                 }
00861              |
00862               COMPARISON constant
00863                  {
00864                   Bem_cond* bc= new Bem_cond();
00865                   bc->op+=$1;
00866                   bc->val+=$2;
00867                   $$=bc;
00868
00869                   }
00870               ;
00871
00872     range:  BETWEEN constant AND constant
00873              {
00874                std::string tmp = "BETWEEN ";
00875                tmp += $2;
00876                tmp += " AND ";
```

```
00877                tmp += $4;
00878                char * cstr = new char [tmp.length()+1];
00879                strcpy (cstr, tmp.c_str());
00880                $$ = cstr;
00881               }
00882             |
00883               COMPARISON constant
00884               {
00885                std::string tmp = $1;
00886                tmp =tmp+ " ";
00887                tmp += $2;
00888                char * cstr = new char [tmp.length()+1];
00889                strcpy (cstr, tmp.c_str());
00890                $$ = cstr;
00891               }
00892           ;
00893
00894     constant_commalist:
00895               constant_commalist ',' constant
00896               {
00897                   std::string tmp = $1;
00898                   tmp =tmp+ ", ";
00899                   tmp += $3;
00900                   char * cstr = new char [tmp.length()+1];
00901                   strcpy (cstr, tmp.c_str());
00902                   $$ = cstr;
00903               }
00904               |
00905               constant
00906               {
00907                   $$ = $1;
00908               }
00909               ;
00910
00911     constant:
00912              ATTR
00913              |
00914              value
00915              |
00916              DATE_MR
00917              |
00918              TIME_
00919              |
00920              bool_literal
00921              |
00922              QUOTED_VALUE
00923              ;
00924
00925     bool_literal:
00926              "true"
00927                  {
00928                  std::string tmp="TRUE";
00929                  $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00930                  }
00931              |
00932              "TRUE"
00933                  {
00934                  std::string tmp="TRUE";
00935                  $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00936                  }
00937              |
00938              "FALSE"
00939                  {
00940                  std::string tmp="FALSE";
00941                  $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00942                  }
00943              |
00944              "false"
00945                  {
00946                  std::string tmp="FALSE";
00947                  $$= strndup(tmp.c_str(),PARSER_BUF_SIZE);
00948                  }
00949                  ;
00950
00951     /*********************************************/
00952
00953 %%
00954
00955
00956 namespace minerule{
00957 void init_mrparser() {
00958         /* initialization of keywords_to_token table */
00959         if(keywords_to_token.size() > 0)
00960                 return;
00961
00962         keywords_to_token["MINE"]=MINE;
00963         keywords_to_token["RULE"]=RULE;
```

```
00964         keywords_to_token["AS"]=AS;
00965         keywords_to_token["FROM"]=FROM;
00966         keywords_to_token["WHERE"]=WHERE;
00967         keywords_to_token["GROUP"]=GROUP;
00968     keywords_to_token["BY"]=BY;
00969         keywords_to_token["HAVING"]=HAVING;
00970         keywords_to_token["EXTRACTING"]=EXTRACTING;
00971         keywords_to_token["RULES"]=RULES;
00972         keywords_to_token["WITH"]=WITH;
00973         keywords_to_token["DISTINCT"]=DISTINCT;
00974         keywords_to_token["SELECT"]=SELECT;
00975         keywords_to_token["BODY"]=BODY;
00976         keywords_to_token["HEAD"]=HEAD;
00977         keywords_to_token["NOT"]=NOT;
00978         keywords_to_token["AND"]=AND;
00979         keywords_to_token["OR"]=OR;
00980         keywords_to_token["CLUSTER"]=CLUSTER;
00981         keywords_to_token["ITEMSET"]=ITEMSET;
00982         keywords_to_token["ITEMSETS"]=ITEMSETS;
00983         keywords_to_token["SUPPORT"]=SUPPORT;
00984         keywords_to_token["CONFIDENCE"]=CONFIDENCE;
00985         keywords_to_token["SEQUENCE"]=SEQUENCE;
00986         keywords_to_token["SEQUENCES"]=SEQUENCES;
00987         keywords_to_token["ORDER"]=ORDER;
00988         keywords_to_token["GAP"]=GAP;
00989     keywords_to_token["BEGIN"]=BEGIN_C;
00990     keywords_to_token["END"]=END;
00991     keywords_to_token["MID"]=MID;
00992     keywords_to_token["BETWEEN"]=BETWEEN;
00993     keywords_to_token["LIKE"]=LIKE;
00994     keywords_to_token["IN"]=IN;
00995
00996 }
00997 }
00998
00999
01000 bool check_body_coincident_head(minerule::ParsedMinerule::AttrVector& a,
       minerule::ParsedMinerule::AttrVector& b){
01001         minerule::ParsedMinerule::AttrVector* a1 = new minerule::ParsedMinerule::AttrVector(a);
01002         minerule::ParsedMinerule::AttrVector* b1        = new minerule::ParsedMinerule::AttrVector(b);
01003 sort(a1->begin(), a1->end());
01004 sort(b1->begin(), b1->end());
01005 if (a1->size()!=b1->size())
01006         return false;
01007 minerule::ParsedMinerule::AttrVector::iterator it1,it2;
01008 for (it1=a1->begin(),it2=b1->begin(); (it1!=a1->end())&&(*it1==*it2);++it1, ++it2);//notare il punto e
       virgola
01009
01010       return (it1==a1->end());
01011
01012 }
01013
01014 bool containing(std::vector<std::string> vect, std::string el){
01015     for(int i=0; i<vect.size(); ++i)
01016         if(vect[i].compare(el)==0) return true;
01017     return false;
01018 }
01019
01020 int find(std::vector<minerule::Bem_cond*> vect,std::string type_cond){
01021     for(int i=0;i<vect.size();++i)
01022         if(vect[i]->type.compare(type_cond)==0)
01023             return i;
01024     return -1;
01025 }
```

7.277 /Users/esposito/Software/minerule/src/Parsers/Parsed↩ Minerule.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>
#include <cstdlib>
#include "minerule/Utils/MineruleException.hpp"
```

```
#include "minerule/PredicateUtils/PredicateUtils.hpp"
#include "minerule/Utils/SQLUtils.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/ResultSetMetaData.hpp"
#include <iterator>
#include "minerule/Parsers/ParsedMinerule.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Parsers/ParserLibrary.hpp"
```

Namespaces

- namespace minerule

Functions

- bool find_in_list (std::string s, minerule::ParsedMinerule::AttrVector l)
- int init_analisi_minerule (const std::string &)
- void print_AND_list (std::ostream &os, list_AND_node *andnode)
- void print_OR_list (std::ostream &os, list_OR_node *ornode)
- void mrerror (char const *)
- list_AND_node * clone_l_AND (list_AND_node *l)
- list_OR_node * clone_l_OR (list_OR_node *l)
- std::ostream & minerule::operator<< (std::ostream &os, const ParsedMinerule &mr)
- float minerule::floatFromText (char *text)
- std::string minerule::trim (std::string text)

7.277.1 Function Documentation

7.277.1.1 clone_l_AND()

```
list_AND_node * clone_l_AND (
            list_AND_node * l )
```

Definition at line 84 of file ParsedMinerule.cpp.

```
00085 {
00086         list_AND_node* result;
00087         simple_pred*   p;
00088
00089         if (l==NULL) return NULL;
00090         result=(list_AND_node*)malloc(sizeof(list_AND_node));
00091
00092         if (result==NULL)
00093                 mrerror("Cannot allocate memory for a new list_AND_node");
00094
00095         p=(simple_pred*)malloc(sizeof(simple_pred));
00096
00097         if (p==NULL)
00098                 mrerror("simple_pred");
00099
00100         p->val1=strdup(l->sp->val1);
00101         p->op=strdup(l->sp->op);
00102         p->val2=strdup(l->sp->val2);
00103
00104         result->sp=p;
00105         result->next=clone_l_AND(l->next);
00106
00107         return result;
00108 }
```

7.277.1.2 clone_l_OR()

```
list_OR_node * clone_l_OR (
            list_OR_node * l )
```

Definition at line 111 of file ParsedMinerule.cpp.

```
00112 {
00113         list_OR_node* result;
00114
00115         if (l==NULL) return NULL;
00116
00117         result=(list_OR_node*)malloc(sizeof(list_OR_node));
00118         if (result==NULL)
00119                 mrerror("Cannot allocate memory for a new list_OR_node");
00120
00121         result->l_and=clone_l_AND(l->l_and);
00122         result->next=clone_l_OR(l->next);
00123         return result;
00124 }
```

7.277.1.3 find_in_list()

```
bool find_in_list (
            std::string s,
            minerule::ParsedMinerule::AttrVector l )
```

7.277.1.4 init_analisi_minerule()

```
int init_analisi_minerule (
            const std::string &  )
```

7.277.1.5 mrerror()

```
void mrerror (
            char const *  )
```

7.277.1.6 print_AND_list()

```
void print_AND_list (
            std::ostream & os,
            list_AND_node * andnode )
```

Definition at line 45 of file ParsedMinerule.cpp.

```
00045                                                   {
00046         if( andnode==NULL )
00047                 return;
00048
00049         if( andnode->sp!=NULL ) {
00050                 assert( andnode->sp->val1!=NULL &&
00051                         andnode->sp->op!=NULL &&
00052                                 andnode->sp->val2 != NULL );
00053
00054                 std::cout « andnode->sp->val1;
00055                 std::cout « " " « andnode->sp->op;
00056                 std::cout « " " « andnode->sp->val2;
00057         }
00058
00059         if( andnode->next!=NULL ) {
00060                 os « " AND ";
00061                 print_AND_list(os, andnode->next);
00062         }
00063 }
```

7.277.1.7 print_OR_list()

```
void print_OR_list (
            std::ostream & os,
            list_OR_node * ornode )
```

Definition at line 66 of file ParsedMinerule.cpp.

```
00066                                                  {
00067         if( ornode==NULL )
00068                 return;
00069
00070         if( ornode->l_and!=NULL ) {
00071                 os « "(";
00072                 print_AND_list(os, ornode->l_and);
00073                 os  « ")";
00074         }
00075
00076         if( ornode->next!=NULL ) {
00077                 os « " OR ";
00078                 print_OR_list(os, ornode->next);
00079         }
00080 }
```

7.278 ParsedMinerule.cpp

Go to the documentation of this file.

```
00001 #include<stdio.h>
00002 #include<stdlib.h>
00003 #include<string>
00004 #include<iostream>
00005 #include<sstream>
00006 #include<fstream>
00007 #include<cstdlib>
00008
00009 /*
00010   #ifndef String
00011   #  include<wxstrgnu.h>
00012 typedef wxString String;
00013  #endif
00014 */
00015
```

```
00016 #include "minerule/Utils/MineruleException.hpp"
00017 #include "minerule/PredicateUtils/PredicateUtils.hpp"
00018 #include <sstream>
00019 #include "minerule/Utils/SQLUtils.hpp"
00020 //#include"Ottimizzatore/ottimizzatore.h"
00021
00022
00023 #include"minerule/mrdb/Connection.hpp"
00024 #include"minerule/mrdb/ResultSet.hpp"
00025 #include"minerule/mrdb/Statement.hpp"
00026 #include"minerule/mrdb/ResultSetMetaData.hpp"
00027 #include<iterator>
00028
00029 #include "minerule/Parsers/ParsedMinerule.hpp"
00030 #include "minerule/Utils/MineruleOptions.hpp"
00031 #include "minerule/Parsers/ParserLibrary.hpp"
00032
00033 /*#define yyerrror predicateerror
00034
00035 extern void yyerror(char*); */
00036
00037
00038
00039         bool find_in_list(std::string s,minerule::ParsedMinerule::AttrVector l);
00040
00041 //int init_analisi_minerule(int ,char **);
00042 int init_analisi_minerule(const std::string&);
00043
00044 void
00045 print_AND_list(std::ostream& os, list_AND_node* andnode) {
00046         if( andnode==NULL )
00047                 return;
00048
00049         if( andnode->sp!=NULL ) {
00050                 assert( andnode->sp->val1!=NULL &&
00051                         andnode->sp->op!=NULL &&
00052                                 andnode->sp->val2 != NULL );
00053
00054                 std::cout « andnode->sp->val1;
00055                 std::cout « " " « andnode->sp->op;
00056                 std::cout « " " « andnode->sp->val2;
00057         }
00058
00059         if( andnode->next!=NULL ) {
00060                 os « " AND ";
00061                 print_AND_list(os, andnode->next);
00062         }
00063 }
00064
00065 void
00066 print_OR_list(std::ostream& os, list_OR_node* ornode) {
00067         if( ornode==NULL )
00068                 return;
00069
00070         if( ornode->l_and!=NULL ) {
00071                 os « "(";
00072                 print_AND_list(os, ornode->l_and);
00073                 os  « ")";
00074         }
00075
00076         if( ornode->next!=NULL ) {
00077                 os « " OR ";
00078                 print_OR_list(os, ornode->next);
00079         }
00080 }
00081
00082 extern void mrerror(char const*);
00083
00084 list_AND_node* clone_l_AND(list_AND_node* l)
00085 {
00086         list_AND_node* result;
00087         simple_pred*   p;
00088
00089         if (l==NULL) return NULL;
00090         result=(list_AND_node*)malloc(sizeof(list_AND_node));
00091
00092         if (result==NULL)
00093                 mrerror("Cannot allocate memory for a new list_AND_node");
00094
00095         p=(simple_pred*)malloc(sizeof(simple_pred));
00096
00097         if (p==NULL)
00098                 mrerror("simple_pred");
00099
00100         p->val1=strdup(l->sp->val1);
00101         p->op=strdup(l->sp->op);
00102         p->val2=strdup(l->sp->val2);
```

```
00103
00104         result->sp=p;
00105         result->next=clone_l_AND(l->next);
00106
00107         return result;
00108 }
00109
00110
00111 list_OR_node* clone_l_OR(list_OR_node* l)
00112 {
00113         list_OR_node* result;
00114
00115         if (l==NULL) return NULL;
00116
00117         result=(list_OR_node*)malloc(sizeof(list_OR_node));
00118         if (result==NULL)
00119                 mrerror("Cannot allocate memory for a new list_OR_node");
00120
00121         result->l_and=clone_l_AND(l->l_and);
00122         result->next=clone_l_OR(l->next);
00123         return result;
00124 }
00125
00126 namespace minerule {
00127
00128         std::ostream& operator«(std::ostream& os, const ParsedMinerule& mr) {
00129                 os « "ParsedMinerule" « std::endl;
00130                 os « " - ba:";
00131                 copy( mr.ba.begin(), mr.ba.end(), std::ostream_iterator<std::string>(os, " "));
00132                 os « std::endl;
00133
00134         os « " - oa:";
00135         copy( mr.oa.begin(), mr.oa.end(), std::ostream_iterator<std::string>(os, " "));
00136         os « std::endl;
00137
00138                 os « " - ha:";
00139                 copy( mr.ha.begin(), mr.ha.end(), std::ostream_iterator<std::string>(os, " "));
00140                 os « std::endl;
00141
00142                 os « " - ga:";
00143                 copy( mr.ga.begin(), mr.ga.end(), std::ostream_iterator<std::string>(os, " "));
00144                 os « std::endl
00145                         « " - ca:";
00146                 copy( mr.ca.begin(), mr.ca.end(), std::ostream_iterator<std::string>(os, " "));
00147                 os « std::endl
00148                         « " - ra:";
00149                 copy( mr.ra.begin(), mr.ra.end(), std::ostream_iterator<std::string>(os, " "));
00150                 os « std::endl
00151                         « " = mc:";
00152                 print_OR_list(os,mr.mc);
00153                 os « std::endl
00154                         « " = gc:";
00155                 print_OR_list(os,mr.gc);
00156                 os « std::endl
00157                         « " = cc:";
00158                 print_OR_list(os,mr.cc);
00159                 os « std::endl;
00160
00161                 os « " ! clust. agg. list:";
00162                 copy( mr.c_aggr_list.begin(),
00163                         mr.c_aggr_list.end(),
00164                                 std::ostream_iterator<std::string>(os, " "));
00165                 os « std::endl;
00166
00167                 os « " * sup:" « mr.sup « std::endl;
00168                 os « " * conf:" « mr.conf « std::endl;
00169                 os « " & tautologies: "«mr.tautologies « std::endl;
00170                 os « " & body coinc head:"«mr.body_coincident_head « std::endl;
00171                 os « " tab_source:" « mr.tab_source « std::endl;
00172                 os « " tab_result:" « mr.tab_result « std::endl;
00173                 os « "ParsedMinerule - end" « std::endl;
00174
00175                 return os;
00176         }
00177
00178
00179   /*
00180         void
00181         clean_parser() {
00182     //  extern struct node *par_tree[4];
00183     //  extern void parser_free_all_memory();
00184     //  parser_free_all_memory();
00185
00186                 for( int i=0; i<4; i++ ) {
00187                         par_tree[i]=NULL;
00188                 }
00189
```

```
00190                 for(int i=0; i<6 ; i++ ) {
00191                         par_sqlcode[i]="";
00192                 }
00193
00194                 for(int i=0; i<10; i++) {
00195                         deall_att_list(par_attlist[i]);
00196                         par_attlist[i]=NULL;
00197                 }
00198         }
00199 */
00200
00201         float
00202         floatFromText(char* text) {
00203                 char* endptr;
00204                 float tmp = strtof(text, &endptr);
00205
00206                 assert( endptr!=text );
00207                 assert( errno!=ERANGE );
00208
00209                 return tmp;
00210         }
00211
00212         std::string
00213         trim(std::string text ) {
00214                 size_t begSpaces = text.find_first_not_of(" ");
00215                 if( begSpaces == text.npos ) // all elements are spaces
00216                         return "";
00217
00218                 text.erase( 0, begSpaces );
00219
00220                 size_t endSpaces = text.find_last_not_of(" ");
00221                 if( endSpaces == text.npos || endSpaces+1 > text.length() )
00222                         return text;
00223
00224                 text.erase( endSpaces+1, text.length()-endSpaces );
00225                 return text;
00226         }
00227
00228 /*
00229         void
00230         ParsedMinerule::fillAttrList(AttrVector& dest, struct att_list* source) {
00231                 struct att_list* it=source;
00232                 while( it!=NULL ) {
00233                         dest.push_back(it->attrname);
00234                         it=it->next;
00235                 }
00236                 } */
00237
00238                         void
00239                 ParsedMinerule::init(const std::string& minerule_text) {
00240                         parseMinerule( minerule_text, *this );
00241                 }
00242
00243                 std::string ParsedMinerule::getAttrText(const AttrVector& l) const {
00244                         std::string result;
00245                         AttrVector::const_iterator it = l.begin();
00246                         assert(it!=l.end());
00247
00248                         result = *it;
00249                         it++;
00250
00251                         for(;it!=l.end();it++) {
00252                                 result += "," + *it;
00253                         }
00254
00255                         return result;
00256                 }
00257                 std::string ParsedMinerule::getSimplePredText(const simple_pred* pred) const {
00258                         assert(pred!=NULL);
00259                         std::string result;
00260                         if( pred->val1!=NULL)
00261                                 result+=pred->val1;
00262
00263                         if(pred->op!=NULL)
00264                                 result+=pred->op;
00265
00266                         if(pred->val2!=NULL)
00267                                 result+=pred->val2;
00268
00269                         return result;
00270                 }
00271                 std::string ParsedMinerule::getAndListText(const list_AND_node* cond)const {
00272                         assert( cond != NULL);
00273                         std::string result="";
00274
00275                         result = getSimplePredText(cond->sp);
00276                         cond= cond->next;
```

```
00277
00278                         while(cond!=NULL) {
00279                                 result+=" AND "+getSimplePredText(cond->sp);
00280                                 cond=cond->next;
00281                         }
00282
00283                         return result;
00284                 }
00285                 std::string ParsedMinerule::getCondText(const list_OR_node* cond)const {
00286                         assert(cond!=NULL);
00287                         std::string result ="";
00288
00289                         result = "("+getAndListText(cond->l_and)+")";
00290                         cond = cond->next;
00291
00292                         while(cond!=NULL) {
00293                                 result += " OR (" + getAndListText(cond->l_and) + ")";
00294                                 cond=cond->next;
00295                         }
00296
00297                         return result;
00298                 }
00299                 std::string ParsedMinerule::getCardsText(const MinMaxPair& mm) const {
00300                         std::string result;
00301                         result=Converter(mm.getMin()).toString()+"..";
00302                         if( mm.getMax()!=mm.getDefaultMax() )
00303                                 result+=Converter(mm.getMax()).toString();
00304                         else
00305                                 result+="n";
00306                         return result;
00307                 }
00308                 std::string ParsedMinerule::getMinesequenceText() const {
00309                         std::string result;
00310             std::string distinct= this->distinct? " DISTINCT " : " " ;
00311             std::string Bem_text= getBEMText();
00312             if(Bem_text.size()>0)
00313                 Bem_text= " HAVING " + Bem_text;
00314                         result =
00315                                 "MINE SEQUENCE " + tab_result + " AS " +
00316                 "SELECT " + distinct +
00317                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) +
00318                     ", SUPPORT FROM "+tab_source+" "+Bem_text+" GROUP BY "+getAttrText(ga)+
00319                                 " ORDER BY "+getAttrText(oa)+ " EXTRACTING SEQUENCES WITH SUPPORT:" +
00320                                 Converter(sup).toString();
00321
00322                         return result;
00323                 }
00324                 std::string ParsedMinerule::getText() const {
00325                         switch( miningTask ) {
00326                                 case MTMineRules:
00327                                 return getMineruleText();
00328                                 case MTMineSequences:
00329                                 return getMinesequenceText();
00330                                 case MTMineItemsets:
00331                                 return getMineitemsetsText();
00332                                 default:
00333                                 throw MineruleException( MR_ERROR_MINERULETEXT_PARSING,
00334                                         "Current query is not currently supported by the system "
00335                                                 "(i.e., it is not a MINE RULE nor a MINE SEQUENCE
    query" );
00336                         }
00337                 }
00338                 std::string ParsedMinerule::getMineruleText() const {
00339                         std::string result;
00340                         result =
00341                                 "MINE RULE " + tab_result + " AS " +
00342                                 "SELECT DISTINCT " +
00343                                 getCardsText(bodyCardinalities) + " " + getAttrText(ba) + " AS BODY,"
    +
00344                                 getCardsText(headCardinalities) + " " + getAttrText(ha) + " AS HEAD"
00345                                 ", SUPPORT, CONFIDENCE ";
00346
00347                         if( mc!=NULL )
00348                                 result += "WHERE "+getCondText(mc) + " ";
00349
00350                         result +=
00351                                 "FROM "+tab_source + " "
00352                                         "GROUP BY " + getAttrText(ga) + " ";
00353
00354                         if( gc!=NULL )
00355                                 result += getCondText(gc) + " ";
00356
00357                         if( !ca.empty() ) {
00358                                 result += "CLUSTER BY " + getAttrText(ca) + " ";
00359
00360                                 if( cc!=NULL )
00361                                         result+= "HAVING " +getCondText(cc) + " ";
```

```
00362                         }
00363
00364                         result +=
00365                                 "EXTRACTING RULES WITH SUPPORT:" + Converter(sup).toString() + ","+
00366                                         "CONFIDENCE:"+Converter(conf).toString();
00367
00368                         return result;
00369                 }
00370                 std::string ParsedMinerule::getMineitemsetsText() const {
00371                         std::string result;
00372                         result =
00373                                 "MINE ITEMSET " + tab_result + " AS " +
00374                                         "SELECT DISTINCT " +
00375                                                 getCardsText(bodyCardinalities) + " " +
      getAttrText(ba) + " AS BODY" +
00376                                                         ", SUPPORT ";
00377
00378                         if( mc!=NULL )
00379                                 result += "WHERE "+getCondText(mc) + " ";
00380
00381                         result +=
00382                                 "FROM "+tab_source + " "
00383                                         "GROUP BY " + getAttrText(ga) + " ";
00384
00385                         if( gc!=NULL )
00386                                 result += getCondText(gc) + " ";
00387
00388                         result +=
00389                                 "EXTRACTING ITEMSETS WITH SUPPORT:" + Converter(sup).toString();
00390
00391                         return result;
00392                 }
00393
00394         std::string ParsedMinerule::getBEMText() const{
00395             std::string ris = "";
00396
00397             for(int i = 0 ; i < seq_bem_vect.size(); ++i)
00398                 ris += seq_bem_vect[i]->type +"."+seq_bem_vect[i]->attr+" "+seq_bem_vect[i]->op+"
      "+seq_bem_vect[i]->val+" ";
00399
00400             return ris;
00401         }
00402
00403                 bool ParsedMinerule::hasCrossConditions(const list_OR_node* cond) const {
00404                         MRLog()<<"Checking whether the minerule contains cross predicates" << std::endl;
00405
00406                         const list_OR_node* curr_OR;
00407                         for(curr_OR=cond; curr_OR!=NULL; curr_OR=curr_OR->next) {
00408                                 list_AND_node* curr_AND;
00409                                 for(curr_AND=curr_OR->l_and; curr_AND!=NULL; curr_AND=curr_AND->next)
      {
00410                                         assert(curr_AND->sp!=NULL);
00411                                         MRLog() << curr_AND->sp->val1 << " " << curr_AND->sp->val2 <<
      std::endl;
00412                                         bool hasHead =
00413                                                 strstr(curr_AND->sp->val1,
      "HEAD.")==curr_AND->sp->val1 ||
00414                                                         strstr(curr_AND->sp->val2,
      "HEAD.")==curr_AND->sp->val2;
00415                                         bool hasBody =
00416                                                 strstr(curr_AND->sp->val1,
      "BODY.")==curr_AND->sp->val1 ||
00417                                                         strstr(curr_AND->sp->val2,
      "BODY.")==curr_AND->sp->val2;
00418                                         if( hasHead && hasBody ) {
00419                                                 MRLog() << "Cross predicate found!" << std::endl;
00420                                                 return true;
00421                                         }
00422                                 }
00423                         }
00424
00425                         MRLog() << "Cross predicate not found!" << std::endl;
00426                         return false;
00427                 }
00428
00429
00430         } // namespace minerule
```

7.279 /Users/esposito/Software/minerule/src/Parsers/ParsedPredicate.cpp File Reference

```
#include "minerule/Parsers/ParsedPredicate.hpp"
```

Namespaces

- namespace minerule

7.280 ParsedPredicate.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Parsers/ParsedPredicate.hpp"
00002
00003
00004 namespace minerule {
00005
00006   PredicateBase* ParsedSimplePredicate::newPredicate() const {
00007     return new ParsedPredicate();
00008   }
00009
00010   PredConjunctionBase* ParsedSimplePredicate::newPredConjunction() const {
00011     return new ParsedPredConjunction();
00012   }
00013
00014   PredicateBase* ParsedPredConjunction::newPredicate() const {
00015     return new ParsedPredicate();
00016   }
00017
00018   PredConjunctionBase* ParsedPredConjunction::newPredConjunction() const {
00019     return new ParsedPredConjunction();
00020   }
00021
00022   PredicateBase* ParsedPredicate::newPredicate() const {
00023     return new ParsedPredicate();
00024   }
00025
00026   PredConjunctionBase* ParsedPredicate::newPredConjunction() const {
00027     return new ParsedPredConjunction();
00028   }
00029
00030
00031
00032
00033
00034
00035   list_AND_node* ParsedPredicate::convert_and_list(const PredConjunctionBase& conj) const {
00036     PredConjunctionBase::const_iterator it_and;
00037     list_AND_node* list_and = NULL;
00038     list_AND_node* and_head = NULL;
00039
00040     for( it_and = conj.begin(); it_and!=conj.end(); ++it_and ) {
00041       if(list_and==NULL) {
00042         list_and = (list_AND_node*)malloc( sizeof(list_AND_node) );
00043         list_and->next = NULL;
00044         and_head = list_and;
00045       } else {
00046         list_and->next = (list_AND_node*) malloc( sizeof( list_AND_node ) );
00047         list_and = list_and->next;
00048         list_and->next = NULL;
00049       }
00050
00051       list_and->sp = (simple_pred*) malloc(sizeof(simple_pred));
00052       list_and->sp->val1=strdup((*it_and)->getVal1().c_str());
00053       list_and->sp->op=strdup((*it_and)->getOp().c_str());
00054       list_and->sp->val2=strdup((*it_and)->getVal2().c_str());
00055     }
00056
00057     return and_head;
00058   }
00059
00060
```

```
00061   list_OR_node* ParsedPredicate::convert() const {
00062     PredicateBase::const_iterator it_or;
00063     list_OR_node* list_or = NULL;
00064     list_OR_node* or_head = NULL;
00065
00066     for( it_or = this->begin(); it_or!=this->end(); ++it_or ) {
00067       if(list_or==NULL) {
00068         list_or = (list_OR_node*) malloc( sizeof(list_OR_node) );
00069         list_or->next=NULL;
00070         or_head=list_or;
00071       } else {
00072         list_or->next = (list_OR_node*) malloc( sizeof(list_OR_node) );
00073         list_or = list_or->next;
00074         list_or->next=NULL;
00075       }
00076
00077       list_or->l_and = convert_and_list(**it_or);
00078     }
00079
00080     return or_head;
00081   }
00082
00083
00084
00085 } // namespace
```

7.281 /Users/esposito/Software/minerule/src/Parsers/ParserLibrary.cpp File Reference

```
#include "minerule/Parsers/ParserLibrary.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include <iostream>
```

Namespaces

- namespace minerule

Typedefs

- typedef struct yy_buffer_state * YY_BUFFER_STATE

Functions

- int mrparse ()
- YY_BUFFER_STATE mr_scan_string (const char *)
- void mr_delete_buffer (YY_BUFFER_STATE)
- void minerule::init_mrparser ()
- void minerule::parseMinerule (std::string minerule_text, ParsedMinerule &output)
- ParsedMinerule & minerule::getParserOutputObj ()

Variables

- const FILE * mrout
- ParsedMinerule * minerule::outputObj

7.281.1 Typedef Documentation

7.281.1.1 YY_BUFFER_STATE

```
typedef struct yy_buffer_state* YY_BUFFER_STATE
```

Definition at line 8 of file ParserLibrary.cpp.

7.281.2 Function Documentation

7.281.2.1 mr_delete_buffer()

```
void mr_delete_buffer (
             YY_BUFFER_STATE  )
```

7.281.2.2 mr_scan_string()

```
YY_BUFFER_STATE mr_scan_string (
             const char *  )
```

7.281.2.3 mrparse()

```
int mrparse ( )
```

7.281.3 Variable Documentation

7.281.3.1 mrout

```
const FILE* mrout  [extern]
```

7.282 ParserLibrary.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Parsers/ParserLibrary.hpp"
00002 #include "minerule/Utils/MineruleOptions.hpp"
00003 #include <iostream>
00004
00005 extern int mrparse();
00006 extern const FILE* mrout;
00007
00008 typedef struct yy_buffer_state *YY_BUFFER_STATE;
00009 extern YY_BUFFER_STATE  mr_scan_string(const char*);
00010 extern void mr_delete_buffer(YY_BUFFER_STATE);
00011
00012
00013 namespace minerule {
00014
00015     extern void init_mrparser();
00016     ParsedMinerule * outputObj;
00017
00018 void parseMinerule(std::string minerule_text, ParsedMinerule& output) {
00019     outputObj = &output;
00020
00021     const FILE* logfile =
00022       minerule::MineruleOptions::getSharedOptions().getParsers().getLogFILE();
00023
00024     mrout=logfile;
00025     YY_BUFFER_STATE buf = mr_scan_string(minerule_text.c_str());
00026
00027     init_mrparser();
00028
00029     try {
00030        mrparse();
00031     } catch (MineruleException& m) {
00032      std::string error;
00033       error = std::string(m.unformattedMessage())+ " Original minerule was:"+minerule_text;
00034       throw MineruleException(MR_ERROR_MINERULETEXT_PARSING, error);
00035     }
00036
00037     mr_delete_buffer(buf);
00038
00039     outputObj = NULL;
00040   }
00041
00042   ParsedMinerule& getParserOutputObj() {
00043     assert(outputObj!=NULL);
00044     return *outputObj;
00045   }
00046 }
```

7.283 /Users/esposito/Software/minerule/src/Parsers/PredicateBase.cpp File Reference

```
#include "minerule/Parsers/PredicateBase.hpp"
#include <algorithm>
#include <string>
```

Namespaces

- namespace minerule

7.284 PredicateBase.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Parsers/PredicateBase.hpp"
```

```
00002
00003 #include <algorithm>
00004 #include <string>
00005
00006
00007
00008 namespace minerule {
00009
00010   PredicateBase* SimplePredicateBase::newPredicate() const {
00011     return new PredicateBase();
00012   }
00013
00014   PredConjunctionBase* SimplePredicateBase::newPredConjunction() const {
00015     return new PredConjunctionBase();
00016   }
00017
00018   PredicateBase* PredConjunctionBase::newPredicate() const {
00019     return new PredicateBase();
00020   }
00021
00022   PredConjunctionBase* PredConjunctionBase::newPredConjunction() const {
00023     return new PredConjunctionBase();
00024   }
00025
00026   PredicateBase* PredicateBase::newPredicate() const {
00027     return new PredicateBase();
00028   }
00029
00030  PredConjunctionBase* PredicateBase::newPredConjunction() const {
00031     return new PredConjunctionBase();
00032   }
00033
00034
00035
00036   PredicateBase&
00037   SimplePredicateBase::operator!() const {
00038    std::string nop;
00039     if (op==">") { nop="<="; }
00040     else if (op==">=") { nop="<"; }
00041     else if (op=="<") { nop=">="; }
00042     else if (op=="<=") { nop=">"; }
00043     else if (op=="=") { nop="<>"; }
00044     else if (op=="<>") { nop="="; }
00045     SimplePredicateBase* sp= this->cloneInstance();
00046     PredConjunctionBase* pc= this->newPredConjunction();
00047     PredicateBase*       pb= this->newPredicate();
00048
00049     sp->op = nop;
00050     pc->push_back(sp);
00051     pb->push_back(pc);
00052
00053     return *pb;
00054   }
00055
00056
00057   // -------------------------------------------------------------------------
00058   // PredConjunctionBase
00059   // -------------------------------------------------------------------------
00060
00061
00062   PredConjunctionBase::PredConjunctionBase(const PredConjunctionBase& rhs) {
00063     PredConjunctionBase::const_iterator it;
00064     for(it=rhs.begin(); it!=rhs.end(); ++it) {
00065       this->push_back( (*it)->cloneInstance() );
00066     }
00067   }
00068
00073   PredConjunctionBase::~PredConjunctionBase() {
00074         for (iterator it=begin();it!=end();++it){
00075                     delete *it;
00076         }
00077
00078    }
00079
00084   bool PredConjunctionBase::find(SimplePredicateBase* sp) const {
00085     PredConjunctionBase::const_iterator it;
00086
00087     // Checking if sp is present in this PredConjunctionBase
00088     for (it=this->begin();
00089          it!=this->end() && **it != *sp ;
00090          it++); // note the body of the for is empty
00091
00092     return it!=this->end();
00093   }
00094
00095
00096   PredConjunctionBase&
```

```
00097   PredConjunctionBase::operator&=(const PredConjunctionBase& rhs) {
00098     PredConjunctionBase::const_iterator it;
00099     for(it=rhs.begin(); it!=rhs.end(); ++it) {
00100       if( !this->find(*it) )
00101         this->push_back( (*it)->cloneInstance() );
00102     }
00103
00104     return *this;
00105   }
00106
00107
00108   PredicateBase&
00109   PredConjunctionBase::operator!() const {
00110     PredicateBase* result = newPredicate();
00111     PredConjunctionBase::const_iterator it;
00112     for(it=this->begin(); it!=this->end(); ++it) {
00113       (*result) |= !(**it);
00114     }
00115
00116     return *result;
00117   }
00118
00119   // ----------------------------------------------------------------------
00120   // PredicateBase
00121   // ----------------------------------------------------------------------
00122
00123
00124   PredicateBase& PredicateBase::operator=(const PredicateBase& rhs) {
00125     //freeing used memory
00126     for( iterator it=begin(); it!=end(); it++ ) {
00127       delete *it;
00128     }
00129
00130     // copying rhs contents
00131     for(PredicateBase::const_iterator it=rhs.begin(); it!=rhs.end(); it++) {
00132       push_back( (*it)->cloneInstance() );
00133     }
00134
00135     return *this;
00136   }
00137
00138   PredicateBase::PredicateBase( const PredicateBase& rhs ) {
00139     PredicateBase::const_iterator it;
00140     for(it=rhs.begin(); it!=rhs.end(); it++) {
00141       push_back( (*it)->cloneInstance() );
00142     }
00143   }
00144
00145   PredicateBase::~PredicateBase() {
00146     for( iterator it=begin(); it!=end(); it++ ) {
00147       delete *it;
00148     }
00149   }
00150
00151   PredicateBase& PredicateBase::operator&=(const PredicateBase& p) {
00152     size_t oldSize = this->size();
00153     // making room for the new OR predicates
00154     PredicateBase* thisCopy = this->cloneInstance();
00155     for(size_t i=0; i<p.size()-1; i++) {
00156       *this |= *thisCopy;
00157     }
00158
00159     // building the conjunctions (now there are oldSize*p.size() conj
00160     // in this predicate. The first oldSize ones need to be conjuncted
00161     // with p[0], the second oldSize ones with p[1]... the p.size()'th
00162     // oldSize ones with p[p.size()-1].
00163     for(size_t i=0; i<p.size(); i++) {
00164       for(size_t j=0; j<oldSize; j++) {
00165         assert(j+i*oldSize < this->size() );
00166         //      std::cout << "Pred("<<i<<","<<j<<"):" << *(*this)[j+i*oldSize] << std::endl;
00167         *(*this)[j+i*oldSize] &= *p[i];
00168       }
00169     }
00170
00171     return *this;
00172   }
00173
00174   PredicateBase&
00175   PredicateBase::operator|=(const PredicateBase& p) {
00176     for(PredicateBase::const_iterator it=p.begin(); it!=p.end(); it++) {
00177       push_back( (*it)->cloneInstance() );
00178     }
00179
00180     return *this;
00181   }
00182
00183
```

```
00184   PredicateBase&
00185   PredicateBase::operator!() const {
00186     assert( this->size() != 0 );
00187
00188     PredicateBase* result = newPredicate();
00189     PredicateBase::const_iterator it=this->begin();
00190
00191     (*result) |= !(**it);
00192     ++it;
00193
00194     for(; it!=this->end(); ++it ) {
00195       (*result) &= !(**it);
00196     }
00197
00198     return *result;
00199   }
00200
00201 #if 0
00202   PredicateBase*
00203   PredicateBase::negate_or() const {
00204     PredicateBase* p1; /* Temporary predicate (it will contain the predicate which will be returned)
      */
00205     PredConjunctionBase* pc1;
00206     SimplePredicateBase* tmp;
00207     PredConjunctionBase::iterator it_pc;
00208     PredicateBase::const_iterator it_p;
00209     PredicateBase::iterator it_p2;
00210
00211     it_p = this->begin();
00212
00213     assert(it_p!=this->end());
00214
00215     // Initializing temporary list (only the first time we enter the loop)
00216     p1= this->newInstance();
00217     for (it_pc=(*it_p)->begin();it_pc!=(*it_p)->end(); it_pc++){
00218       pc1= (*it_p)->newInstance();
00219       // aggiungo a p1 il risultato dell'append tra la lista pc1 (predConjunction) e l'elemento *it_pc
      (SimplePredicateBase) negato
00220       pc1->push_back((*it_pc)->negate_predicate());
00221       p1->push_back(pc1);
00222     }
00223
00224     it_p++;
00225
00226     for (;it_p!=this->end() ;it_p++) {
00227       /* Temporary predicate (used at each iteration to hold*/
00228       PredicateBase* p2 = this->newInstance();
00229
00230       for (it_pc=(*it_p)->begin();it_pc!=(*it_p)->end(); it_pc++){//ciclo per ogni elemento di *it_p
00231         tmp=(*it_pc)->negate_predicate();
00232         for (it_p2=p1->begin();it_p2!=p1->end();it_p2++){//ciclo per ogni sottolista di p1
00233           //se l'elemento e' gia presente non lo inserisco
00234           if (!(*it_p2)->find(tmp))
00235             p2->push_back((*it_p2)->copy_and_append(tmp));
00236           else
00237             p2->push_back(*it_p2);
00238         }
00239         delete tmp;
00240         delete p1;
00241         p1=p2;
00242       }
00243     }
00244
00245     return p1;
00246   }
00247
00248 #endif
00249
00250 } // namespace
```

7.285 /Users/esposito/Software/minerule/src/Parsers/Support↩ Measure.cpp File Reference

```
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Parsers/SupportMeasure.hpp"
#include "minerule/Parsers/EvaluationMeasure.hpp"
```

Namespaces

- namespace minerule

7.286 SupportMeasure.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Utils/MineruleOptions.hpp"
00002 #include "minerule/Parsers/SupportMeasure.hpp"
00003 #include "minerule/Parsers/EvaluationMeasure.hpp"
00004
00005
00006
00007 namespace minerule {
00008 /*
00009  * this method return if a frequency of an itemset satisfy the support or not satisfy this measure
00010  **/
00011   bool SupportMeasure::evalSupport(double freq) {
00012     switch (relOp){
00013     case EvaluationMeasure::LessEqual:
00014       return freq <= threshold;
00015         break;
00016     case EvaluationMeasure::Less:
00017       return freq < threshold;
00018         break;
00019     case EvaluationMeasure::Greater:
00020       return freq > threshold;
00021         break;
00022     case EvaluationMeasure::GreaterEqual:
00023       return freq >= threshold;
00024         break;
00025     default: assert(false);
00026     }
00027   }
00028
00029   EvaluationMeasure::MeasureType SupportMeasure::getMeasureType(){
00030     if(relOp == EvaluationMeasure::LessEqual || relOp == EvaluationMeasure::Less)
00031       return EvaluationMeasure::MONOTONE;
00032     else
00033       if(relOp == EvaluationMeasure::GreaterEqual || relOp == EvaluationMeasure::Greater)
00034         return EvaluationMeasure::ANTIMONOTONE;
00035       else assert(false);
00036   }
00037
00038
00039
00040
00041
00042 }//end namespace
```

7.287 /Users/esposito/Software/minerule/src/PredicateUtils/ExpressionNFCoder.cpp File Reference

```
#include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
```

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const EncodedNF &nf)

7.288 ExpressionNFCoder.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
00017
00018 namespace minerule {
00019
00020   /* ========================================================================
00021    * ExpressionNFCoder
00022    * ======================================================================== */
00023
00024   const size_t ExpressionNFCoder::bitsPerCell = (sizeof(EncodingBaseType)«3);
00025
00026
00027   EncodedNF
00028   ExpressionNFCoder::encode(Predicate& ee)
00029     {
00030
00031     cleanBuf();
00032
00033     size_t nvars = ee.getNumVariables();
00034     VarSetEnumerator vse(nvars);
00035
00036     if( nvars > 31 ) {
00037      std::string error =
00038         "ExpressionNFCoder::encode - the current implementation "
00039         "does not support expression with more than 31 variables";
00040
00041       throw MineruleException(MR_ERROR_INTERNAL,error);
00042     }
00043
00044
00045
00046     // We are now going to evaluate the maximal storage required for the
00047     // encoding. Its value can be evaluated as nvars*numSets/bitsPerCell
00048     // Where bitsPerCells is the number of bits that each cell can hold
00049     // (which is sizeof(EncodingBaseType)*8 == sizeof(EncodingBaseType)«3);
00050     // nvars is the number of variables and numSets is the value returned
00051     // by numEnumerations (i.e., 2^numVars).
00052     // In order to avoid overflows in the integer representation of the
00053     // result, we postpone the multiplication after the division. Thanks to
00054     // this little trick, we can manage up to 31 variables (in fact the maximal
00055     // value we can hold in a size_t element is 2^32. Assuming 31 variables we
00056     // end up to the following number: (2^31/2^5) * 31, the division output
00057     // 2^26 which can be safely multiplied by 31 without causing any overflow.
00058     // Note that the calculation has been performed assuming that
00059     // sizeof(EncodingBaseType)*8 == 32, which is true in the current
00060     // implementation but should be checked when the system is ported to other
00061     // platforms or when EncodingBaseType is set to be different from uint.
00062
00063
00064     assert(sizeof(EncodingBaseType)==4);
00065     size_t numCells = (1«nvars) / bitsPerCell;
00066     numCells *= nvars;
00067     if((1«nvars)%bitsPerCell > 0)
00068       numCells++;
00069
00070     // we use malloc instead of new, because we need to use realloc later on
00071     buf = (EncodingBaseType*) calloc( numCells, sizeof(EncodingBaseType));
00072
00073     size_t cellCounter = 0;
00074     size_t elemCounter = 0;
00075
00076     while( vse++ ) {
00077       if(ee.evaluate(*vse) && !filter(*vse)) {
00078         for( size_t i=0; i<(*vse).size(); i++ ) {
00079           if((*vse).getVar(i))
00080             buf[cellCounter] |=  1«elemCounter;
00081           else
00082             buf[cellCounter] &= ~(1«elemCounter);
```

```
00083
00084           if(++elemCounter>=bitsPerCell) {
00085             cellCounter++;
00086             elemCounter=0;
00087           }
00088         }
00089       }
00090     }
00091
00092     buf = (EncodingBaseType*) realloc(buf,sizeof(EncodingBaseType) * (cellCounter+1));
00093     EncodedNF result;
00094     result.encVector = buf;
00095     result.vecSize = cellCounter+1;
00096     result.codingLen = cellCounter*bitsPerCell+elemCounter;
00097     result.numVars = ee.getNumVariables();
00098
00099     return  result;
00100   }
00101
00102
00103   /* ========================================================================
00104    * EncodedNFIterator
00105    * ========================================================================
00106    */
00107
00108   bool
00109   EncodedNFIterator::operator++(int) {
00110     static const size_t baseTypeSize = (sizeof(EncodingBaseType)*8);
00111
00112     if(cellCounter*baseTypeSize+elemCounter>=target.codingLen) {
00113       setOk(false);
00114       return false;
00115     }
00116
00117     setOk(true);
00118
00119     for( size_t i=0; i<target.numVars; i++ ) {
00120       bool elem = target.encVector[cellCounter] & (1<<elemCounter);
00121       currentVarSet.setVar(i, elem);
00122
00123       if(++elemCounter >= baseTypeSize) {
00124         ++cellCounter;
00125         elemCounter=0;
00126       }
00127     }
00128
00129     return true;
00130   }
00131
00132
00133   /* ========================================================================
00134    * EncodedNF
00135    * ========================================================================
00136    */
00137
00138   bool
00139   EncodedNF::operator==(const EncodedNF& nf) const {
00140     if(numVars   != nf.numVars ||
00141        vecSize   != nf.vecSize ||
00142        codingLen != nf.codingLen )
00143       return false;
00144
00145     for(size_t i=0; i<vecSize; i++) {
00146       if( encVector[i]!=nf.encVector[i] )
00147         return false;
00148     }
00149
00150     return true;
00151   }
00152
00153 #if 0
00154   // Old version of getCodesRelationship. This version seems to be buggy.
00155   // The problem is that it compares the two encoded vectors element by element.
00156   // This would be correct if the vectors would represent the last column of the
00157   // truth table of the two predicates, which is false. The vectors represent
00158   // the lines in the table having one in the last column. Hence we could have
00159   // something similar to (I'm assuming 3 variables): v1[0]= 001 101 11 and
00160   // v1[1]= 1 000 000 0 (only the first one represent an interesting bit in
00161   // v[1]), and v2[0]= 101 000 00 (only the first three ones are interesting).
00162   // Here we have that v2 is more specific than v1, since v1 contains all the
00163   // lines that have a one in v2 (i.e., it contains 101).
00164
00165   EncodedNF::CodesRelationship
00166   EncodedNF::getCodesRelationship( const EncodedNF& nf1,
00167                                    const EncodedNF& nf2 ) {
00168     const char* UNEXPECTED_ERRMSG =
00169       "Unexpected relationship state... (this is a bug!)";
```

```
00170
00171
00172     if(nf1.numVars   != nf2.numVars ||
00173        nf1.vecSize   != nf2.vecSize ||
00174        nf1.codingLen != nf2.codingLen )
00175       return Unrelated;
00176
00177     CodesRelationship relationship = Equivalent;
00178
00179     for(size_t i=0; i<nf1.vecSize; i++) {
00180
00181       if( nf1.encVector[i]==nf2.encVector[i] )
00182         continue;
00183
00184       EncodingBaseType codeIntersection =
00185         nf1.encVector[i] & nf2.encVector[i];
00186
00187       if( codeIntersection == nf1.encVector[i] ) {
00188         // nf1.encVector[i] is included in nf2.encVector[i]
00189         switch( relationship ) {
00190         case Equivalent:
00191           relationship = FirstMoreSpecific;
00192           break;
00193         case FirstMoreSpecific:
00194           continue;
00195         case FirstMoreGeneral:
00196           return Unrelated;
00197         default:
00198           throw MineruleException(MR_ERROR_INTERNAL,UNEXPECTED_ERRMSG);
00199         }
00200       } // if codeIntersection==nf1...
00201
00202       if( codeIntersection == nf2.encVector[i] ) {
00203         // nf2.encVector[i] is included in nf1.encVector[i]
00204         switch( relationship ) {
00205         case Equivalent:
00206           relationship = FirstMoreGeneral;
00207           break;
00208         case FirstMoreSpecific:
00209           return Unrelated;
00210         case FirstMoreGeneral:
00211           continue;
00212         default:
00213           throw MineruleException(MR_ERROR_INTERNAL,UNEXPECTED_ERRMSG);
00214         }
00215       } // if codeIntersection==nf2
00216
00217       return Unrelated;
00218     }
00219
00220     return relationship;
00221   }
00222 #endif
00223
00224
00225   std::ostream& operator«(std::ostream& os, const EncodedNF& nf) {
00226     EncodedNFIterator it(nf);
00227     it++;
00228     while(it.ok()) {
00229       os « *it « ".";
00230       it++;
00231     }
00232
00233     return os;
00234   }
00235
00236
00237   EncodedNF::CodesRelationship
00238   EncodedNF::getCodesRelationship( const EncodedNF& nf1,
00239                                    const EncodedNF& nf2 ) {
00240     if(nf1.numVars   != nf2.numVars)
00241       return Unrelated;
00242
00243     CodesRelationship relationship = Equivalent;
00244
00245     EncodedNFIterator it1(nf1);
00246     EncodedNFIterator it2(nf2);
00247     it1++;
00248     it2++;
00249     while(it1.ok() && it2.ok()) {
00250       if(*it1 == *it2) {
00251         it1++;
00252         it2++;
00253         continue;
00254       }
00255
00256       if(*it1 < *it2 ) {
```

```
00257         // in nf2 there is *not* the current value of
00258         // it1. Then nf1 does not contain at least one
00259         // disjunct that is present in nf2, and hence
00260         // this is evidence that nf1 is more general
00261         if(relationship==FirstMoreSpecific)
00262           return Unrelated;
00263         else
00264           relationship=FirstMoreGeneral;
00265
00266         it1++;
00267         continue;
00268       }
00269
00270       if(*it1 > *it2 ) {
00271         // in nf1 there is *not* the current value of
00272         // it2. Then nf1 does not contain at least one
00273         // disjunct that is present in nf2, and hence
00274         // this is evidence that nf2 is more general
00275         if(relationship==FirstMoreGeneral)
00276           return Unrelated;
00277         else
00278           relationship=FirstMoreSpecific;
00279
00280         it2++;
00281       }
00282     }
00283
00284     if( !it1.ok() && !it2.ok() )
00285       return relationship;
00286
00287     if( it1.ok() ) {
00288       // it1.ok() && !it2.ok() suggests
00289       // nf1 is more general
00290       if( relationship==FirstMoreSpecific )
00291         return Unrelated;
00292       else
00293         return FirstMoreGeneral;
00294     }
00295
00296     // !it1.ok() & it2.ok() suggests that
00297     //  nf2 is more general
00298     if( relationship==FirstMoreGeneral )
00299       return Unrelated;
00300     else
00301       return FirstMoreSpecific;
00302   }
00303
00323   size_t EncodedNF::computeHammingDistance(const EncodedNF& nf1,
00324                                            const EncodedNF& nf2) {
00325     assert(nf1.numVars == nf2.numVars);
00326     size_t edt = 0;
00327
00328     EncodedNFIterator it1(nf1);
00329     EncodedNFIterator it2(nf2);
00330     it1++;
00331     it2++;
00332     while(it1.ok() && it2.ok()) {
00333       if( *it1 == *it2 ) {
00334         it1++;
00335         it2++;
00336         continue;
00337       }
00338
00339       if(*it1 < *it2) {
00340         edt++;
00341         it1++;
00342         continue;
00343       }
00344
00345       assert(*it1>*it2);
00346       edt++;
00347       it2++;
00348     }
00349
00350     while(it1.ok()) {
00351       edt++;
00352       it1++;
00353     }
00354
00355     while(it2.ok()) {
00356       edt++;
00357       it2++;
00358     }
00359
00360     return edt;
00361   }
00362
```

```
00363
00364 } // namespace
00365
00366
```

7.289 /Users/esposito/Software/minerule/src/PredicateUtils/HeadBody↩ PredicatesSeparator.cpp File Reference

```
#include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
#include "minerule/Utils/SQLUtils.hpp"
```

Namespaces

- namespace minerule

7.290 HeadBodyPredicatesSeparator.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/HeadBodyPredicatesSeparator.hpp"
00017 #include "minerule/Utils/SQLUtils.hpp"
00018
00019 namespace minerule {
00020
00021         void HeadBodyPredicatesSeparator::setState( ConstraintClass& current, ConstraintClass newone )
      {
00022                 if( current==NO_INFO || current==newone ) {
00023                         current=newone;
00024                         return;
00025                 }
00026
00027                 current=ERROR;
00028                 throw MineruleException(MR_ERROR_INTERNAL,
00029                         "The optimizer found a constraint defined  on both BODY and HEAD!");
00030
00031         }
00032
00033         HeadBodyPredicatesSeparator::ConstraintClass
00034         HeadBodyPredicatesSeparator::setStateAccordinglyToString(std::string& str, ConstraintClass
      state)  {
00035                 std::string substr = str.substr(0,5);
00036                 if(substr=="BODY.")
00037                         setState(state, BODY_CONSTR);
00038                 else if(substr=="HEAD.")
00039                         setState(state, HEAD_CONSTR);
00040                 else if(SQLUtils::isAttribute(str))
00041                         throw MineruleException(MR_ERROR_INTERNAL,
00042                                 "The optimizer found a constraint defined neither on BODY nor on
      HEAD!");
00043                 else return state;
00044
00045   // we need to remove the HEAD. and BODY. part of the string
00046                 SQLUtils::removeHeadBodyFromAttrName( str );
00047                 return state;
```

```
00048         }
00049
00050         void
00051         HeadBodyPredicatesSeparator::updateConstraint( std::string& str, const std::string& v1, const
      std::string& op, const std::string& v2) {
00052                 if(str!="")
00053                         str+=" AND ";
00054
00055                 str+=v1+" "+op+" "+v2;
00056         }
00057
00058         void
00059                 HeadBodyPredicatesSeparator::separate(list_AND_node* l_and,
00060                         std::string& bodyConstraints,
00061         std::string& headConstraints) {
00062                 if( l_and == NULL ) {
00063                         bodyConstraints = "1=1";
00064                         headConstraints = "1=1";
00065                         return;
00066                 }
00067
00068                 while(l_and!=NULL) {
00069                         ConstraintClass state = NO_INFO;
00070                         std::string v1 = l_and->sp->val1;
00071                         std::string v2 = l_and->sp->val2;
00072
00073                         state = setStateAccordinglyToString(v1, state);
00074                         state = setStateAccordinglyToString(v2, state);
00075
00076                         switch(state) {
00077                                 case BODY_CONSTR:
00078                                         updateConstraint(bodyConstraints, v1, l_and->sp->op, v2);
00079                                         break;
00080                                 case HEAD_CONSTR:
00081                                         updateConstraint(headConstraints, v1, l_and->sp->op, v2);
00082                                         break;
00083                                 default:
00084                                         throw MineruleException(MR_ERROR_INTERNAL,
00085                                                 "The optimizer found a constraint defined neither on
      BODY nor on HEAD!");
00086                         };
00087
00088                         l_and=l_and->next;
00089                 }
00090         }
00091
00092 } // namespace
```

7.291 /Users/esposito/Software/minerule/src/PredicateUtils/Interval.cpp File Reference

```
#include "minerule/PredicateUtils/Interval.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/PredicateUtils/Predicate.hpp"
```

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &os, const Interval &i)

7.292 Interval.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/Interval.hpp"
00017 #include "minerule/Utils/MineruleOptions.hpp"
00018 #include "minerule/PredicateUtils/Predicate.hpp"
00019
00020 namespace minerule {
00021
00022   /* ============================================================================
00023    * Interval
00024    * ============================================================================
00025    */
00026
00027   const char* Interval::POSINFTY="+";
00028   const char* Interval::NEGINFTY="-";
00029
00030   std::ostream& operator<<(std::ostream& os, const Interval& i) {
00031     if( i.lwrOpen || i.openPoints.find(i.lwr)!=i.openPoints.end()) os << "("; else os << "[";
00032     os << i.lwr << "," << i.upp;
00033     if( i.uppOpen || i.openPoints.find(i.upp)!=i.openPoints.end()) os << ")"; else os << "]";
00034
00035     return os;
00036   }
00037
00038
00039   Interval::Interval( Operator op, const char* val, SQLUtils::Type t )  {
00040     upp=POSINFTY;
00041     lwr=NEGINFTY;
00042     uppOpen=lwrOpen=true;
00043     setType(t);
00044
00045     switch(op) {
00046     case Less:
00047       upp=val;
00048       break;
00049     case LessEq:
00050       upp=val;
00051       uppOpen=false;
00052       break;
00053     case Grt:
00054       lwr=val;
00055       break;
00056     case GrtEq:
00057       lwr=val;
00058       lwrOpen=false;
00059       break;
00060     case Eq:
00061       lwr=val;
00062       upp=val;
00063       lwrOpen=false;
00064       uppOpen=false;
00065       break;
00066     case NotEq:
00067       openPoints.insert(val);
00068       break;
00069     }
00070   }
00071
00072   Interval::Operator Interval::getOperator( const char* op, bool negateIt ) const {
00073     static const char* errMsg = "Do not know how to handle operator: ";
00074     bool twoCharOp = op[1]!='\0';
00075     if(op[0]=='<') {
00076       if(!twoCharOp)
00077         return negateIt?GrtEq:Less;
00078       else if(op[1]=='>')
00079         return negateIt?Eq:NotEq;
00080       else if(op[1]=='=')
00081         return negateIt?Grt:LessEq;
00082       else throw MineruleException( MR_ERROR_OPTIMIZER_ERROR,
```

```
00083                                   errMsg+ std::string(op) );
00084     } else if( op[0]=='>' ) {
00085       if(!twoCharOp)
00086         return negateIt ? LessEq : Grt;
00087       else if(op[1]=='=')
00088         return negateIt ? Less : GrtEq;
00089       else throw MineruleException(  MR_ERROR_OPTIMIZER_ERROR,
00090                                      errMsg+ std::string(op) );
00091
00092     }   else if( op[0]=='=' ) {
00093       if(twoCharOp)
00094         throw  MineruleException(  MR_ERROR_OPTIMIZER_ERROR,
00095                                    errMsg+ std::string(op) );
00096       else
00097         return negateIt ? NotEq : Eq;
00098     } else throw MineruleException( MR_ERROR_OPTIMIZER_ERROR,
00099                                     errMsg+ std::string(op) );
00100   }
00101
00102   int Interval::compareValues( const char* val1, const char* val2) const {
00103     assert(typeOk);
00104
00105     if(val1==NEGINFTY) {
00106       if(val2==NEGINFTY)
00107         return 0;
00108       else
00109         return -1;
00110     }
00111     if(val1==POSINFTY) {
00112       if(val2==POSINFTY)
00113         return 0;
00114       else
00115         return 1;
00116     }
00117     if(val2==NEGINFTY)
00118       return 1;
00119     if(val2==POSINFTY)
00120       return -1;
00121
00122     if( type==SQLUtils::String) {
00123       return strcmp(val1,val2);
00124     } else if( type==SQLUtils::Numeric ) {
00125       return int(Converter(val1).toDouble()-Converter(val2).toDouble());
00126     } else {
00127       throw MineruleException(MR_ERROR_INTERNAL,
00128                               "SimplePredAnalyzer still does not support SQL types"
00129                               " which differ fromstd::string and numeric");
00130     }
00131   }
00132
00133   const char* Interval::getMaxLwr( Interval& lhs, Interval& rhs, bool& maxIsOpen ) const {
00134     int order = compareValues(lhs.lwr, rhs.lwr);
00135
00136     if( order<0 ) {
00137       maxIsOpen = rhs.lwrOpen;
00138       return rhs.lwr;
00139     } else if(order>0) {
00140       maxIsOpen = lhs.lwrOpen;
00141       return lhs.lwr;
00142     } else {
00143       maxIsOpen = lhs.lwrOpen || rhs.lwrOpen;
00144       return lhs.lwr;
00145     }
00146   }
00147
00148   const char* Interval::getMinUpp( Interval& lhs, Interval& rhs, bool& minIsOpen ) const {
00149     int order = compareValues(lhs.upp, rhs.upp);
00150
00151     if( order<0 ) {
00152       minIsOpen = lhs.uppOpen;
00153       return lhs.upp;
00154     } else if( order>0 ){
00155       minIsOpen = rhs.uppOpen;
00156       return rhs.upp;
00157     } else {
00158       minIsOpen = lhs.uppOpen || rhs.uppOpen;
00159       return lhs.upp;
00160     }
00161   }
00162
00163
00164   const char* Interval::getValue( const char* val1, const char* val2 ) {
00165     if( !SQLUtils::isAttribute(val1) )
00166       return val1;
00167
00168     if( !SQLUtils::isAttribute(val2) )
00169       return val2;
```

```
00170
00171     return NULL;
00172   }
00173
00174   const char* Interval::getAttribute(const char* val1, const char* val2 ) {
00175     if( SQLUtils::isAttribute(val1) )
00176       return val1;
00177
00178     if( SQLUtils::isAttribute(val2) )
00179       return val2;
00180
00181     return NULL;
00182   }
00183
00184
00185
00186   void Interval::update(const list_AND_node* l, bool negateIt) {
00187     assert(l!=NULL && l->sp!=NULL &&
00188            l->sp->val1!=NULL && l->sp->op!=NULL && l->sp->val2!=NULL);
00189
00190     const char* value = getValue(l->sp->val1, l->sp->val2);
00191     if(value==NULL) // this is a cross condition, we just skip the update
00192       return;
00193
00194     Interval i( getOperator( l->sp->op, negateIt ), value, type );
00195     intersect(i);
00196   }
00197
00198   /* ========================================================================
00199    * IntervalChecker
00200    * ========================================================================
00201    */
00202
00203   SQLUtils::Type IntervalChecker::typeForAttribute(const char* attr) {
00204     std::map<const char*, SQLUtils::Type>::const_iterator it =
00205       typesMap.find(attr);
00206
00207     SQLUtils::Type t;
00208     if(it!=typesMap.end()) {
00209       t = it->second;
00210     } else {
00211       t = SQLUtils::getType( MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection(),
00212                              tab_source,
00213                              attr );
00214
00215       typesMap[attr] = t;
00216     }
00217
00218     return t;
00219   }
00220
00221
00222
00223   bool IntervalChecker::impossibleVariableSetting( const VarSet& vset, const list_AND_node* l) {
00224     std::map<const char*, Interval, StringCompare> intervals;
00225     //    std::cout << std::endl;
00226
00227     while( l!=NULL ) {
00228       const char* attr = Interval::getAttribute( l->sp->val1, l->sp->val2 );
00229       if(attr!=NULL) {
00230         SimplePredicate& pred = SimplePredicate::newSimplePredicate(l->sp);
00231         Interval& i=intervals[attr];
00232         i.setType(typeForAttribute(attr));
00233         i.update(l, !vset.getVar(pred.getVarId()));
00234
00235         //      std::cout << l->sp->val1 << l->sp->op << l->sp->val2 << " ";
00236         //      std::cout << "id:" << pred.getVarId() << " v(id):" << vset.getVar(pred.getVarId()) << " ";
00237         //      std::cout << i << std::endl;
00238
00239         if(i.isEmpty())
00240           return true;
00241       };
00242
00243       l=l->next;
00244     }
00245
00246     //    std::cout << intervals["price"] << " ";
00247
00248     return false;
00249   }
00250
00251
00252 } // namespace
```

7.293 /Users/esposito/Software/minerule/src/PredicateUtils/↩ Predicate.cpp File Reference

```
#include "minerule/PredicateUtils/Predicate.hpp"
#include <algorithm>
#include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
```

Namespaces

- namespace minerule

7.294 Predicate.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/Predicate.hpp"
00017
00018 #include <algorithm>
00019 #include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
00020
00021
00022 namespace minerule {
00023
00024   // ------------------------------------------------------------------------
00025   // SimplePredicate
00026   // ------------------------------------------------------------------------
00027
00028   SimplePredicate::PredicatePool SimplePredicate::predicatePool; // definition of static member
00029
00030
00031   SimplePredicate&
00032   SimplePredicate::newSimplePredicate( const simple_pred* pred ) {
00033     SimplePredicate* newpred = new SimplePredicate(pred);
00034
00035     PredicatePool::iterator it=predicatePool.find(newpred);
00036     if( it==predicatePool.end() ) {
00037       predicatePool.insert(newpred);
00038       return *newpred;
00039     } else {
00040       delete newpred;
00041       return **it;
00042     }
00043   }
00044
00045   SimplePredicate&
00046   SimplePredicate::newSimplePredicate(const std::string& val1,
00047                                       const std::string& op,
00048                                       const std::string& val2) {
00049     SimplePredicate* newpred = new SimplePredicate(val1,op,val2);
00050     PredicatePool::iterator it=predicatePool.find(newpred);
00051     if( it==predicatePool.end() ) {
00052       predicatePool.insert(newpred);
00053       return *newpred;
00054     } else {
00055       delete newpred;
00056       return **it;
00057     }
```

```
00058   }
00059
00060
00061   void
00062   SimplePredicate::freeSimplePredicatePool() {
00063     PredicatePool::iterator it;
00064     for( it=predicatePool.begin(); it!=predicatePool.end(); it++ ) {
00065       delete *it;
00066     }
00067
00068     predicatePool.clear();
00069   }
00070
00071   // ------------------------------------------------------------------------
00072   // PredConjunction
00073   // ------------------------------------------------------------------------
00074
00075   PredConjunction::PredConjunction(const list_AND_node* land) {
00076     while(land!=NULL) {
00077       push_back(&SimplePredicate::newSimplePredicate(land->sp));
00078       land = land->next;
00079     }
00080   }
00081
00082   PredConjunction::PredConjunction(const PredConjunction& rhs) {
00083     *this = rhs;
00084   }
00085
00086
00087   bool
00088   PredConjunction::evaluate( const VarSet& vset ) const {
00089     for(const_iterator it=begin(); it!=end(); it++) {
00090       if( vset.getVar((*it)->getVarId())==false )
00091         return false;
00092     }
00093
00094     return true;
00095   }
00096
00097   PredConjunction&
00098   PredConjunction::operator&=(const PredConjunction& p) {
00099     insert( end(), p.begin(), p.end() );
00100     return *this;
00101   }
00102
00103   // ------------------------------------------------------------------------
00104   // Predicate
00105   // ------------------------------------------------------------------------
00106
00107   Predicate::Predicate( const list_OR_node* lor ) :
00108     predList(NULL),
00109     numVariables(0) {
00110     while( lor!=NULL ) {
00111       push_back( new PredConjunction(lor->l_and) );
00112       lor=lor->next;
00113     }
00114   }
00115
00116   Predicate::Predicate( const Predicate& rhs ) :
00117     predList(NULL),
00118     numVariables(0) {
00119     Predicate::const_iterator it;
00120     for(it=rhs.begin(); it!=rhs.end(); it++) {
00121       push_back( new PredConjunction(**it) );
00122     }
00123   }
00124
00125
00126
00127
00128   Predicate::~Predicate() {
00129     for( iterator it=begin(); it!=end(); it++ ) {
00130       delete *it;
00131     }
00132   }
00133
00134   Predicate& Predicate::operator&=(const Predicate& p) {
00135     size_t oldSize = this->size();
00136     // making room for the new OR predicates
00137     Predicate thisCopy(*this);
00138     for(size_t i=0; i<p.size()-1; i++) {
00139       *this |= thisCopy;
00140     }
00141
00142     //    std::cout « "This copy:" « thisCopy « std::endl;
00143     //    std::cout « "P:" « p « std::endl;
00144
```

```
00145     // building the conjunctions (now there are oldSize*p.size() conj
00146     // in this predicate. The first oldSize ones need to be conjuncted
00147     // with p[0], the second oldSize ones with p[1]... the p.size()'th
00148     // oldSize ones with p[p.size()-1].
00149     for(size_t i=0; i<p.size(); i++) {
00150       for(size_t j=0; j<oldSize; j++) {
00151         assert(j+i*oldSize < this->size() );
00152         //      std::cout << "Pred("<<i<<","<<j<<"):" << *(*this)[j+i*oldSize] << std::endl;
00153         *(*this)[j+i*oldSize] &= *p[i];
00154       }
00155     }
00156 
00157     return *this;
00158   }
00159 
00160   Predicate& Predicate::operator|=(const Predicate& p) {
00161     for(Predicate::const_iterator it=p.begin(); it!=p.end(); it++) {
00162       push_back( new PredConjunction( **it ) );
00163     }
00164 
00165     return *this;
00166   }
00167 
00168 
00169   std::set<SimplePredicate*,PtrSimplePredComp>&
00170   Predicate::getPredicateList(bool rebuildSet)  {
00171     if(rebuildSet && predList!=NULL) {
00172       delete predList;
00173       predList=NULL;
00174     }
00175 
00176     if( predList!=NULL )
00177       return *predList;
00178 
00179     predList = new std::set<SimplePredicate*,PtrSimplePredComp>();
00180 
00181     iterator it;
00182     for( it=begin(); it!=end(); it++ ) {
00183       PredConjunction::iterator cit;
00184       for( cit=(*it)->begin(); cit!=(*it)->end(); cit++ ) {
00185         predList->insert(*cit);
00186       }
00187     }
00188 
00189     return *predList;
00190   }
00191 
00192 
00193 
00194   bool
00195   Predicate::evaluate( const VarSet& vset ) const {
00196     if(empty())
00197       return true;
00198 
00199     for( const_iterator it=begin(); it!=end(); it++ ) {
00200       if( (*it)->evaluate(vset)==true )
00201         return true;
00202     }
00203 
00204     return false;
00205   }
00206 } // namespace
```

7.295 /Users/esposito/Software/minerule/src/PredicateUtils/PredicateUtils.cpp File Reference

```
#include "minerule/PredicateUtils/PredicateUtils.hpp"
#include <algorithm>
#include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
```

Namespaces

- namespace minerule

7.296 PredicateUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/PredicateUtils.hpp"
00017
00018 #include <algorithm>
00019 #include "minerule/PredicateUtils/ExpressionNFCoder.hpp"
00020
00021 namespace minerule {
00022
00023   // ------------------------------------------------------------------------
00024   // PredicateUtils
00025   // ------------------------------------------------------------------------
00026
00027   list_AND_node* CountingIterator::new_list_AND_node( const char* v1,
00028                                                       const char* op,
00029                                                       const char* v2,
00030                                                       list_AND_node* oldList ) {
00031     list_AND_node* tmp = new list_AND_node;
00032     tmp->sp = new simple_pred;
00033     tmp->sp->val1 = const_cast<char*>(v1);
00034     tmp->sp->op = const_cast<char*>(op);
00035     tmp->sp->val2 = const_cast<char*>(v2);
00036     tmp->next = oldList;
00037     return tmp;
00038   }
00039
00040   void CountingIterator::delete_list_AND_node(list_AND_node*& l) {
00041     while( l!=NULL ) {
00042       list_AND_node* tmp = l->next;;
00043
00044       delete l->sp;
00045       delete l;
00046
00047       l=tmp;
00048     }
00049   }
00050
00051
00052
00053   // The following returns the semantic relationship between
00054   // two predicates (A and B). The possible answers are:
00055   // FirstMoreGeneral,FirstMoreSpecific, Equivalent, Unrelated
00056   // Their meaning is the following:
00057   // Equivalet:
00058   // two predicates are equivalent if they have the same
00059   // canonical form representation (a predicate is in
00060   // canonical form if it is in disjunctive normal form and
00061   // if every disjunct contains every predicate (being it
00062   // negated or not)). Obviously the canonical form must
00063   // be evaluated on the union of the simple predicates appearing
00064   // in the two predicates.
00065   // FirstMoreGeneral:
00066   // A is more general than B if its canonical form is a superset
00067   // of B's one.
00068   // FirstMoreSpecific:
00069   // A is more specific than B iff B is more general than A.
00070   // Unrelated:
00071   // None of the definitions above holds for A,B.
00072
00073
00074   PredicateUtils::PredicateRelationship
00075   PredicateUtils::getPredicateRelationship(Predicate& p1,
00076                                            Predicate& p2,
00077                                            const std::string& tab_source) {
00078     // The following two lines assign unique variables id
00079     // to elements in p1,p2. It do so setting the variable id
00080     // for each element of the union (through ci) to the position
00081     // in which the predicate appear in the union itself. Note
00082     // that this work only because two equivalent predicates are
```

```
00083     // indeed two references to the same object (look at SimplePredicate
00084     // class to understand why this is true).
00085     size_t counter=0;
00086     list_AND_node* allPreds=NULL;
00087 
00088     CountingIterator ci(counter,allPreds);
00089     std::set<SimplePredicate*, PtrSimplePredComp>& sp1 =
00090       p1.getPredicateList();
00091     std::set<SimplePredicate*, PtrSimplePredComp>& sp2 =
00092       p2.getPredicateList();
00093 
00094     std::set_union( sp1.begin(), sp1.end(),
00095                     sp2.begin(), sp2.end(),
00096                     ci );
00097 
00098 
00099     p1.setNumVariables(counter);
00100     p2.setNumVariables(counter);
00101 
00102     InvalidConfigurationFilter filter( tab_source, allPreds );
00103 
00104     ExpressionNFCoder e1(filter);
00105     ExpressionNFCoder e2(filter);
00106 
00107     PredicateUtils::PredicateRelationship rel=
00108       EncodedNF::getCodesRelationship(e1.encode(p1),e2.encode(p2));
00109 
00110     CountingIterator::delete_list_AND_node(allPreds);
00111     return rel;
00112   }
00113 
00114 
00115   // This is a more specific version of getPredicateRelationship.
00116   // It's presence is motivated because it shorten a little bit
00117   // the notation and because operator== is slightly more efficient
00118   // than getCodesRelationship (In fact the first returns as soon
00119   // as the first not identical element of the NFCode is found, the
00120   // latter has to wait until an incongruence is found).
00121   // Actually the difference between the two should not be noticeable,
00122   // but since this function has been done first, there is no point
00123   // in deleting it...
00124   bool
00125   PredicateUtils::predicatesAreEquivalent(Predicate& p1,
00126                                           Predicate& p2,
00127                                           const std::string& tab_source) {
00128     // The following two lines assign unique variables id
00129     // to elements in p1,p2. It do so setting the variable id
00130     // for each element of the union (through ci) to the position
00131     // in which the predicate appear in the union itself. Note
00132     // that this work only because two equivalent predicates are
00133     // indeed to references to the same object (look at SimplePredicate
00134     // class to understand why this is true).
00135     size_t counter=0;
00136     list_AND_node* allPreds=NULL;
00137 
00138     CountingIterator ci(counter, allPreds);
00139     std::set<SimplePredicate*, PtrSimplePredComp>& sp1 =
00140       p1.getPredicateList();
00141     std::set<SimplePredicate*, PtrSimplePredComp>& sp2 =
00142       p2.getPredicateList();
00143 
00144     std::set_union( sp1.begin(), sp1.end(),
00145                     sp2.begin(), sp2.end(),
00146                     ci );
00147 
00148 
00149     p1.setNumVariables(counter);
00150     p2.setNumVariables(counter);
00151 
00152     InvalidConfigurationFilter filter( tab_source, allPreds );
00153 
00154     ExpressionNFCoder e1(filter);
00155     ExpressionNFCoder e2(filter);
00156 
00157     return e1.encode(p1)==e2.encode(p2);
00158   }
00159 
00160 
00161 
00162 
00163 
00164 } // namespace
```

7.297 /Users/esposito/Software/minerule/src/PredicateUtils/SimplePredAnalyzer.cpp File Reference

```
#include "minerule/PredicateUtils/SimplePredAnalyzer.hpp"
```

Namespaces

- namespace minerule

7.298 SimplePredAnalyzer.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/PredicateUtils/SimplePredAnalyzer.hpp"
00017
00018
00019 namespace minerule {
00020
00021
00022
00023   const char* SimplePredAnalyzer::opRelationTable[6][6]=
00024     //                Yop
00025     //  -------------------------------
00026     {//  <     <=    =      >=    >     <>
00027       {"lbr","lrr","l!!","/!!","/!!","/rr"}, // <   |
00028       {"llr","lbr","ll!","/=!","/!!","/<r"}, // <=  |
00029       {"!!r","!rr","!b!","rr!","r!!","r!r"}, // =   | Xop
00030       {"!!/","!=/","!ll","rbl","rll","r>/"}, // >=  |
00031       {"!!/","!!/","!!l","rrl","rbl","rr/"}, // >   |
00032       {"ll/","l</","l!l","/>l","/ll","/b/"}  // <>  |
00033     };
00034
00035   /*const char* SimplePredAnalyzer::opRelationTable[6][6]=
00036     //                Yop
00037     //  -------------------------------
00038     {//  <     <=    =      >=    >     <>
00039       {"rbl","rll","r!!","/!!","/!!","/ll"}, // <   |
00040       {"rrl","rbl","rr!","/=!","/!!","/<l"}, // <=  |
00041       {"!!l","!ll","!b!","ll!","l!!","l!l"}, // =   | Xop
00042       {"!!/","!=/","!rr","lbr","lrr","l>/"}, // >=  |
00043       {"!!/","!!/","!!r","llr","lbr","ll/"}, // >   |
00044       {"rr/","r</","r!r","/>r","/rr","/b/"}  // <>  |
00045       }; */
00046
00047
00048   // operators are ordered as follows: "< <= = >= > <>"
00049   size_t SimplePredAnalyzer::getOperatorIndex(const char* op)
00050     {
00051
00052     if(op[0]=='<') {
00053       if(op[1]=='\0')
00054         return 0;
00055       if(op[1]=='=')
00056         return 1;
00057       if(op[1]=='>')
00058         return 5;
00059
00060       throw MineruleException(MR_ERROR_INTERNAL, (std::string) "Operator "+op+" not recognized");
```

```
00061     } else if(op[0]=='>') {
00062       if(op[1]=='\0')
00063         return 4;
00064       if(op[1]=='=')
00065         return 3;
00066 
00067       throw MineruleException(MR_ERROR_INTERNAL,(std::string) "Operator"+op+" not recognized");
00068     } else  if(op[0]=='=') {
00069       if(op[1]=='\0')
00070         return 2;
00071     }
00072 
00073     throw MineruleException(MR_ERROR_INTERNAL,(std::string) "Operator"+op+" not recognized");
00074   }
00075 
00076   size_t SimplePredAnalyzer::getValuesRelationshipIndex(const char* value1,
00077                                                         const char* value2,
00078                                                         SQLUtils::Type type)
00079     {
00080 
00081     double order;
00082     if( type==SQLUtils::String) {
00083       order=strcmp(value1,value2);
00084     } else if( type==SQLUtils::Numeric ) {
00085       order =
00086         Converter(value1).toDouble()-Converter(value2).toDouble();
00087     } else {
00088       throw MineruleException(MR_ERROR_INTERNAL,
00089                   "SimplePredAnalyzer still does not support SQL types"
00090                   " which differ fromstd::string and numeric");
00091     }
00092 
00093     /*
00094      * It seems that the table is built using the order x>y x=y x<y...
00095      */
00096     if( order<0 )
00097       return 2;
00098     else if(order==0)
00099       return 1;
00100     else return 0;
00101   }
00102 
00103   bool
00104   SimplePredAnalyzer::isAttrOpValuePredicate(simple_pred* X,
00105                                              char*& attr,
00106                                              char*& value,
00107                                              bool& reverseOp) {
00108     if(SQLUtils::isAttribute(X->val1)) {
00109       attr=X->val1;
00110       value=X->val2;
00111       reverseOp=false;
00112     } else if(SQLUtils::isAttribute(X->val2)) {
00113       attr=X->val2;
00114       value=X->val1;
00115       reverseOp=true;
00116     }
00117 
00118     if(attr==NULL || SQLUtils::isAttribute(value))
00119       return false;
00120     else
00121       return true;
00122   }
00123 
00124   char
00125   SimplePredAnalyzer::getRelation(const char* Xop,
00126                                   const char* Xvalue,
00127                                   const char* Yop,
00128                                   const char* Yvalue,
00129                                   SQLUtils::Type type)
00130     {
00131 
00132     size_t XopIndex = getOperatorIndex(Xop);
00133     size_t YopIndex = getOperatorIndex(Yop);
00134     size_t XYvalsIndex = getValuesRelationshipIndex(Xvalue,Yvalue,type);
00135     return opRelationTable[XopIndex][YopIndex][XYvalsIndex];
00136   }
00137 
00138 } // namespace minerule
```

7.299 /Users/esposito/Software/minerule/src/Programs/↩ Minerule/minerule.cpp File Reference

```
#include <string>
#include <iostream>
#include <unistd.h>
#include <assert.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/MineruleErrors.hpp"
#include "minerule/Utils/StringUtils.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
```

Functions

- const std::vector< std::string > & knownCommands ()
- std::string buildPath (std::string cmdName)
- void execSubCommand (std::string cmd, int argc, char **argv, std::string cmdPath)
- void printMRHelp ()
- std::string expandSubCommand (std::string cmdPrefix)
- void checkSubCommands (int argc, char **argv)
- int main (int argc, char *argv[])

7.299.1 Function Documentation

7.299.1.1 buildPath()

```
std::string buildPath (
            std::string cmdName )
```

Definition at line 46 of file minerule.cpp.

```
00046                                       {
00047         size_t last_slash_pos = cmdName.find_last_of("/");
00048         if(last_slash_pos == std::string::npos)
00049                 return "";
00050
00051         return cmdName.substr(0, last_slash_pos+1);
00052 }
```

7.299.1.2 checkSubCommands()

```
void checkSubCommands (
            int argc,
            char ** argv )
```

Definition at line 102 of file minerule.cpp.

```
00102                                           {
00103         if(argc < 2 || std::string(argv[1]) == "-h") {
00104                 printMRHelp();
00105                 exit(0);
00106         }
00107
00108         std::string subCmd( expandSubCommand(argv[1]) );
00109         const std::vector<std::string>& cmds = knownCommands();
00110
00111         if( std::find( cmds.begin(), cmds.end(), subCmd.c_str() ) != cmds.end() ) {
00112                 execSubCommand(subCmd, argc-2, &argv[2], buildPath(argv[0]));
00113         }
00114
00115         throw MineruleException( MR_ERROR_OPTION_PARSING, "Command not recognized, see help for a list
      of available commands." );
00116 }
```

7.299.1.3 execSubCommand()

```
void execSubCommand (
            std::string cmd,
            int argc,
            char ** argv,
            std::string cmdPath )
```

Definition at line 55 of file minerule.cpp.

```
00055                                                                                                   {
00056         const char** cmdArgv = new char const*[argc+2];
00057         assert(cmdArgv != NULL );
00058         std::string executableName = cmdPath + std::string("mr-")+cmd;
00059
00060         cmdArgv[0] = executableName.c_str();
00061         for(size_t i=1; i<argc+1; ++i) {
00062                 cmdArgv[i] = argv[i-1];
00063         }
00064         cmdArgv[argc+1] = NULL;
00065
00066         errno = 0;
00067         execvp(executableName.c_str(), (char* const*) cmdArgv);
00068         if(errno!=0) {
00069                 MRErr() << "Could not launch command:" + executableName << std::endl
00070                         << "       The system reported error: " << strerror(errno) << std::endl;
00071                 exit(1);
00072         }
00073 }
```

7.299.1.4 expandSubCommand()

```
std::string expandSubCommand (
            std::string cmdPrefix )
```

Definition at line 90 of file minerule.cpp.

```
00090                                                {
00091         const std::vector<std::string>& cmds  = knownCommands();
00092         std::vector<std::string>::const_iterator it;
00093         for(it=cmds.begin(); it!=cmds.end(); ++it) {
00094                 if( it->find(cmdPrefix) == 0 ) // cmdPrefix is a prefix of commands[i]
00095                         return *it;
00096         }
00097
00098         return cmdPrefix;
00099 }
```

7.299.1.5 knownCommands()

```
const std::vector< std::string > & knownCommands ( )
```

Definition at line 32 of file minerule.cpp.

```
00032                                           {
00033         static std::vector<std::string> _knownCommands;
00034         if( _knownCommands.empty() ) {
00035                 _knownCommands.push_back("print");
00036                 _knownCommands.push_back("catalogue");
00037                 _knownCommands.push_back("defaults");
00038                 _knownCommands.push_back("match");
00039                 _knownCommands.push_back("query");
00040         }
00041
00042         return _knownCommands;
00043 }
```

7.299.1.6 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 118 of file minerule.cpp.

```
00119 {
00120         try {
00121                 checkSubCommands(argc, argv);
00122         } catch (MineruleException& e) {
00123                 MRErr() « "Minerule Exception:" « e.what() « std::endl;
00124                 exit( e.getErrorCode() );
00125         } catch (std::exception& e) {
00126                 MRErr() « "Exception:" « e.what() « std::endl;
00127                 exit( MR_ERROR_UNKNOWN );
00128         }
00129         catch(...) {
00130                 MRErr() « "An unknown exception has been thrown..." « std::endl;
00131                 exit( MR_ERROR_UNKNOWN );
00132         }
00133
00134         return 0;
00135 }
```

7.299.1.7 printMRHelp()

```
void printMRHelp ( )
```

Definition at line 75 of file minerule.cpp.

```
00075                    {
00076         std::cout « StringUtils::toBold("Usage:") « std::endl
00077                 « StringUtils::toBold("   mr -h                              ") « " - shows this message."
      « std::endl
00078                 « StringUtils::toBold("   mr <sub-command> [-h] [options]")
00079                         « " - executes a sub-command. Use the -h option for further information." «
      std::endl
00080                 « std::endl
00081                 « StringUtils::toBold("Available sub-commands are: ")
00082                 « StringUtils::join(knownCommands(), ", ") « "."
00083                 « std::endl
00084                 « StringUtils::toBold("Note: ") « "Sub-commands can be shortened at will"
00085                 « ", e.g.: 'mr pr' is equivalent to 'mr print'" « std::endl;
00086 }
```

7.300 minerule.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include<string>
00017 #include<iostream>
00018 #include<unistd.h>
00019 #include<assert.h>
00020 #include<errno.h>
00021 #include<string.h>
00022 #include<stdlib.h>
00023 #include <algorithm>
00024
00025 #include "minerule/Utils/MineruleException.hpp"
00026 #include "minerule/Utils/MineruleErrors.hpp"
00027 #include "minerule/Utils/StringUtils.hpp"
00028 #include "minerule/Utils/MineruleLogs.hpp"
00029
00030 using namespace minerule;
00031
00032 const std::vector<std::string>& knownCommands() {
00033         static std::vector<std::string> _knownCommands;
00034         if( _knownCommands.empty() ) {
00035                 _knownCommands.push_back("print");
00036                 _knownCommands.push_back("catalogue");
00037                 _knownCommands.push_back("defaults");
00038                 _knownCommands.push_back("match");
00039                 _knownCommands.push_back("query");
00040         }
00041
00042         return _knownCommands;
00043 }
00044
00045
00046 std::string buildPath(std::string cmdName) {
00047         size_t last_slash_pos = cmdName.find_last_of("/");
00048         if(last_slash_pos == std::string::npos)
00049                 return "";
00050
00051         return cmdName.substr(0, last_slash_pos+1);
00052 }
00053
00054
00055 void execSubCommand( std::string cmd, int argc, char** argv, std::string cmdPath ) {
00056         const char** cmdArgv = new char const*[argc+2];
00057         assert(cmdArgv != NULL );
00058         std::string executableName = cmdPath + std::string("mr-")+cmd;
00059
00060         cmdArgv[0] = executableName.c_str();
00061         for(size_t i=1; i<argc+1; ++i) {
00062                 cmdArgv[i] = argv[i-1];
00063         }
00064         cmdArgv[argc+1] = NULL;
00065
00066         errno = 0;
00067         execvp(executableName.c_str(), (char* const*) cmdArgv);
00068         if(errno!=0) {
00069                 MRErr() « "Could not launch command:" + executableName « std::endl
00070                         « "       The system reported error: " « strerror(errno) « std::endl;
00071                 exit(1);
00072         }
00073 }
00074
00075 void printMRHelp() {
00076         std::cout « StringUtils::toBold("Usage:") « std::endl
00077                 « StringUtils::toBold("   mr -h                          ") « " - shows this message."
      « std::endl
00078                 « StringUtils::toBold("   mr <sub-command> [-h] [options]")
00079                         « " - executes a sub-command. Use the -h option for further information." «
      std::endl
00080                 « std::endl
```

```
00081                 « StringUtils::toBold("Available sub-commands are: ")
00082                 « StringUtils::join(knownCommands(), ", ") « "."
00083                 « std::endl
00084                 « StringUtils::toBold("Note: ") « "Sub-commands can be shortened at will"
00085                 « ", e.g.: 'mr pr' is equivalent to 'mr print'" « std::endl;
00086 }
00087
00088
00089
00090 std::string expandSubCommand(std::string cmdPrefix) {
00091         const std::vector<std::string>& cmds  = knownCommands();
00092         std::vector<std::string>::const_iterator it;
00093         for(it=cmds.begin(); it!=cmds.end(); ++it) {
00094                 if( it->find(cmdPrefix) == 0 ) // cmdPrefix is a prefix of commands[i]
00095                         return *it;
00096         }
00097
00098         return cmdPrefix;
00099 }
00100
00101
00102 void checkSubCommands(int argc, char** argv) {
00103         if(argc < 2 || std::string(argv[1]) == "-h") {
00104                 printMRHelp();
00105                 exit(0);
00106         }
00107
00108         std::string subCmd( expandSubCommand(argv[1]) );
00109         const std::vector<std::string>& cmds = knownCommands();
00110
00111         if( std::find( cmds.begin(), cmds.end(), subCmd.c_str() ) != cmds.end() ) {
00112                 execSubCommand(subCmd, argc-2, &argv[2], buildPath(argv[0]));
00113         }
00114
00115         throw MineruleException( MR_ERROR_OPTION_PARSING, "Command not recognized, see help for a list
      of available commands." );
00116 }
00117
00118 int main (int argc, char *argv[])
00119 {
00120         try {
00121                 checkSubCommands(argc, argv);
00122         } catch (MineruleException& e) {
00123                 MRErr() « "Minerule Exception:" « e.what() « std::endl;
00124                 exit( e.getErrorCode() );
00125         } catch (std::exception& e) {
00126                 MRErr() « "Exception:" « e.what() « std::endl;
00127                 exit( MR_ERROR_UNKNOWN );
00128         }
00129         catch(...) {
00130                 MRErr() « "An unknown exception has been thrown..." « std::endl;
00131                 exit( MR_ERROR_UNKNOWN );
00132         }
00133
00134         return 0;
00135 }
00136
00137
```

7.301 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ ErrorCodes.hpp File Reference

Namespaces

- namespace mrc

Enumerations

- enum mrc::Results {
  mrc::SUCCESS =0 , mrc::NOTHING_TO_DO , mrc::QUERY_NAME_FOUND , mrc::QUERY_NAME_NOT_FOUND
  ,
  mrc::CATALOGUE_NOT_INSTALLED , mrc::ERROR_OPTION_PARSING , mrc::ERROR_CANNOT_OPEN_OPTION_FILE
  , mrc::ERROR_EXCEPTION_THROWN }

7.302 ErrorCodes.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef ERRORCODES_H_307OH17
00017 #define ERRORCODES_H_307OH17
00018
00019 namespace mrc {
00020
00021         typedef enum {
00022           SUCCESS=0,
00023           NOTHING_TO_DO,
00024           QUERY_NAME_FOUND,
00025           QUERY_NAME_NOT_FOUND,
00026           CATALOGUE_NOT_INSTALLED,
00027           ERROR_OPTION_PARSING,
00028           ERROR_CANNOT_OPEN_OPTION_FILE,
00029           ERROR_EXCEPTION_THROWN,
00030         } Results;
00031
00032 }
00033 #endif /* end of include guard: ERRORCODES_H_307OH17 */
```

7.303 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ Exception.hpp File Reference

```
#include "ErrorCodes.hpp"
#include <exception>
#include <string>
```

Data Structures

- class mrc::Exception

Namespaces

- namespace mrc

Macros

- #define _NOEXCEPT throw()

7.303.1 Macro Definition Documentation

7.303.1.1 _NOEXCEPT

```
#define _NOEXCEPT throw()
```

Definition at line 24 of file Exception.hpp.

7.304 Exception.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef EXCEPTION_H_U3CYE3H3
00017 #define EXCEPTION_H_U3CYE3H3
00018
00019 #include "ErrorCodes.hpp"
00020 #include <exception>
00021 #include <string>
00022
00023 #ifndef _NOEXCEPT
00024 #define _NOEXCEPT throw()
00025 #endif
00026
00027 namespace mrc {
00028
00029         class Exception : public std::exception {
00030                   mrc::Results result;
00031                   std::string errmsg;
00032         public:
00033                   Exception(mrc::Results res,
00034                           const std::string& msg) :  result(res),errmsg(msg) { }
00035
00036                   virtual ~Exception() _NOEXCEPT {};
00037
00038                   virtual const char* what() const _NOEXCEPT {
00039                           return errmsg.c_str();
00040                   }
00041
00042                   virtual mrc::Results getResultID() const {
00043                           return result;
00044                   }
00045         };
00046
00047 }
00048 #endif /* end of include guard: EXCEPTION_H_U3CYE3H3 */
```

7.305 /Users/esposito/Software/minerule/src/Programs/↩ MRCatalogue/mrcatalogue.cpp File Reference

```
#include "minerule/Utils/MineruleOptions.hpp"
#include "ErrorCodes.hpp"
#include "MRCUtils.hpp"
#include "Options.hpp"
```

Functions

- int main (int argc, char **argv)

7.305.1 Function Documentation

7.305.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 25 of file mrcatalogue.cpp.

```
00025                                   {
00026         if( argc < 2 ) {
00027                 mrc::printHelp(argc, argv);
00028                 exit(0);
00029         }
00030
00031         int resultVal;
00032         try {
00033
00034                 MineruleOptions& mr = MineruleOptions::getSharedOptions();
00035                 mrc::Options options;
00036                 mrc::parseOptions(argc, argv, mr, options);
00037
00038                 resultVal=mrc::performCommands(options);
00039         } catch (mrc::Exception& e) {
00040                 resultVal = e.getResultID();
00041                 if(resultVal!=mrc::NOTHING_TO_DO)
00042                         std::cout << e.what() << std::endl;
00043         } catch (std::exception& e) {
00044                 std::cout << "An error occurred, the error msg is:" << std::endl;
00045                 std::cout << e.what() << std::endl;
00046                 resultVal=mrc::ERROR_EXCEPTION_THROWN;
00047         }
00048
00049         return resultVal;
00050 }
```

7.306 mrcatalogue.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/MineruleOptions.hpp"
00017
00018 #include "ErrorCodes.hpp"
00019 #include "MRCUtils.hpp"
00020 #include "Options.hpp"
00021
00022 using namespace minerule;
00023
00024 int
```

```
00025 main(int argc, char** argv) {
00026         if( argc < 2 ) {
00027                 mrc::printHelp(argc, argv);
00028                 exit(0);
00029         }
00030
00031         int resultVal;
00032         try {
00033
00034                 MineruleOptions& mr = MineruleOptions::getSharedOptions();
00035                 mrc::Options options;
00036                 mrc::parseOptions(argc, argv, mr, options);
00037
00038                 resultVal=mrc::performCommands(options);
00039         } catch (mrc::Exception& e) {
00040                 resultVal = e.getResultID();
00041                 if(resultVal!=mrc::NOTHING_TO_DO)
00042                         std::cout << e.what() << std::endl;
00043         } catch (std::exception& e) {
00044                 std::cout << "An error occurred, the error msg is:" << std::endl;
00045                 std::cout << e.what() << std::endl;
00046                 resultVal=mrc::ERROR_EXCEPTION_THROWN;
00047         }
00048
00049         return resultVal;
00050 }
```

7.307 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ MRCUtils.cpp File Reference

```
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
#include <iostream>
#include <assert.h>
#include <string>
#include <algorithm>
#include <getopt.h>
#include "MRCUtils.hpp"
#include "Printer.hpp"
#include "minerule/Utils/FileUtils.hpp"
#include "minerule/Utils/StringUtils.hpp"
```

Namespaces

- namespace mrc

Functions

- void mrc::doLoadOptions (const char *fname, minerule::MineruleOptions &opt)
- void mrc::parseOptions (int argc, char **argv, minerule::MineruleOptions &mopt, Options &popt)
- void mrc::printVersion ()
- void mrc::printHelp (int argc, char **argv)
- int mrc::checkQueryName (const Options &options)
- void mrc::printMRQueryList (const Options &options)
- void mrc::deleteQuery (const Options &options)
- Results mrc::checkCatalogue ()
- Results mrc::installCatalogue ()
- Results mrc::uninstallCatalogue ()
- Results mrc::addDerivedQuery (const Options &options)
- int mrc::performCommands (Options &options)

7.308 MRCUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/mrdb/ResultSet.hpp"
00017 #include "minerule/mrdb/Connection.hpp"
00018 #include "minerule/mrdb/Statement.hpp"
00019 #include "minerule/mrdb/PreparedStatement.hpp"
00020
00021 #include <iostream>
00022 #include <assert.h>
00023 #include <string>
00024 #include <algorithm>
00025 #include <getopt.h>
00026
00027 #include "MRCUtils.hpp"
00028 #include "Printer.hpp"
00029 #include "minerule/Utils/FileUtils.hpp"
00030 #include "minerule/Utils/StringUtils.hpp"
00031
00032
00033 namespace mrc {
00034         // ------------------------------------------------------------------------------
00035         // OPTION HANDLING
00036         // ------------------------------------------------------------------------------
00037
00038         void doLoadOptions(const char* fname, minerule::MineruleOptions& opt) {
00039
00040                 if( minerule::FileUtils::fileExists(fname) ) {
00041                         opt.readFromFile(fname);
00042                 } else {
00043                         std::string errstr;
00044                         if(errno==0) {
00045                                 errstr = "Not a regular file!";
00046                         } else {
00047                                 errstr = strerror(errno);
00048                         }
00049
00050                         throw Exception(mrc::ERROR_CANNOT_OPEN_OPTION_FILE,
00051                                 std::string("Could not open file:") + fname + ". The reason is:" +
      errstr);
00052                 }
00053         }
00054
00055         void parseOptions(int argc, char** argv, minerule::MineruleOptions& mopt, Options& popt) {
00056                 const char* optstr = "cCIUhf:lF:n:d:va:";
00057
00058                 int opt;
00059                 bool didLoadOptions=false;
00060                 while((opt=getopt(argc, argv, optstr))!=-1) {
00061                         switch( opt ) {
00062                         case 'c':
00063                                 minerule::StringUtils::setColorsEnabled(false);
00064                                 break;
00065                         case 'C':
00066                                 popt.setCheckCatalogue();
00067                                 break;
00068                         case 'U':
00069                                 popt.setUninstallCatalogue();
00070                                 break;
00071                         case 'I':
00072                                 popt.setInstallCatalogue();
00073                                 break;
00074                         case 'f':
00075                                 doLoadOptions(optarg, mopt);
00076                                 didLoadOptions=true;
00077                                 break;
00078                         case 'h':
00079                                 printHelp(argc, argv);
00080                                 break;
00081                         case 'F':
```

```
00082                                   popt.setListFormat(optarg);
00083                                   break;
00084                           case 'l':
00085                                   popt.setShowList();
00086                                   break;
00087                           case 'n':
00088                                   popt.setSearchQry();
00089                                   popt.setSearchParam(Options::QryName, optarg);
00090                                   break;
00091                           case 'd':
00092                                   popt.setDeleteQry();
00093                                   popt.setSearchParam(Options::QryName, optarg);
00094                                   break;
00095                           case 'a': {
00096                                           std::vector<std::string> queries =
      minerule::StringUtils::split(optarg, ",");
00097                                           if( queries.size() != 2 ) {
00098                                                   throw Exception( mrc::ERROR_OPTION_PARSING, "Wrong
      format for -a options, you should pass the original query name and the derived query name separated
      by a ',' (no spaces)");
00099                                           }
00100                                          popt.setAddDerivedQuery(queries[0], queries[1]);
00101                                  }
00102                                  break;
00103                          case 'v':
00104                                  printVersion();
00105                                  exit(0);
00106                          case '?':
00107                          default:
00108                                  throw Exception( mrc::ERROR_OPTION_PARSING, "Error while parsing
      options, use -h to show" " the help message" );
00109                          }
00110                 }
00111                 if( !didLoadOptions  ) {
00112                         if(
      !minerule::FileUtils::fileExists(minerule::MineruleOptions::DEFAULT_FILE_NAME) )
00113                                 throw Exception( mrc::ERROR_OPTION_PARSING,"You should specify at
      least one -f option or provide ./"+minerule::MineruleOptions::DEFAULT_FILE_NAME);
00114                         else
00115                                 doLoadOptions(minerule::MineruleOptions::DEFAULT_FILE_NAME.c_str(),
      mopt);
00116                 }
00117         }
00118
00119         void printVersion() {
00120                 std::cout « "MRCatalogue v:" « MR_VERSION « std::endl;
00121         }
00122
00123         // ------------------------------------------------------------------------------
00124         // ACTIONS
00125         // ------------------------------------------------------------------------------
00126
00127         void printHelp(int argc, char** argv) {
00128                 using namespace minerule;
00129                 std::cout « StringUtils::toBold("Usage:") « std::endl
00130                         « "   " « StringUtils::toBold(argv[0]) « " [-v] [-h] [-f  <optionfile>] [-C]
      [-I] [-U] [-n <queryname>] [-l] [-d] [-a <ori>,<der>][-c] [-F <formatSpecs>]" « std::endl
00131                         « std::endl « std::endl
00132                         « "Handles the minerule catalogue." « std::endl
00133                         « std::endl
00134                         « StringUtils::toBold("Program Information") « std::endl
00135                         « StringUtils::toBold("-v") « " - Output the version message and exits." «
      std::endl
00136                         « StringUtils::toBold("-h") « " - Output this message and returns
      NOTHING_TO_DO." « std::endl
00137                         « std::endl
00138                         « StringUtils::toBold("Option Handling") « std::endl
00139                         « StringUtils::toBold("-f") « " - file name of the option file (if omitted the
      program will)" « std::endl
00140                         « "     look for a file named 'option.txt' in the current directory" «
      std::endl
00141                         « std::endl
00142                         « StringUtils::toBold("Installation") « std::endl
00143                         « StringUtils::toBold("-C") « " - Checks if the optimizer catalogue is
      installed correctly and exits." «std::endl
00144                         « StringUtils::toBold("-I") « " - Installs the optimizer catalogue and exits."
      «std::endl
00145                         « StringUtils::toBold("-U") « " - Uninstall the catalogue (be careful, this
      cannot be undone!)." « std::endl
00146                         « "     dbms must be either 'mysql' or 'postgres'." « std::endl
00147                         « std::endl
00148                         « StringUtils::toBold("Dealing with Catalogue Entries") « std::endl
00149                         « StringUtils::toBold("-n") « " - Look in the catalogue for the specified
      query." « std::endl
00150                         « "     it returns QUERY_NAME_FOUND, or QUERY_NAME_NOT_FOUND" « std::endl
00151                         « "     accordingly to whether the query could be found."«std::endl
00152                         « StringUtils::toBold("-l") « " - Print the list of already executed queries,
```

```
returns" << std::endl
00153                         << "     SUCCESS upon completion." << std::endl
00154                         << StringUtils::toBold("-d") << " - Delete the given minerule from the system
(notice that" << std::endl
00155                         << "     the safety options in the option file must be setted" << std::endl
00156                         << "     in such a way to allow the deletion. " << std::endl
00157                         << StringUtils::toBold("-a") << " - Adds a derived result set. You must provide
the name <ori> of the original" << std::endl
00158                         << "     minerule and the name <der> of the derived result. The database must
then" << std::endl
00159                         << "     contain a table named <der> and two additional tables <der>_body_elems
and " << std::endl
00160                         << "     <der>_head_elems" << std::endl
00161                         << std::endl
00162                         << StringUtils::toBold("Output Handling") << std::endl
00163                         << StringUtils::toBold("-c") << " - Disables color output." << std::endl
00164                         << StringUtils::toBold("-F") << " - Format specifiers for printing the list of
queries (-l)" <<std::endl
00165                         << "     Valid specifiers: s - Print the size of the result set" << std::endl
00166                         << "                       t - Print the text of the original mr" <<std::endl
00167                         << "                       r - Print the result set table names" << std::endl
00168                         << "     the default is ", i.e.: print onlty the qry name." << std::endl
00169                         << std::endl
00170                         << StringUtils::toBold("Return Values:") << std::endl
00171                         << "      SUCCESS = " << mrc::SUCCESS << std::endl
00172                         << "      NOTHING_TO_DO = " << mrc::NOTHING_TO_DO << std::endl
00173                         << "      QUERY_NAME_FOUND = " << mrc::QUERY_NAME_FOUND << std::endl
00174                         << "      CATALOGUE_NOT_INSTALLED = " << mrc::CATALOGUE_NOT_INSTALLED <<
std::endl
00175                         << "      QUERY_NAME_NOT_FOUND = " << mrc::QUERY_NAME_NOT_FOUND << std::endl
00176                         << "      ERROR_OPTION_PARSING = " << mrc::ERROR_OPTION_PARSING << std::endl
00177

                                                    << "      ERROR_CANNOT_OPEN_OPTION_FILE = " <<
mrc::ERROR_CANNOT_OPEN_OPTION_FILE <<std::endl;
00178         }
00179
00180         int checkQueryName(const Options& options) {
00181                 const std::string& tablename =
00182                         options.getSearchParam(Options::QryName);
00183
00184                 try {
00185                         Printer printer(std::cout, options);
00186                         minerule::CatalogueInfo info;
00187                         minerule::OptimizerCatalogue::getMRQueryInfo(tablename,info,
options.getListFormat().size);
00188                         printer.print(info);
00189                 } catch (minerule::MineruleException& mr) {
00190                         std::cout << "The query you specified is NOT present in the catalogue." <<
std::endl;
00191                         return mrc::QUERY_NAME_NOT_FOUND;
00192                 }
00193
00194                 return mrc::QUERY_NAME_FOUND;
00195         }
00196
00197         void printMRQueryList(const Options& options) {
00198                 Printer printer(std::cout, options);
00199                 std::vector<minerule::CatalogueInfo> mrqlist;
00200                 minerule::OptimizerCatalogue::getMRQueryInfos(mrqlist, options.getListFormat().size );
00201
00202                 printer.print(mrqlist);
00203         }
00204
00205         void deleteQuery(const Options& options) {
00206                 if( checkQueryName(options)==mrc::QUERY_NAME_FOUND )
00207                         minerule::OptimizerCatalogue::deleteMinerule(
options.getSearchParam(Options::QryName) );
00208         }
00209
00210         Results checkCatalogue()  {
00211                 if(minerule::OptimizerCatalogue::checkInstallation()) {
00212                         minerule::MRLog() << "The catalogue appears to be installed properly." <<
std::endl;
00213                         return SUCCESS;
00214                 } else {
00215                         minerule::MRLog() << "The catalogue is " << minerule::StringUtils::toRed("NOT")
<<" installed" << std::endl;
00216                         exit(CATALOGUE_NOT_INSTALLED);
00217                 }
00218         }
00219
00220         Results installCatalogue() {
00221                 minerule::MRLog() << "Checking current installation status." << std::endl;
00222                 if(minerule::OptimizerCatalogue::checkInstallation()) {
00223                         minerule::MRLog() << minerule::StringUtils::toBold("Minerule catalogue seems
already installed.") << std::endl;
```

```
00224                         minerule::MRLog() « minerule::StringUtils::toRed("Cowardly refusing to install
      over a working catalogue...") « std::endl;
00225                         return NOTHING_TO_DO;
00226                 } else {
00227                         minerule::MRLog() « "Checking if everything is ok." « std::endl;
00228                         minerule::MRLog() « minerule::StringUtils::toBold("The catalogue is not
      installed (properly).") « std::endl;
00229                         minerule::MRLog() « "Proceeding with the installation..." « std::endl;
00230                         minerule::OptimizerCatalogue::install();
00231                         minerule::MRLog() « "Done!" « std::endl;
00232                         minerule::MRLog() « "Checking the installation..." « std::endl;

00233                         if(!minerule::OptimizerCatalogue::checkInstallation()) {
00234                                 minerule::MRLog() « "Something went wrong. This is likely a bug.
      Please report it!" « std::endl;
00235                                 return CATALOGUE_NOT_INSTALLED;
00236                         } else {
00237                                 minerule::MRLog() « minerule::StringUtils::toBold("The catalogue is
      now properly installed.") « std::endl;
00238                                 return SUCCESS;
00239                         }
00240                 }
00241         }
00242
00243         Results uninstallCatalogue() {
00244                 minerule::MRLog() « "You choose to " « minerule::StringUtils::toBold(" uninstall ") «
      "the minerule catalogue" « std::endl;
00245                 minerule::MRLog() « "This implies " « minerule::StringUtils::toBold(" destroying ") «
      "all its tables" « std::endl;
00246
00247                 std::cout « "Are you " « minerule::StringUtils::toBold("sure ") « "you want to
      proceed? (y/N): ";
00248                 std::string confirm;
00249                 std::cin » confirm;
00250                 transform(confirm.begin(), confirm.end(), confirm.begin(), ::toupper);
00251                 if( confirm != "Y" && confirm != "YES") {
00252                         minerule::MRLog() « "Ok. Bailing out." « std::endl;
00253                         return NOTHING_TO_DO;
00254                 }
00255
00256                 minerule::MRLog() « "Uninstalling the catalogue..." « std::endl;
00257                 minerule::OptimizerCatalogue::uninstall();
00258                 minerule::MRLog() « "Done" « std::endl;
00259
00260                 return SUCCESS;
00261         }
00262
00263
00264         Results addDerivedQuery(const Options& options) {
00265                 minerule::OptimizerCatalogue::addDerivedResult(options.getOriginalQuery(),
      options.getDerivedQuery());
00266                 return SUCCESS;
00267         }
00268
00269         // ------------------------------------------------------------------------
00270         // MAIN ACTION HANDLER
00271         // ------------------------------------------------------------------------
00272
00273         int     performCommands(Options& options) {
00274                 if( options.getCommand()==Options::ShowList ) {
00275                         printMRQueryList(options);
00276                         return mrc::SUCCESS;
00277                 }
00278
00279                 if( options.getCommand()==Options::SearchQry) {
00280                         return checkQueryName(options);
00281                 }
00282
00283                 if( options.getCommand()==Options::DeleteQry ) {
00284                         deleteQuery(options);
00285                         return mrc::SUCCESS;
00286                 }
00287
00288                 if( options.getCommand()==Options::CheckCatalogue ) {
00289                         return checkCatalogue();
00290                 }
00291
00292                 if( options.getCommand()==Options::InstallCatalogue ) {
00293                         return installCatalogue();
00294                 }
00295
00296                 if( options.getCommand()==Options::UninstallCatalogue ) {
00297                         return uninstallCatalogue();
00298                 }
00299
00300                 if( options.getCommand()==Options::AddDerivedQuery ) {
00301                         return addDerivedQuery(options);
```

```
00302                 }
00303
00304
00305                 return mrc::NOTHING_TO_DO;
00306         }
00307 } // namespace mrc
```

7.309 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ MRCUtils.hpp File Reference

```
#include "Options.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
```

Namespaces

- namespace mrc

Functions

- void mrc::printVersion ()
- void mrc::printHelp (int argc, char **argv)
- void mrc::parseOptions (int argc, char **argv, minerule::MineruleOptions &mopt, Options &popt)
- int mrc::performCommands (Options &options)

7.310 MRCUtils.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef MRCATALOGUE_H_89D0JOQH
00017 #define MRCATALOGUE_H_89D0JOQH
00018
00019 #include "Options.hpp"
00020 #include "minerule/Utils/MineruleOptions.hpp"
00021
00022 namespace mrc { // minerule catalogue
00023
00024         // ------------------------------------------------------------------------
00025         // Functions
00026         // ------------------------------------------------------------------------
00027         void printVersion();
00028         void printHelp(int argc, char** argv);
00029         void parseOptions(int argc, char** argv, minerule::MineruleOptions& mopt, Options& popt);
00030         int  performCommands(Options& options);
00031 }
00032
00033 #endif /* end of include guard: MRCATALOGUE_H_89D0JOQH */
```

7.311 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/Printer.cpp File Reference

```
#include "Printer.hpp"
#include "minerule/Utils/StringUtils.hpp"
#include <sstream>
```

Namespaces

- namespace mrc

7.312 Printer.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "Printer.hpp"
00017 #include "minerule/Utils/StringUtils.hpp"
00018 #include <sstream>
00019
00020 namespace mrc {
00021         const size_t  Printer::OUTPUT_MAX_LEN = 70;
00022
00023
00024         void Printer::printIndex() const {
00025                 std::stringstream ss;
00026                 ss «  _result_index;
00027                 _out « "-" « minerule::StringUtils::toBold( ss.str() ) « "- ";
00028         }
00029
00030         void Printer::format(std::string header, std::string info) {
00031                 printIndex();
00032                 std::vector<std::string>* chunks = minerule::StringUtils::splitToLength(info,
      OUTPUT_MAX_LEN);
00033                 assert(chunks->size()!=0);
00034
00035                 std::vector<std::string>::const_iterator it = chunks->begin();
00036                 _out « minerule::StringUtils::toGreen(header) « "\t" « *it « std::endl;

00037                 ++it;
00038
00039                 for(; it!=chunks->end(); ++it) {
00040                         _out « "\t" « *it « std::endl;
00041                 }
00042
00043                 delete chunks;
00044         }
00045
00046         void Printer::format(std::string header, size_t info) {
00047                 printIndex();
00048                 _out « minerule::StringUtils::toGreen(header) « "\t"  « info « std::endl;
00049         }
00050
00051
00052         void Printer::print(const minerule::CatalogueInfo& info) {
00053                 format("name:", info.qryName);
00054
00055                 if(_options.getListFormat().size) {
```

```
00056                         format("size:", info.resSize);
00057                 }
00058
00059                 if(_options.getListFormat().result) {
00060                         format("tables:\n", minerule::StringUtils::join(info.resTables, ", "));
00061                 }
00062
00063                 if(_options.getListFormat().text) {
00064                         format("text:\n", info.qryText);
00065                 }
00066         }
00067
00068         void Printer::print(const std::vector<minerule::CatalogueInfo>& list)  {
00069                 _out « "Found " « list.size() « " result sets:" « std::endl;
00070                 _result_index = 1;
00071                 std::vector<minerule::CatalogueInfo>::const_iterator it;
00072                 for(it=list.begin(); it!=list.end(); it++) {
00073                         print(*it);
00074                         _result_index++;
00075                 }
00076         }
00077 }
```

7.313 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ Printer.hpp File Reference

```
#include <iostream>
#include "Options.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
```

Data Structures

- class mrc::Printer

Namespaces

- namespace mrc

7.314 Printer.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef PRINTER_H_LDJGVI61
00017 #define PRINTER_H_LDJGVI61
00018
00019 #include <iostream>
00020 #include "Options.hpp"
00021 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00022
```

```
00023 namespace mrc {
00024         class Printer {
00025                 std::ostream& _out;
00026                 const Options& _options;
00027                 size_t _result_index;
00028                 static const size_t OUTPUT_MAX_LEN;
00029         public:
00030                 Printer(std::ostream& out, const Options& options) : _out(out), _options(options),
      _result_index(1) {}
00031
00032                 void print(const minerule::CatalogueInfo& info);
00033                 void print(const std::vector<minerule::CatalogueInfo>& list);
00034         private:
00035                 void printIndex() const;
00036                 void format(std::string header, std::string info);
00037                 void format(std::string header, size_t info);
00038         };
00039 }
00040
00041
00042 #endif /* end of include guard: PRINTER_H_LDJGVI61 */
```

7.315 /Users/esposito/Software/minerule/src/Programs/MRDefaultOptions/MRDefaultOptions.cpp File Reference

```
#include <iostream>
#include <unistd.h>
#include "minerule/Utils/MineruleOptions.hpp"
```

Functions

- void printHelp (std::string progname)
- void printVersion ()
- void parseOptions (int argc, char **argv, MineruleOptions &mrOpts)
- int main (int argc, char **argv)

7.315.1 Function Documentation

7.315.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 86 of file MRDefaultOptions.cpp.

```
00086                                {
00087   try {
00088     MineruleOptions& mropts = MineruleOptions::getSharedOptions();
00089     parseOptions(argc,argv,mropts);
00090     mropts.saveOptions(std::cout);
00091   } catch( std::exception& e) {
00092     std::cerr << e.what() << std::endl;
00093     exit(1);
00094   }
00095 }
```

7.315.1.2 parseOptions()

```
void parseOptions (
            int argc,
            char ** argv,
            MineruleOptions & mrOpts )
```

Definition at line 53 of file MRDefaultOptions.cpp.

```
00053                                                                   {
00054   int opt;
00055
00056
00057   while((opt=getopt(argc,argv,"hf:O:v"))!=-1) {
00058     switch(opt) {
00059     case 'h':
00060       printHelp(argv[0]);
00061       exit(0);
00062     case 'v':
00063       printVersion();
00064       exit(0);
00065     case 'O':
00066       try {
00067         mrOpts.readFromString(std::string(optarg));
00068       } catch( mrdb::SQLException& e) {
00069         // do nothing
00070       }
00071       break;
00072     case 'f':
00073       mrOpts.readFromFile(std::string(optarg));
00074       break;
00075     default:
00076       std::cerr « "Option -" « opt
00077            « " not recognized! Please check the parameters" « std::endl;
00078       exit(1);
00079     }
00080   }
00081
00082 }
```

7.315.1.3 printHelp()

```
void printHelp (
            std::string progname )
```

Definition at line 24 of file MRDefaultOptions.cpp.

```
00024                           {
00025   std::cout « StringUtils::toBold("Usage: ") « progname « " [-h] [-v] [-f optionfile] [-O
       optionstring] [optfile]" « std::endl;
00026   std::cout « std::endl;
00027   std::cout
00028            « "  If invoked without arguments the program output (stdout)" « std::endl
00029        « "  the default options used by the Minerule system." « std::endl « std::endl
00030            « "  If " « StringUtils::toBold("-v") «" is given, the program outputs its version number
       and exits" « std::endl «std::endl
00031        « "  If " « StringUtils::toBold("-h") « " is given, the program outputs this message." «
       std::endl
00032        « "  If an option file is specified, the program reads it and" « std::endl
00033        « "  outputs the resulting options immediately afterwards" « std::endl
00034        « "  (useful to check that the file syntax is correct and that"«std::endl
00035        « "  the program correctly parses it)." « std::endl
00036        « "  If more than one option file is specified, latter options" «std::endl
00037        « "  override former ones. The same holds with the options" « std::endl
00038        « "  given using the "«StringUtils::toBold("-O")«" flag" « std::endl « std::endl;
00039   std::cout « StringUtils::toBold("NOTE:") « " all options regarding streams will be either
       commented"« std::endl
00040        « "  or set to default values (for implementation reasons it is"« std::endl
00041        « "  not possible to retrieve the file name they point to and " « std::endl
00042        « "  printing a number representing a pointer in memory is likely"«std::endl
00043        « "  to lead to some kind of misunderstanding)" «std::endl;
00044 }
```

7.315.1.4 printVersion()

```
void printVersion ( )
```

Definition at line 48 of file MRDefaultOptions.cpp.

```
00048                     {
00049   std::cout « "mrdefaults v:" « MR_VERSION « std::endl;
00050 }
```

7.316 MRDefaultOptions.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <unistd.h>
00018 #include "minerule/Utils/MineruleOptions.hpp"
00019
00020
00021 using namespace minerule;
00022
00023 void
00024 printHelp(std::string progname) {
00025   std::cout « StringUtils::toBold("Usage: ") « progname « " [-h] [-v] [-f optionfile] [-O
       optionstring] [optfile]" « std::endl;
00026   std::cout « std::endl;
00027   std::cout
00028            « "  If invoked without arguments the program output (stdout)" « std::endl
00029        « "  the default options used by the Minerule system." « std::endl « std::endl
00030            « "  If " « StringUtils::toBold("-v") «" is given, the program outputs its version number
       and exits" « std::endl «std::endl
00031        « "  If " « StringUtils::toBold("-h") « " is given, the program outputs this message." «
       std::endl
00032        « "  If an option file is specified, the program reads it and" « std::endl
00033        « "  outputs the resulting options immediately afterwards" « std::endl
00034        « "  (useful to check that the file syntax is correct and that"«std::endl
00035        « "  the program correctly parses it)." « std::endl
00036        « "  If more than one option file is specified, latter options" «std::endl
00037        « "  override former ones. The same holds with the options" « std::endl
00038        « "  given using the "«StringUtils::toBold("-O")«" flag" « std::endl « std::endl;
00039   std::cout « StringUtils::toBold("NOTE:") « " all options regarding streams will be either
       commented"« std::endl
00040        « "  or set to default values (for implementation reasons it is"« std::endl
00041        « "  not possible to retrieve the file name they point to and " « std::endl
00042        « "  printing a number representing a pointer in memory is likely"«std::endl
00043        « "  to lead to some kind of misunderstanding)" «std::endl;
00044 }
00045
00046
00047
00048 void printVersion() {
00049   std::cout « "mrdefaults v:" « MR_VERSION « std::endl;
00050 }
00051
00052 void
00053 parseOptions(int argc, char** argv, MineruleOptions& mrOpts) {
00054   int opt;
00055
00056
00057   while((opt=getopt(argc,argv,"hf:O:v"))!=-1) {
00058     switch(opt) {
00059     case 'h':
00060       printHelp(argv[0]);
00061       exit(0);
00062     case 'v':
```

```
00063       printVersion();
00064       exit(0);
00065     case 'O':
00066       try {
00067         mrOpts.readFromString(std::string(optarg));
00068       } catch( mrdb::SQLException& e) {
00069         // do nothing
00070       }
00071       break;
00072     case 'f':
00073       mrOpts.readFromFile(std::string(optarg));
00074       break;
00075     default:
00076       std::cerr « "Option -" « opt
00077            « " not recognized! Please check the parameters" « std::endl;
00078       exit(1);
00079     }
00080   }
00081
00082 }
00083
00084
00085 int
00086 main(int argc, char** argv) {
00087   try {
00088     MineruleOptions& mropts = MineruleOptions::getSharedOptions();
00089     parseOptions(argc,argv,mropts);
00090     mropts.saveOptions(std::cout);
00091   } catch( std::exception& e) {
00092     std::cerr « e.what() « std::endl;
00093     exit(1);
00094   }
00095 }
00096
```

7.317 /Users/esposito/Software/minerule/src/Programs/MRMatch/GidRulesMatcher.cpp File Reference

```
#include "GidRulesMatcher.hpp"
#include "minerule/Result/RuleFormatter.hpp"
```

Namespaces

- namespace mrmatch

7.318 GidRulesMatcher.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "GidRulesMatcher.hpp"
00017 #include "minerule/Result/RuleFormatter.hpp"
00018
00019 using namespace minerule;
00020
```

```
00021 namespace mrmatch {
00022
00023         Rule& GidRulesMatcher::addRule() {
00024                 rules.push_back(Rule());
00025                 return rules.back();
00026         }
00027
00028         void GidRulesMatcher::matchItemTransaction(const minerule::ItemType& gid, const
      ItemTransaction<RulesMatcher::ItemSetType>& bodies,const ItemTransaction<RulesMatcher::ItemSetType>&
      heads) {
00029                 _gidRulesVector.push_back( GidRules() );
00030                 GidRules& gidRules = _gidRulesVector.back();
00031                 gidRules.first = gid;
00032
00033                 for(RuleVector::iterator ruleIt = rules.begin(); ruleIt!=rules.end(); ++ruleIt) {

00034                         if( RulesMatcher::match( *ruleIt, bodies, heads ) ) {
00035                                 gidRules.second.push_back(& (*ruleIt) );
00036                         }
00037                 }
00038         }
00039
00040         void GidRulesMatcher::matchRuleTransaction(const minerule::ItemType& gid, const
      RuleTransaction<RulesMatcher::RuleSetType>& transaction) {
00041                 _gidRulesVector.push_back( GidRules() );
00042                 GidRules& gidRules = _gidRulesVector.back();
00043                 gidRules.first = gid;
00044
00045                 for(RuleVector::iterator ruleIt = rules.begin(); ruleIt!=rules.end(); ++ruleIt) {

00046                         if( RulesMatcher::match( *ruleIt, transaction ) ) {
00047                                 gidRules.second.push_back(& (*ruleIt) );
00048                         }
00049                 }
00050         }
00051
00052         void GidRulesMatcher::printMatches() const {
00053                 SimpleRuleFormatter sf;
00054                 sf.setFieldWidths( SimpleRuleFormatter::FieldWidths(1,1,1,1) );
00055
00056                 for( GidRulesVector::const_iterator it= _gidRulesVector.begin();
      it!=_gidRulesVector.end(); ++it ) {
00057                                 MRLog() « StringUtils::toBold("Gid: ") « it->first « std::endl;
00058                                 const RulePtrVector& matchedRules = it->second;
00059
00060                                 for( RulePtrVector::const_iterator rIt=matchedRules.begin();
      rIt!=matchedRules.end(); ++rIt) {
00061                                         MRLog() « StringUtils::toBold("\tRule: ") « sf.formatRule(
      **rIt ) « std::endl;
00062                                 }
00063                 }
00064         }
00065
00066 }
```

7.319 /Users/esposito/Software/minerule/src/Programs/MRMatch/Gid↩ RulesMatcher.hpp File Reference

```
#include "Matcher.hpp"
```

Data Structures

- class mrmatch::GidRulesMatcher

Namespaces

- namespace mrmatch

7.320 GidRulesMatcher.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef GIDRULESMATCHER_H_E8FYP5QZ
00017 #define GIDRULESMATCHER_H_E8FYP5QZ
00018
00019 #include "Matcher.hpp"
00020
00021 namespace mrmatch {
00022
00023         class GidRulesMatcher : public Matcher {
00024         private:
00025                 typedef std::vector< minerule::Rule > RuleVector;
00026                 typedef std::vector< minerule::Rule* > RulePtrVector;
00027                 typedef std::pair< minerule::ItemType, RulePtrVector > GidRules;
00028                 typedef std::vector< GidRules > GidRulesVector;
00029
00030                 RuleVector rules;                                                                    //
       stores all known rules
00031                 GidRulesVector _gidRulesVector; // stores gid -> rules association
00032
00033
00034         protected:
00035                 virtual void matchItemTransaction(const minerule::ItemType& gid, const
       minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& bodies,const
       minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& heads);
00036                 virtual void matchRuleTransaction(const minerule::ItemType& gid, const
       minerule::RuleTransaction<minerule::RulesMatcher::RuleSetType>& transaction);
00037
00038         public:
00039                 GidRulesMatcher() {};
00040                 virtual ~GidRulesMatcher() {};
00041
00042                 // create a new empty rule and returns it (so that it can be intialized)
00043                 virtual minerule::Rule& addRule();
00044
00045                 // outputs the results
00046                 virtual void printMatches() const;
00047         };
00048
00049 }
00050
00051 #endif /* end of include guard: GIDRULESMATCHER_H_E8FYP5QZ */
```

7.321 /Users/esposito/Software/minerule/src/Programs/MRMatch/Matcher.cpp File Reference

```
#include "Matcher.hpp"
#include "RuleGidsMatcher.hpp"
#include "RuleGidsDBMatcher.hpp"
#include "GidRulesMatcher.hpp"
#include "minerule/Utils/MineruleErrors.hpp"
```

Namespaces

- namespace mrmatch

7.322 Matcher.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "Matcher.hpp"
00017 #include "RuleGidsMatcher.hpp"
00018 #include "RuleGidsDBMatcher.hpp"
00019 #include "GidRulesMatcher.hpp"
00020 #include "minerule/Utils/MineruleErrors.hpp"
00021
00022 using namespace minerule;
00023
00024 namespace mrmatch {
00025         Matcher* Matcher::newMatcher(const Options& opts, const ParsedMinerule& minerule) {
00026                 MatcherSpecs specs = opts.matcherSpecs();
00027                 if((specs & MatchOutputMask) == OutOnDB) {
00028                         return new RuleGidsDBMatcher(opts.getMatchOutputTableName(), minerule);
00029                 }
00030
00031                 switch(specs & MatchKindMask) {
00032                         case RuleGids: return new RuleGidsMatcher();
00033                         case GidRules: return new GidRulesMatcher();
00034                         default: throw MineruleException( MR_ERROR_INTERNAL, "Unknown or unsupported
      matcher kind");
00035                 }
00036         }
00037
00038         void Matcher::matchWithCrossProduct(SourceTable& st) {
00039                 SourceTable::Iterator it = st.newIterator(SourceTable::FullIterator);
00040
00041                 while(!it.isAfterLast()) {
00042                         ItemType gid = it->getGroup();
00043
00044                         RuleTransaction<RulesMatcher::RuleSetType> transaction;
00045                         transaction.load(gid, it);
00046
00047                         matchRuleTransaction(gid, transaction);
00048                 }
00049         }
00050
00051         void Matcher::matchWithoutCrossProduct(SourceTable& st) {
00052                 SourceTable::Iterator bodyIt = st.newIterator(SourceTable::BodyIterator);
00053                 SourceTable::Iterator headIt = st.newIterator(SourceTable::HeadIterator);
00054
00055                 while(!bodyIt.isAfterLast()) {
00056                         ItemType gid = bodyIt->getGroup();
00057
00058                         ItemTransaction<RulesMatcher::ItemSetType> bodies;
00059                         ItemTransaction<RulesMatcher::ItemSetType> heads;
00060
00061                         bodies.loadBody(gid, bodyIt);                   // this advances the body
      iterator
00062
00063                         if( !TransactionBase<RulesMatcher::ItemSetType>::findGid(gid, headIt) ) {//
      positioning the head iterator
00064                                 break;                                                                  // no
      more heads to load
00065                         }
00066
00067                         heads.loadHead(gid, headIt);                    // loading the heads
00068
00069                         matchItemTransaction(gid, bodies, heads);
00070                 } // while
00071
00072         } // matchWithoutCrossProduct
00073
00074 }
```

7.323 /Users/esposito/Software/minerule/src/Programs/MRMatch/↩ Matcher.hpp File Reference

```
#include "minerule/Result/Rule.hpp"
#include "minerule/Database/SourceTable.hpp"
#include "minerule/Database/Transaction.hpp"
#include "minerule/Result/RulesMatcher.hpp"
#include "Options.hpp"
#include "mrmatch.hpp"
```

Data Structures

- class mrmatch::Matcher

Namespaces

- namespace mrmatch

7.324 Matcher.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef MATCHERS_H_Q12443YU
00017 #define MATCHERS_H_Q12443YU
00018
00019 #include "minerule/Result/Rule.hpp"
00020 #include "minerule/Database/SourceTable.hpp"
00021 #include "minerule/Database/Transaction.hpp"
00022 #include "minerule/Result/RulesMatcher.hpp"
00023
00024 #include "Options.hpp"
00025 #include "mrmatch.hpp"
00026
00027 namespace mrmatch {
00028
00029 // Base class for all Matchers
00030         class Matcher {
00031         public:
00032         // creates a new matcher and returns it
00033                 static Matcher* newMatcher(const Options& opts, const minerule::ParsedMinerule& p);
00034
00035         // create a new empty rule and returns it (so that it can be intialized)
00036                 virtual minerule::Rule& addRule() = 0;
00037
00038         // Match the given source table with the current set of rules
00039                 virtual void match(minerule::SourceTable& st) {
00040                         st.usesCrossProduct() ? matchWithCrossProduct(st) :
      matchWithoutCrossProduct(st);
00041                 }
00042
00043                 virtual void matchWithCrossProduct(minerule::SourceTable& st);
00044                 virtual void matchWithoutCrossProduct(minerule::SourceTable& st);
```

```
00045
00046         // outputs the results
00047                 virtual void printMatches() const = 0;
00048
00049         protected:
00050                 virtual void matchItemTransaction(const minerule::ItemType& gid, const
       minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& bodies,const
       minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& heads) = 0;
00051                 virtual void matchRuleTransaction(const minerule::ItemType& gid, const
       minerule::RuleTransaction<minerule::RulesMatcher::RuleSetType>& transaction) = 0;
00052         };
00053
00054 }
00055
00056 #endif /* end of include guard: MATCHERS_H_Q12443YU */
```

7.325 /Users/esposito/Software/minerule/src/Programs/↩ MRMatch/mrmatch.cpp File Reference

```
#include <iostream>
#include <getopt.h>
#include "minerule/Result/RulesMatcher.hpp"
#include "minerule/Result/RuleFormatter.hpp"
#include "Matcher.hpp"
#include "mrmatch.hpp"
#include "Options.hpp"
#include <memory>
```

Namespaces

- namespace mrmatch

Functions

- SourceTableRequirements mrmatch::sourceTableRequirements (const ParsedMinerule &minerule)
- void mrmatch::execute (const Options &options)
- int main (int argc, char ∗const argv[ ])

7.325.1 Function Documentation

7.325.1.1 main()

```
int main (
            int argc,
            char *const argv[] )
```

Definition at line 79 of file mrmatch.cpp.

```
00079                                       {
00080         try {
00081                 mrmatch::Options options;
00082
00083                 options.parse(argc, argv);
```

```
00084                 if(!options.initMineruleOptions()) {
00085                         std::cout « "Minerule Options not specified." « std::endl
00086                                 « "They can be specified in several ways, you can:" « std::endl
00087                                 « " provide the path to an option file using flag " «
      StringUtils::toBold("-f") « "." « std::endl
00088                                 « " launch the program from a directory containing an option file
      named 'options.txt'" « std::endl
00089                                 « " provide the options on the command line using " «
      StringUtils::toBold("-O") « " flags." « std::endl;
00090
00091                         exit(mrmatch::MRMATCH_OPTION_PARSING_ERROR);
00092                 }
00093
00094                 execute(options);
00095
00096         } catch( std::exception& e ) {
00097                 std::cerr « e.what() « std::endl;
00098         }
00099
00100         return 0;
00101 }
```

7.326 mrmatch.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <getopt.h>
00018
00019 #include "minerule/Result/RulesMatcher.hpp"
00020 #include "minerule/Result/RuleFormatter.hpp"
00021
00022 #include "Matcher.hpp"
00023 #include "mrmatch.hpp"
00024 #include "Options.hpp"
00025 #include <memory>
00026
00027
00028 using namespace minerule;
00029
00030 namespace mrmatch {
00031
00032         SourceTableRequirements sourceTableRequirements(const ParsedMinerule& minerule) {
00033                 if(minerule.hasCrossConditions() || !OptimizerCatalogue::hasIDConstraints(minerule))
00034                         return SourceTableRequirements( SourceTableRequirements::CrossProduct |
      SourceTableRequirements::SortedGids );
00035                 else
00036                         return SourceTableRequirements(SourceTableRequirements::SortedGids);

00037         }
00038
00039         void execute(const Options& options) {
00040                 MRLogPush("Building source table...");
00041                 // rebuild source table
00042                 CatalogueInfo info;
00043                 OptimizerCatalogue::getMRQueryInfo(options.queryName(), info, false);
00044                 ParsedMinerule minerule(info.qryText);
00045                 if( options.tableName() != "" ) {
00046                         minerule.tab_source = options.tableName();
00047                 }
00048
00049                 SourceTableRequirements requirements = sourceTableRequirements(minerule);
00050                 SourceTable st( minerule, requirements);
00051                 MRLogPop();
00052
00053                 MRLogPusher("Reading rules...");
00054                 // load past minerule result
```

```
00055                 QueryResult::Iterator it;
00056                 OptimizerCatalogue::getMRQueryResultIterator(options.queryName(), it, minerule.sup,
      minerule.conf);
00057
00058                 std::auto_ptr<Matcher> matcher(Matcher::newMatcher(options, minerule));
00059
00060                 while(it.next()) {
00061                         Rule& rule = matcher->addRule();
00062                         it.getRule(rule);
00063                 }
00064                 MRLogPop();
00065
00066                 MRLogPush("Matching...");
00067                 matcher->match(st);
00068                 MRLogPop();
00069
00070                 MRLogPush("Printing results:");
00071                 matcher->printMatches( );
00072
00073                 MRLogPop();
00074         }
00075
00076 }
00077
00078
00079 int main (int argc, char* const argv[]) {
00080         try {
00081                 mrmatch::Options options;
00082
00083                 options.parse(argc, argv);
00084                 if(!options.initMineruleOptions()) {
00085                         std::cout « "Minerule Options not specified." « std::endl
00086                                 « "They can be specified in several ways, you can:" « std::endl
00087                                 « " provide the path to an option file using flag " «
      StringUtils::toBold("-f") « "." « std::endl
00088                                 « " launch the program from a directory containing an option file
      named 'options.txt'" « std::endl
00089                                 « " provide the options on the command line using " «
      StringUtils::toBold("-O") « " flags." « std::endl;
00090
00091                         exit(mrmatch::MRMATCH_OPTION_PARSING_ERROR);
00092                 }
00093
00094                 execute(options);
00095
00096         } catch( std::exception& e ) {
00097                 std::cerr « e.what() « std::endl;
00098         }
00099
00100         return 0;
00101 }
```

7.327 /Users/esposito/Software/minerule/src/Programs/↩ MRMatch/mrmatch.hpp File Reference

```
#include <vector>
#include "minerule/Result/Rule.hpp"
```

Namespaces

- namespace mrmatch

Typedefs

- typedef unsigned int mrmatch::MatcherSpecs

Enumerations

- enum mrmatch::ExitCodes { mrmatch::SUCCESS =0 , mrmatch::MRMATCH_OPTION_PARSING_ERROR }
- enum mrmatch::MatchKind { mrmatch::None = 0 , mrmatch::RuleGids = 1 , mrmatch::GidRules = 2 , mrmatch::MatchKindMask = 3 }
- enum mrmatch::MatchOutput { mrmatch::OutOnConsole = 4 , mrmatch::OutOnDB = 8 , mrmatch::MatchOutputMask = 12 }

7.328 mrmatch.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef MRMATCH_H_MP1T5YIM
00017 #define MRMATCH_H_MP1T5YIM
00018
00019 #include <vector>
00020 #include "minerule/Result/Rule.hpp"
00021
00022 namespace mrmatch {
00023         typedef enum { SUCCESS=0, MRMATCH_OPTION_PARSING_ERROR  } ExitCodes;
00024
00025         // A matcher spec is the bit and of a match kind and a matchoutput.
00026         // the values of the two types have been designed so that one can
00027         // combine a matchkind and a matchoutput using the or bit-operator
00028         // and retrieve the components by anding the result with the appropriate
00029         // matchmask.
00030         // Eg. GidRules | OutOnDb --> 10
00031         //     10 & MatchKindMask --> 2 == GidRules
00032         //     10 & MatchOutputMask --> 8 ==- OutOnDb
00033         // Alternatively, one can check if a particular flag is set by anding the
00034         // result with one of the MatchKind/MatchOutput.
00035         // Eg. GidRules | OutOnDb --> 10
00036         //     10 & GidRules --> true
00037         //     10 & RuleGids --> false
00038         //     10 & OutOnConsole --> false
00039         //     10 & OutOnDb --> true
00040         typedef enum { None = 0, RuleGids = 1, GidRules = 2, MatchKindMask = 3 } MatchKind;
00041         typedef enum { OutOnConsole = 4, OutOnDB = 8, MatchOutputMask = 12 } MatchOutput;
00042         typedef unsigned int MatcherSpecs;
00043 }
00044
00045
00046 #endif /* end of include guard: MRMATCH_H_MP1T5YIM */
```

7.329 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ Options.cpp File Reference

```
#include "Options.hpp"
```

Namespaces

- namespace mrc

7.330 Options.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "Options.hpp"
00017
00018 namespace mrc {
00019
00020
00021         void Options::setListFormat(const char* format) {
00022                 if(format==NULL)
00023                         return;
00024
00025                 for(int i=0; format[i]!='\0'; i++) {
00026                         switch( format[i] ) {
00027                                 case 's':
00028                                         listFormat.size=true;
00029                                         break;
00030                                 case 't':
00031                                         listFormat.text = true;
00032                                         break;
00033                                 case 'r':
00034                                         listFormat.result = true;
00035                                         break;
00036                                 default:
00037                                         throw Exception(mrc::ERROR_OPTION_PARSING,
       std::string("Unknown flag:")+format[i]);
00038                         }
00039                 }
00040         }
00041
00042 }
```

7.331 /Users/esposito/Software/minerule/src/Programs/MRMatch/↩ Options.cpp File Reference

```
#include "mrmatch.hpp"
#include "Options.hpp"
#include <getopt.h>
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Utils/FileUtils.hpp"
```

Namespaces

- namespace mrmatch

7.332 Options.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
```

```
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "mrmatch.hpp"
00017 #include "Options.hpp"
00018 #include <getopt.h>
00019
00020 #include "minerule/Utils/MineruleOptions.hpp"
00021 #include "minerule/Utils/FileUtils.hpp"
00022
00023 using namespace minerule;
00024
00025 namespace mrmatch {
00026         void Options::printUsage(int argc, char* const argv[]) {
00027                 char* progName = argv[0];
00028
00029                 std::cout « StringUtils::toBold("Usage:") « std::endl
00030                         « "   " « StringUtils::toBold(progName) « " [-h] [-c] [-n <num query>] [-O
       <optionlist>] [-f <optionfile>] [-d <outputtable>]  [-t <table name>] [-s <sort order>] [<query
       name>] " « std::endl « std::endl;
00031
00032                 std::cout
00033                         « StringUtils::toBold("   -h") « " - prints this message." « std::endl
00034                         « StringUtils::toBold("   -c") « " - suppress colors." « std::endl
00035                         « StringUtils::toBold("   -n") « " - specifies a query number (instead of a
       query name)." « std::endl
00036                         « StringUtils::toBold("   -O") « " - specifies a minerule option on the
       command line (overrides those read from file)." « std::endl
00037                         « StringUtils::toBold("   -f") « " - specify a file name containing the
       Options to be used." « std::endl
00038                         « "         default is 'optins.txt'"« std::endl
00039                         « StringUtils::toBold("   -d") « " - redirect the output onto the database.
       The output table must" «std::endl
00040                         « "         be specified as the argument to this option" « std::endl
00041                         « StringUtils::toBold("   -t") « " - specify a table name for the match (the
       table *must* have the" « std::endl
00042                         « "         same schema as the mining table used for the query)" « std::endl
00043                         « StringUtils::toBold("   -s") « " - sets the output sorting order. The
       parameter can be set to:" « std::endl
00044                         « StringUtils::toBold("         RuleGids") « ", to produce an output with the
       format: 'rule -> list of matching gids'" « std::endl
00045                         « StringUtils::toBold("         GidRules") « ", to produce an output with the
       format: 'gid -> list of matching rules'."  « std::endl
00046                         « "         default is 'RuleGids'. You can shorten the parameters as you like
       (e.g., 'G' for GidRules)." « std::endl;
00047
00048                 exit(SUCCESS);
00049         }
00050
00051
00052         MatchKind Options::stringToMatchKind(const std::string& str) {
00053                 if(std::string("RuleGids").find(str) == 0) // if str is a prefix for RuleGids
00054                         return RuleGids;
00055                 else if(std::string("GidRules").find(str) == 0) // if str is a prefix for GidRules
00056                         return GidRules;
00057                 else {
00058                         std::cout « "Option parsing error: option -s can take only values in
       {RuleGids,GidRules}" « std::endl;
00059                         exit(MRMATCH_OPTION_PARSING_ERROR);
00060                 }
00061         }
00062
00063
00064         bool Options::initMineruleOptions() const {
00065                 bool ok=false;
00066                 MineruleOptions& options = MineruleOptions::getSharedOptions();
00067
00068                 if( !_mrOptionsFileName.empty() ) {
00069                         options.readFromFile(_mrOptionsFileName);
00070                         ok=true;
00071                 } else if( FileUtils::fileExists("options.txt") ) {
00072                         options.readFromFile("options.txt");
00073                         ok=true;
00074                 }
00075
00076                 std::vector<std::string>::const_iterator it;
```

```
00077                 for(it=_mrOptionsInline.begin(); it!=_mrOptionsInline.end(); ++it) {
00078                         options.readFromString(*it);
00079                         ok=true;
00080                 }
00081
00082                 return ok;
00083         }
00084
00085         std::string Options::queryName() const {
00086                 if( _queryNumber != 0 && !_queryName.empty() ) {
00087                         std::cout << StringUtils::toBold("Error:") << "It appears that you specified
      both a query name and a query number" << std::endl;
00088                         exit(MRMATCH_OPTION_PARSING_ERROR);
00089                 }
00090
00091                 if( _queryNumber != 0 )
00092                         return OptimizerCatalogue::getMRQueryName(_queryNumber);
00093
00094                 if( !_queryName.empty() )
00095                         return _queryName;
00096
00097                 std::cout << StringUtils::toBold("Error:") << "It appears that you specified neither a
      query name nor a query number" << std::endl;
00098                 exit(MRMATCH_OPTION_PARSING_ERROR);
00099         }
00100
00101         const std::string& Options::tableName() const {
00102                 return _tableName;
00103         }
00104
00105         void Options::setTableName(const std::string& tableName) {
00106                 _tableName = tableName;
00107         }
00108
00109
00110         void Options::parse(int argc,  char* const argv[]) {
00111                 char ch;
00112
00113                 while( (ch=getopt(argc, argv, "hcn:s:d:t:")) != -1 ) {
00114                         switch(ch) {
00115                                 case 'h':
00116                                         printUsage(argc, argv);
00117                                         break;
00118                                 case 'c':
00119                                         StringUtils::setColorsEnabled(false);
00120                                         break;
00121                                 case 'n':
00122                                         setQueryNumber(Converter(optarg).toLong());
00123                                         break;
00124                                 case 's':
00125                                         setMatchKind(stringToMatchKind(optarg));
00126                                         break;
00127                                 case 'd':
00128                                         setMatchOutputTableName(optarg);
00129                                         setMatchOutput(OutOnDB);
00130                                         break;
00131                                 case 't':
00132                                         setTableName(optarg);
00133                                         break;
00134                                 case '?':
00135                                 default:
00136                                         std::cout << StringUtils::toBold("Option not recognized:") <<
      "-" << ch << std::endl;
00137                                         printUsage(argc, argv);
00138                                         exit(MRMATCH_OPTION_PARSING_ERROR);
00139                         }
00140                 }
00141
00142                 if( optind < argc-1 ) {
00143                         std::cout << StringUtils::toBold("Error:") << "some of the options were not
      recognized" << std::endl;
00144
00145                         printUsage(argc, argv);
00146                         exit(MRMATCH_OPTION_PARSING_ERROR);
00147                 }
00148
00149                 if( optind == argc-1 )
00150                         setQueryName(argv[argc-1]);
00151         }
00152
00153
00154
00155
00156 }
```

7.333 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/↩ Options.cpp File Reference

```
#include "Options.hpp"
#include <getopt.h>
#include <iostream>
#include "minerule/Utils/StringUtils.hpp"
#include "minerule/Utils/Converter.hpp"
#include "minerule/Utils/FileUtils.hpp"
```

Namespaces

- namespace mrprint

Functions

- void mrprint::printError (const std::string &error)

7.334 Options.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "Options.hpp"
00017 #include <getopt.h>
00018 #include <iostream>
00019
00020 #include "minerule/Utils/StringUtils.hpp"
00021 #include "minerule/Utils/Converter.hpp"
00022 #include "minerule/Utils/FileUtils.hpp"
00023
00024 namespace mrprint {
00025
00026         void printError(const std::string& error) {
00027                 std::cout « minerule::StringUtils::toRed("Error:") « error « std::endl;
00028         }
00029
00030         void Options::printHelp() const {
00031                 std::cout « minerule::StringUtils::toBold("Usage:") « std::endl
00032                         « "   " « minerule::StringUtils::toBold(_argv[0]) « " [-h] [-f <optionfile>]
      [-0] [-c] [-n minerule-number] [-s <order>] resultsetname" « std::endl
00033                         « "The program allows printing results of minerule queries." « std::endl
00034                         « std::endl
00035                         « minerule::StringUtils::toBold("-h") « " - prints this message " « std::endl
00036                         « minerule::StringUtils::toBold("-f") « " - Specifies a minerule option file
      (default: '"« minerule::MineruleOptions::DEFAULT_FILE_NAME « "')" « std::endl
00037                         « minerule::StringUtils::toBold("-c") « " - suppress colors " « std::endl
00038                         « minerule::StringUtils::toBold("-0") « " - suppresses logging artifacts " «
      std::endl
00039                         « minerule::StringUtils::toBold("-l") « " - do not filter out rules having low
      confidence" « std::endl
00040                         « minerule::StringUtils::toBold("-n") « " - specifies a query number to be
      printed (this is an alternative to" « std::endl
```

```
00041                         « "     specifying the query name)" « std::endl
00042                         « minerule::StringUtils::toBold("-s") « " - sorts the rules in a specified
      order." « std::endl
00043                         « "   supported orders are: " « std::endl
00044                         « "     'no' -> no particular order (fastest display)" « std::endl
00045                         « "     'scbh' -> order is support, confidence, body, head"«std::endl
00046                         « "     'bhsc' -> order is body, head, supp, conf"«std::endl
00047                         « "     'hbsc' -> order is head, body, supp,  conf"«std::endl
00048                         « "     'csbh' -> order is conf, supp, body, head" « std::endl
00049                         « "     'cbhs' -> order is conf, body, head, supp" « std::endl
00050                         « "     'cbsh' -> order is conf, body, supp, head" « std::endl
00051                         « "   the default is 'no'" « std::endl
00052                         « std::endl « std::endl;
00053         }
00054
00055         void Options::parse() {
00056                 const char* optstr = "hc0ln:s:";
00057
00058                 int opt;
00059                 while( (opt = getopt(_argc, _argv, optstr)) != -1 ) {
00060                         switch(opt) {
00061                                 case 'h':
00062                                 printHelp();
00063                                 break;
00064                                 case 'f':
00065                                 _mroptFileName = optarg;
00066                                 break;
00067                                 case 'c':
00068                                 minerule::StringUtils::setColorsEnabled(false);
00069                                 break;
00070                                 case '0':
00071                                 _noLogArtifacts=true;
00072                                 break;
00073                                 case 'l':
00074                                 _noLowConfidenceFilter=true;
00075                                 break;
00076                                 case 'n':
00077                                 _queryNumber = minerule::Converter(optarg).toLong();
00078                                 break;
00079                                 case 's':
00080                                 _sortOrder = optarg;
00081                                 break;
00082                                 default:
00083                                 std::cout « minerule::StringUtils::toRed("Error:") « " option " « opt
      « " not recognized." « std::endl;
00084                                 exit(MRPRINT_OPTION_PARSING_ERROR);
00085                         }
00086                 }
00087
00088                 if(optind == _argc-1) {
00089                         _queryName = _argv[optind];
00090                 }
00091
00092                 bool noQueryName = _queryName.empty() && _queryNumber == 0;
00093                 bool noOptionFile = !minerule::FileUtils::fileExists(_mroptFileName);
00094
00095                 if( noQueryName ) {
00096                         printError("no query name has been given. Please give one or use -n to specify
      a query number.");
00097                         exit(MRPRINT_OPTION_PARSING_ERROR);
00098                 }
00099
00100                 if(noOptionFile) {
00101                         printError("option file " + _mroptFileName + " not found.");
00102                         exit(MRPRINT_OPTION_PARSING_ERROR);
00103                 }
00104         }
00105
00106 }
```

7.335 /Users/esposito/Software/minerule/src/Programs/MRCatalogue/↩ Options.hpp File Reference

```
#include "Exception.hpp"
#include "minerule/Optimizer/CatalogueInstaller.hpp"
```

Data Structures

- class mrc::Options
- class mrc::Options::ListFormat

Namespaces

- namespace mrc

7.336 Options.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef OPTIONS_H_C27RB72E
00017 #define OPTIONS_H_C27RB72E
00018
00019 #include "Exception.hpp"
00020 #include "minerule/Optimizer/CatalogueInstaller.hpp"
00021
00022 namespace mrc {
00023
00024         class Options {
00025         public:
00026                 typedef enum {NoCommand,ShowList,SearchQry,DeleteQry, CheckCatalogue,
       InstallCatalogue, UninstallCatalogue, AddDerivedQuery} Command;
00027
00028                 class ListFormat {
00029                 public:
00030                         bool size;
00031                         bool text;
00032                         bool result;
00033
00034                         ListFormat() : size(false),       text(false), result(false) {}
00035                 };
00036
00037                 typedef enum {QryName=0,LastParam} QryParams;
00038
00039         protected:
00040                 Command command;
00041
00042                 std::string searchParam[LastParam];
00043                 std::string originalQuery;
00044                 std::string derivedQuery;
00045
00046                 ListFormat listFormat;
00047         public:
00048
00049                 Options():       command(NoCommand)  {  }
00050
00051                 void setListFormat(const char* format);
00052
00053                 void setShowList() {
00054                         if(command!=NoCommand && command!=ShowList) {
00055                                 throw Exception( mrc::ERROR_OPTION_PARSING,   "You can specify either
       -l or -n option, but not both!" );
00056                         }
00057
00058                         command = ShowList;
00059                 }
00060
00061                 void setSearchQry() {
00062                         if(command!=NoCommand && command!=SearchQry) {
```

```
00063                                   throw Exception( mrc::ERROR_OPTION_PARSING, "You can specify either -l
      or -n option, but not both!" );
00064                           }
00065
00066                           command = SearchQry;
00067                   }
00068
00069                   void setDeleteQry() {
00070                           if(command!=NoCommand && command!=DeleteQry) {
00071                                   throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00072                           }
00073
00074                           command = DeleteQry;
00075                   }
00076
00077                   void setCheckCatalogue() {
00078                           if(command!=NoCommand && command != CheckCatalogue) {
00079                                   throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00080                           }
00081
00082                           command = CheckCatalogue;
00083                   }
00084
00085                   void setInstallCatalogue() {
00086                           if(command!=NoCommand && command != InstallCatalogue) {
00087                                   throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00088                           }
00089
00090                           command = InstallCatalogue;
00091                   }
00092
00093                   void setUninstallCatalogue() {
00094                           if(command!=NoCommand && command != UninstallCatalogue) {
00095                                   throw Exception( mrc::ERROR_OPTION_PARSING, "Too many different
      options have been given, check out the parameters");
00096                           }
00097
00098                           command = UninstallCatalogue;
00099                   }
00100
00101                   void setSearchParam(QryParams qryParam, const std::string& param) {
00102                           searchParam[qryParam]=param;
00103                   }
00104
00105                   const std::string& getSearchParam(QryParams qryParam) const {
00106                           return searchParam[qryParam];
00107                   }
00108
00109                   void setAddDerivedQuery(const std::string& original, const std::string& derived) {
00110                           command = AddDerivedQuery;
00111                           originalQuery = original;
00112                           derivedQuery = derived;
00113                   }
00114
00115                   const std::string& getOriginalQuery() const {
00116                           return originalQuery;
00117                   }
00118
00119                   const std::string& getDerivedQuery() const {
00120                           return derivedQuery;
00121                   }
00122
00123                   Command getCommand() const {
00124                           return command;
00125                   }
00126
00127                   const ListFormat& getListFormat() const {
00128                           return listFormat;
00129                   }
00130           };
00131
00132 }; // namespace mrc
00133
00134 #endif /* end of include guard: OPTIONS_H_C27RB72E */
```

7.337 /Users/esposito/Software/minerule/src/Programs/MRMatch/↩ Options.hpp File Reference

```
#include <string>
#include <vector>
#include "mrmatch.hpp"
```

Data Structures

- class mrmatch::Options

Namespaces

- namespace mrmatch

7.338 Options.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef OPTIONS_H_U6THS9P
00017 #define OPTIONS_H_U6THS9P
00018
00019 #include <string>
00020 #include <vector>
00021
00022 #include "mrmatch.hpp"
00023
00024 namespace mrmatch {
00025         class Options {
00026         public:
00027                 Options() : _queryNumber(0), _matchKind(RuleGids), _matchOutput(OutOnConsole) {};
00028                 virtual ~Options () {};
00029
00030                 void setQueryName(std::string name) { _queryName   = name;          }
00031                 void setQueryNumber(size_t number)  { _queryNumber = number;  }
00032                 void setTableName(const std::string&);
00033                 const std::string& tableName() const;
00034                 std::string queryName() const;
00035
00036                 MatcherSpecs matcherSpecs() const { return _matchKind | _matchOutput; }
00037
00038                 void setMatchKind(MatchKind kind) { _matchKind = kind; }
00039                 void setMatchOutput(MatchOutput out) { _matchOutput = out; }
00040                 void setMatchOutputTableName(const std::string& name) { _matchOutputTableName = name;
      }
00041                 const std::string& getMatchOutputTableName() const { return _matchOutputTableName; }
00042
00043                 bool initMineruleOptions() const;
00044
00045                 void printUsage(int argc, char* const argv[]);
00046                 void parse(int argc, char* const argv[]);
00047         private:
00048                 std::string _mrOptionsFileName;
```

```
00049                 std::vector<std::string> _mrOptionsInline;
00050
00051                 std::string _queryName;
00052                 std::string _tableName;
00053                 size_t _queryNumber;
00054
00055                 MatchKind _matchKind;
00056                 MatchOutput _matchOutput;
00057                 std::string _matchOutputTableName;
00058
00059                 static MatchKind stringToMatchKind(const std::string&);
00060         };
00061
00062 }
00063
00064 #endif /* end of include guard: OPTIONS_H_U6THS9P */
```

7.339 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/Options.hpp File Reference

```
#include <string>
#include "minerule/Utils/MineruleOptions.hpp"
```

Data Structures

- class mrprint::Options

Namespaces

- namespace mrprint

Enumerations

- enum mrprint::MRPRINT_ERRORS { mrprint::MRPRINT_OPTION_PARSING_ERROR =1 }

Functions

- void mrprint::printError (const std::string &error)

7.340 Options.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef OPTIONS_H_IK79Q6DV
00017 #define OPTIONS_H_IK79Q6DV
00018
00019 #include <string>
00020 #include "minerule/Utils/MineruleOptions.hpp"
00021
00022 namespace mrprint {
00023         enum MRPRINT_ERRORS { MRPRINT_OPTION_PARSING_ERROR=1 };
00024
00025         void printError(const std::string& str);
00026
00027         class Options    {
00028         public:
00029                 Options(int argc, char** argv) :
00030                                 _argv(argv),
00031                                 _argc(argc),
00032                                 _mroptFileName(minerule::MineruleOptions::DEFAULT_FILE_NAME),
00033                                 _noLogArtifacts(false),
00034                                 _noLowConfidenceFilter(false),
00035                                 _sortOrder("no"),
00036                                 _queryNumber(0),
00037                                 _queryName("") {};
00038                 virtual ~Options () {};
00039
00040                 void parse();
00041                 void printHelp() const;
00042
00043                 // accessors
00044                 std::string mroptFileName() const { return _mroptFileName; }
00045                 bool noLogArtifacts() const { return _noLogArtifacts; }
00046                 bool noLowConfidenceFilter() const { return _noLowConfidenceFilter; }
00047                 std::string sortOrder() const { return _sortOrder; }
00048                 size_t queryNumber() const { return _queryNumber; }
00049                 std::string queryName() const { return _queryName; }
00050
00051
00052         private:
00053                 char** _argv;
00054                 int _argc;
00055
00056                 std::string _mroptFileName;
00057                 bool _noLogArtifacts;
00058                 bool _noLowConfidenceFilter;
00059                 std::string _sortOrder;
00060                 size_t _queryNumber;
00061                 std::string _queryName;
00062         };
00063
00064
00065 }
00066
00067 #endif /* end of include guard: OPTIONS_H_IK79Q6DV */
```

7.341 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.cpp File Reference

```
#include "RuleGidsDBMatcher.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
#include <memory>
```

Namespaces

- namespace mrmatch

7.342 RuleGidsDBMatcher.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "RuleGidsDBMatcher.hpp"
00017 #include "minerule/Utils/MineruleOptions.hpp"
00018 #include "minerule/mrdb/Connection.hpp"
00019 #include "minerule/mrdb/Statement.hpp"
00020 #include "minerule/mrdb/PreparedStatement.hpp"
00021 #include <memory>
00022
00023 namespace mrmatch {
00024         void RuleGidsDBMatcher::createOutputTable(const minerule::ItemType& item) const {
00025                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00026                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00027                 minerule::SourceRowMetaInfo metaInfo(con, _minerule);
00028
00029                 minerule::MRLog() « "Creating dest table: " « _outTableName « std::endl;
00030                 state->execute("\
00031                         CREATE TABLE " + _outTableName + " \
00032                                 (\
00033                                   bodyid integer,\
00034                                   headid integer," +
00035                                   metaInfo.getGroup().getSQLDataDefinition() + ");");
00036
00037                 std::string colNamesWithCommas = metaInfo.getGroup().getSQLColumnNames();
00038                 std::string colNamesWithUnderscores = minerule::StringUtils::join(
      minerule::StringUtils::split(colNamesWithCommas, ","), "_" );
00039
00040                 createOutputTableIndex(_outTableName + "_bodyhead_index", "(bodyid,headid)");
00041                 createOutputTableIndex(_outTableName + "_" +colNamesWithUnderscores + "_index", "("+
      colNamesWithCommas +")");
00042         }
00043
00044         void RuleGidsDBMatcher::createOutputTableIndex(const std::string& indexName, const
      std::string& indexCols) const {
00045                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00046                 std::auto_ptr<mrdb::Statement> state(con->createStatement());
00047
00048                 minerule::MRLog() « "Building index on body/head..." « std::endl;
00049                 state->execute("\
00050                         CREATE INDEX " + indexName +
00051                         " ON " + _outTableName + indexCols);
00052         }
00053
00054
00055         void RuleGidsDBMatcher::printMatches() const {
00056                 bool outTableCreated = false;
00057
00058                 minerule::MRLogPusher _("Saving matches onto the database...");
00059                 mrdb::Connection*  con =
      minerule::MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00060
00061                 minerule::MRLog() « "Inserting rules..." « std::endl;
00062                 std::auto_ptr<mrdb::PreparedStatement> ps(con->prepareStatement("\
00063                         INSERT INTO " + _outTableName + " VALUES (?,?,?);"));
00064
00065                 for(RuleGidsVector::const_iterator it=_ruleGidsVector.begin();
      it!=_ruleGidsVector.end(); ++it) {
00066                         Gids::const_iterator gids_it;
00067                         for( gids_it = it->second.begin(); gids_it!=it->second.end(); ++gids_it ) {
00068                                 if(!outTableCreated) {
00069                                         createOutputTable(*gids_it);
00070                                         outTableCreated=true;
00071                                 }
00072
00073                                 ps->setInt(1,it->first.getBodyId());
00074                                 ps->setInt(2,it->first.getHeadId());
00075                                 gids_it->setPreparedStatementParameters(ps.get(),3);
```

```
00076                                   ps->execute();
00077                           }
00078                   }
00079
00080                   minerule::MRLog() « "All done!" « std::endl;
00081           }
00082 }
```

7.343 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsDBMatcher.hpp File Reference

```
#include "RuleGidsMatcher.hpp"
```

Data Structures

- class mrmatch::RuleGidsDBMatcher

Namespaces

- namespace mrmatch

7.344 RuleGidsDBMatcher.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef RULEGIDSDBMATCHER_H_45M76FV9
00017 #define RULEGIDSDBMATCHER_H_45M76FV9
00018
00019 #include "RuleGidsMatcher.hpp"
00020
00021 namespace mrmatch {
00022         class RuleGidsDBMatcher : public RuleGidsMatcher {
00023         protected:
00024                 std::string _outTableName;
00025                 minerule::ParsedMinerule _minerule;
00026
00027                 void createOutputTable(const minerule::ItemType& item) const;
00028                 void createOutputTableIndex(const std::string& indexName, const std::string&
       indexCols) const;
00029         public:
00030                 RuleGidsDBMatcher(const std::string& outTableName, const minerule::ParsedMinerule&
       minerule)
00031                         : RuleGidsMatcher(), _outTableName(outTableName), _minerule(minerule) {}
00032                 virtual ~RuleGidsDBMatcher() {}
00033
00034                 virtual void printMatches() const;
00035         };
00036 }
00037
00038 #endif /* end of include guard: RULEGIDSDBMATCHER_H_45M76FV9 */
00039
00040
```

7.345 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.cpp File Reference

```
#include "RuleGidsMatcher.hpp"
#include "minerule/Database/Transaction.hpp"
#include "minerule/Result/RulesMatcher.hpp"
#include "minerule/Result/RuleFormatter.hpp"
```

Namespaces

- namespace mrmatch

7.346 RuleGidsMatcher.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "RuleGidsMatcher.hpp"
00017 #include "minerule/Database/Transaction.hpp"
00018 #include "minerule/Result/RulesMatcher.hpp"
00019 #include "minerule/Result/RuleFormatter.hpp"
00020
00021 using namespace minerule;
00022
00023 namespace mrmatch {
00024
00025         // create a new empty rule and returns it (so that it can be intialized)
00026         Rule& RuleGidsMatcher::addRule() {
00027                 _ruleGidsVector.push_back( RuleGids() );
00028                 return _ruleGidsVector.back().first;
00029         }
00030
00031         void RuleGidsMatcher::matchItemTransaction(const ItemType& gid, const
      ItemTransaction<RulesMatcher::ItemSetType>& bodies,const ItemTransaction<RulesMatcher::ItemSetType>&
      heads) {
00032                         for(RuleGidsVector::iterator ruleIt = _ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00033                                 if( RulesMatcher::match( ruleIt->first, bodies, heads ) ) {
00034                                         ruleIt->second.push_back( gid );
00035                                 }
00036                         }
00037         }
00038
00039         void RuleGidsMatcher::matchRuleTransaction(const minerule::ItemType& gid, const
      RuleTransaction<RulesMatcher::RuleSetType>& transaction) {
00040                         for(RuleGidsVector::iterator ruleIt=_ruleGidsVector.begin();
      ruleIt!=_ruleGidsVector.end(); ++ruleIt) {
00041                                 if( RulesMatcher::match(ruleIt->first, transaction) ) {
00042                                         ruleIt->second.push_back( gid );
00043                                 }
00044                         }
00045         }
00046
00047
00048
00049         std::string RuleGidsMatcher::formatGids( const Gids& gids ) const {
00050                 std::stringstream str;
00051                 for(std::vector<ItemType>::const_iterator it = gids.begin(); it!=gids.end(); ++it) {
```

```
00052                         str « *it « " ";
00053                 }
00054
00055                 return StringUtils::toBold(StringUtils::toGreen(str.str()));
00056         }
00057
00058         void RuleGidsMatcher::printMatches( ) const {
00059                 SimpleRuleFormatter sf;
00060                 sf.setFieldWidths( SimpleRuleFormatter::FieldWidths(1,1,1,1) );
00061                 for(RuleGidsVector::const_iterator it=_ruleGidsVector.begin();
      it!=_ruleGidsVector.end(); ++it) {
00062                         MRLog()    « StringUtils::toBold("Rule: ") « sf.formatRule(it->first) « " "
00063                                                 « StringUtils::toBold("\tGids: ") «
      formatGids(it->second) « std::endl;
00064                 }
00065         }
00066
00067
00068
00069 }
```

7.347 /Users/esposito/Software/minerule/src/Programs/MRMatch/RuleGidsMatcher.hpp File Reference

```
#include "Matcher.hpp"
```

Data Structures

- class mrmatch::RuleGidsMatcher

Namespaces

- namespace mrmatch

7.348 RuleGidsMatcher.hpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #ifndef RULEGIDMATCHER_H_X9STRBC4
00017 #define RULEGIDMATCHER_H_X9STRBC4
00018
00019 #include "Matcher.hpp"
00020
00021 namespace mrmatch {
00022
00023         class RuleGidsMatcher : public Matcher {
00024         protected:
00025                 typedef std::vector<minerule::ItemType> Gids;
00026                 typedef std::pair< minerule::Rule, Gids > RuleGids;
00027                 typedef std::vector< RuleGids > RuleGidsVector;
```

```
00028                 RuleGidsVector _ruleGidsVector;
00029
00030                 virtual void matchItemTransaction(const minerule::ItemType& gid, const
      minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& bodies,const
      minerule::ItemTransaction<minerule::RulesMatcher::ItemSetType>& heads);
00031                 virtual void matchRuleTransaction(const minerule::ItemType& gid, const
      minerule::RuleTransaction<minerule::RulesMatcher::RuleSetType>& transaction);
00032
00033                 std::string formatGids( const Gids& gids ) const;
00034
00035         public:
00036                 RuleGidsMatcher() {}
00037                 virtual ~RuleGidsMatcher() {}
00038
00039         // create a new empty rule and returns it (so that it can be intialized)
00040                 virtual minerule::Rule& addRule();
00041
00042         // outputs the results
00043                 virtual void printMatches() const;
00044
00045         };
00046
00047 }
00048
00049 #endif /* end of include guard: RULEGIDMATCHER_H_X9STRBC4 */
```

7.349 /Users/esposito/Software/minerule/src/Programs/MRPrintRules/↩ PrintRules.cpp File Reference

```
#include <string.h>
#include "minerule/mrdb/ResultSet.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/Statement.hpp"
#include "minerule/mrdb/PreparedStatement.hpp"
#include <iostream>
#include <assert.h>
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Database/SourceRowElement.hpp"
#include "minerule/Utils/FileUtils.hpp"
#include "minerule/Result/RuleFormatter.hpp"
#include "Options.hpp"
```

Namespaces

- namespace mrprint

Functions

- void mrprint::printRules (std::string queryname, RuleFormatter &formatter, double conf)
- std::string mrprint::getQueryName (const Options &options)
- RuleFormatter * mrprint::newFormatter (const Options &options)
- int main (int argc, char **argv)

7.349.1 Function Documentation

7.349.1.1 main()

```
int main (
              int argc,
              char ** argv )
```

Definition at line 97 of file PrintRules.cpp.

```
00097                                   {
00098         try {
00099                 MineruleOptions& mr = MineruleOptions::getSharedOptions();
00100
00101                 RuleFormatter* rf=NULL;
00102                 double conf = -1;
00103
00104                 mrprint::Options options(argc, argv);
00105                 options.parse();
00106
00107                 mr.readFromFile(options.mroptFileName());
00108
00109                 rf = newFormatter(options);
00110                 if( options.noLowConfidenceFilter() ) {
00111                         conf = 0.0;
00112                 }
00113
00114
00115                 std::string queryName = getQueryName(options);
00116
00117                 if(!rf->suppressLog()) MRLogPush("Printing result set...");
00118
00119                 mrprint::printRules( queryName, *rf, conf);
00120
00121                 if(!rf->suppressLog()) MRLogPop();
00122
00123
00124                 delete rf;
00125         } catch ( minerule::MineruleException& e ) {
00126                 MRErr() << "MineruleError:" << e.what() << std::endl;
00127                 throw;
00128         }  catch( mrdb::SQLException& e) {
00129                 MRErr() << "SQLError:" << e.what() << std::endl;
00130                 throw;
00131         }  catch (std::exception& e) {
00132                 MRErr() << "Couldn't execute your request, the reason is:"
00133                         << e.what() << std::endl;
00134                 throw;
00135         }  catch (...) {
00136                 MRErr() << "Uncought exception!" << std::endl;
00137                 throw;
00138         }
00139
00140         return 0;
00141 }
```

7.350 PrintRules.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <string.h>
00017 #include "minerule/mrdb/ResultSet.hpp"
00018 #include "minerule/mrdb/Connection.hpp"
00019 #include "minerule/mrdb/Statement.hpp"
00020 #include "minerule/mrdb/PreparedStatement.hpp"
00021
```

```
00022 #include <iostream>
00023 #include <assert.h>
00024
00025 #include "minerule/Utils/MineruleOptions.hpp"
00026 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00027 #include "minerule/Database/SourceRowElement.hpp"
00028 #include "minerule/Utils/FileUtils.hpp"
00029 #include "minerule/Result/RuleFormatter.hpp"
00030
00031 #include "Options.hpp"
00032
00033
00034 using namespace minerule;
00035
00036 namespace mrprint {
00037
00038         void
00039         printRules(std::string queryname,     RuleFormatter& formatter, double conf) {
00040                 QueryResult::Iterator qit;
00041                 OptimizerCatalogue::getMRQueryResultIterator( queryname, qit, -1, conf );
00042
00043                 while( qit.next() ) {
00044                         Rule r;
00045                         qit.getRule(r);
00046
00047                         formatter.printRule(r);
00048                 }
00049
00050                 formatter.postExec();
00051         }
00052
00053
00054         std::string getQueryName(const Options& options) {
00055                 if( !options.queryName().empty() )
00056                         return options.queryName();
00057
00058                 return minerule::OptimizerCatalogue::getMRQueryName(options.queryNumber());
00059         }
00060
00061
00062         RuleFormatter* newFormatter(const Options& options) {
00063                 RuleFormatter* rf = NULL;
00064                 std::string sortOrder = options.sortOrder();
00065
00066                 if(sortOrder=="no") {
00067                                 rf=new SimpleRuleFormatter();
00068                 } else if(sortOrder=="scbh") {
00069                                 rf=new SortedRuleFormatter<QueryResult::SortSuppConfBodyHead>();
00070                 } else if(sortOrder=="bhsc") {
00071                                 rf=new SortedRuleFormatter<QueryResult::SortBodyHeadSuppConf>();
00072                 } else if(sortOrder=="hbsc") {
00073                                 rf=new SortedRuleFormatter<QueryResult::SortHeadBodySuppConf>();
00074                 } else if(sortOrder=="csbh") {
00075                                 rf=new SortedRuleFormatter<QueryResult::SortConfSuppBodyHead>();
00076                 } else if(sortOrder=="cbhs") {
00077                                 rf=new SortedRuleFormatter<QueryResult::SortConfBodyHeadSupp>();
00078                 } else if(sortOrder=="cbsh") {
00079                                 rf=new SortedRuleFormatter<QueryResult::SortConfBodySuppHead>();
00080                 } else {
00081                                 printError("Unsupported sort order:" + sortOrder);
00082                                 exit(MRPRINT_OPTION_PARSING_ERROR);
00083                 }
00084
00085                 if(rf==NULL) {
00086                         rf = new SimpleRuleFormatter();
00087                 }
00088
00089                 if( options.noLogArtifacts() )
00090                         rf->setSuppressLog(true);
00091
00092                 return rf;
00093         }
00094
00095 }
00096
00097 int main(int argc, char** argv) {
00098         try {
00099                 MineruleOptions& mr = MineruleOptions::getSharedOptions();
00100
00101                 RuleFormatter* rf=NULL;
00102                 double conf = -1;
00103
00104                 mrprint::Options options(argc, argv);
00105                 options.parse();
00106
00107                 mr.readFromFile(options.mroptFileName());
00108
```

```
00109                 rf = newFormatter(options);
00110                 if( options.noLowConfidenceFilter() ) {
00111                         conf = 0.0;
00112                 }
00113
00114
00115                 std::string queryName = getQueryName(options);
00116
00117                 if(!rf->suppressLog()) MRLogPush("Printing result set...");
00118
00119                 mrprint::printRules( queryName, *rf, conf);
00120
00121                 if(!rf->suppressLog()) MRLogPop();
00122
00123
00124                 delete rf;
00125         } catch ( minerule::MineruleException& e ) {
00126                 MRErr() << "MineruleError:" << e.what() << std::endl;
00127                 throw;
00128         }  catch( mrdb::SQLException& e) {
00129                 MRErr() << "SQLError:" << e.what() << std::endl;
00130                 throw;
00131         }  catch (std::exception& e) {
00132                 MRErr() << "Couldn't execute your request, the reason is:"
00133                         << e.what() << std::endl;
00134                 throw;
00135         }  catch (...) {
00136                 MRErr() << "Uncought exception!" << std::endl;
00137                 throw;
00138         }
00139
00140         return 0;
00141 }
```

7.351 /Users/esposito/Software/minerule/src/Programs/MRQuery/mrquery.cpp File Reference

```
#include <iostream>
#include <string>
#include <fstream>
#include <getopt.h>
#include "minerule/Utils/StringUtils.hpp"
#include "minerule/Utils/MineruleErrors.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Utils/FileUtils.hpp"
#include "minerule/Optimizer/OptimizedMinerule.hpp"
#include "minerule/Algorithms/Algorithms.hpp"
```

Functions

- void printHelp (char *progName)
- void printVersion ()
- std::string parseMineruleName (const std::string &mrtext)
- void parseOptions (int argc, char **argv, MineruleOptions &mrOpts, std::string &mrtext)
- void execQuery (int argc, char **argv)
- int main (int argc, char *argv[])

7.351.1 Function Documentation

7.351.1.1 execQuery()

```
void execQuery (
             int argc,
             char ** argv )
```

Definition at line 230 of file mrquery.cpp.

```
00230                                        {
00231         MineruleOptions& mrOpts =  MineruleOptions::getSharedOptions();
00232         std::string mrtext;
00233
00234         parseOptions(argc, argv, mrOpts, mrtext);
00235
00236         MRLogPush("Initialization...");
00237         MRLog() « "Minerule source text:" « std::endl;
00238         MRLog(mrtext);
00239         MRLogPop();
00240
00241         MRLogPush("Minerule system starting");
00242
00243         OptimizedMinerule optMR(mrtext);
00244         Algorithms::executeMinerule(optMR);
00245
00246         MRLogPop();
00247
00248         MRLogShowMeasurements();
00249
00250 }
```

7.351.1.2 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 252 of file mrquery.cpp.

```
00253 {
00254         try {
00255                 execQuery(argc, argv);
00256         } catch (mrdb::SQLException& e) {
00257                 MRErr() « "MRDB Exception:" « e.what() « std::endl;
00258                 exit(MR_ERROR_DATABASE_ERROR);
00259         } catch (MineruleException& e) {
00260                 MRErr() « e.what() « std::endl;
00261                 exit( e.getErrorCode() );
00262         } catch (std::exception& e) {
00263                 MRErr() « "Exception:" « e.what() « std::endl;
00264                 exit( MR_ERROR_UNKNOWN );
00265         }       catch(...) {
00266                 MRErr() « "An unknown exception has been thrown..." « std::endl;
00267                 exit( MR_ERROR_UNKNOWN );
00268         }
00269
00270         return 0;
00271 }
```

7.351.1.3 parseMineruleName()

```
std::string parseMineruleName (
             const std::string & mrtext )
```

Definition at line 69 of file mrquery.cpp.

```
00069                                                {
00070         std::string mrtextcopy = mrtext;
00071         std::string spacechars = " \t\n";
```

```
00072
00073         std::string mstringrule="mine rule ";
00074         std::string mstringsequence="mine sequence";
00075         std::string mstringitemset="mine itemset";
00076         std::string mstring;
00077
00078         size_t minerulepos;
00079         size_t startnamepos = std::string::npos;
00080         size_t endnamepos = std::string::npos;
00081
00082   // strlwr seems to be not standard and hence not
00083   // present in all systems. The following substitue it
00084   // (it is needed in order to implement case insensitivity)
00085         for(size_t i=0; i<mrtext.size(); i++) {
00086                 mrtextcopy[i]=tolower(mrtextcopy[i]);
00087         }
00088
00089         if(mrtextcopy.find(mstringrule,0)!=std::string::npos)
00090                 mstring=mstringrule;
00091         else if(mrtextcopy.find(mstringitemset,0)!=std::string::npos)
00092                 mstring=mstringitemset;
00093         else
00094                 mstring=mstringsequence;
00095
00096         minerulepos = mrtextcopy.find(mstring,0);
00097
00098         if(minerulepos!=std::string::npos)
00099                 startnamepos = mrtextcopy.find_first_not_of(spacechars ,
00100                         minerulepos+mstring.size());
00101
00102         if(startnamepos!=std::string::npos)
00103                 endnamepos = mrtextcopy.find_first_of(spacechars,startnamepos);
00104
00105
00106         if(endnamepos==std::string::npos ) {
00107     // notice we will be here, if any of the searches above fails
00108                 MRErr() « "Error in retrieving the minerule name from the "
00109                         « "text which define the minerule. I could not "
00110                         « "initialize, then, the text for %m parameter. "
00111                         « "I'm now switching to the default 'mrnameerror' mine rule name" « std::endl
00112                         « "NOTE: If the minerule main parser succeed in parsing the" « std::endl
00113                         « " minerule text, then, this message is a BUG and should be"«std::endl
00114                         « " reported!" « std::endl;
00115                 return "mrnameerror";
00116         }
00117
00118         return mrtext.substr(startnamepos, endnamepos-startnamepos);
00119 }
```

7.351.1.4 parseOptions()

```
void parseOptions (
             int argc,
             char ** argv,
             MineruleOptions & mrOpts,
             std::string & mrtext )
```

Definition at line 123 of file mrquery.cpp.

```
00123                                                                                {
00124         int opt;
00125         bool okMRText=false;
00126         bool okMROptions=false;
00127         std::vector<std::string> cmd_line_mr_options;
00128
00129         while((opt=getopt(argc,argv,"chi:f:m:O:ev"))!=-1) {
00130                 switch(opt) {
00131                         case 'c':
00132                                 StringUtils::setColorsEnabled(false);
00133                                 break;
00134                         case 'h':
00135                                 printVersion();
00136                                 printHelp(argv[0]);
00137                                 exit(0);
00138                         case 'v':
00139                                 printVersion();
00140                                 exit(0);
00141                         case 'i':
```

```
00142                           {
00143                                   if(okMRText) {
00144                                           MRErr() « "More than one minerule has been specified! "
00145                                                   "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00146
00147                                           std::cerr  « "More than one minerule has been specified! "
00148                                                   "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00149                                           exit(MR_ERROR_OPTION_PARSING);
00150                                   }
00151                                   std::ifstream infile(optarg);
00152                                   if(!infile) {
00153                                           MRErr() « "Cannot open input file" « std::endl;
00154                                           std::cerr «  "Cannot open input file" « std::endl;
00155                                           exit(MR_ERROR_INPUT_FILE_NOT_FOUND);
00156                                   }
00157                                   std::string buf;
00158                                   mrtext = "";
00159                                   while( infile » buf ) {
00160                                           mrtext += " " + buf;
00161                                   }
00162
00163                                   okMRText=true;
00164                                   mrOpts.setMineruleName(parseMineruleName(mrtext));
00165                                   mrOpts.setMineruleSourceName(optarg);
00166                           }
00167                           break;
00168                           case 'f':
00169                                   if( FileUtils::fileExists(optarg) ) {
00170                                           mrOpts.readFromFile(optarg);
00171                                           MRLog() « "Options read from '" « optarg «"'." « std::endl;
00172                                   } else {
00173                                           MRLog() « optarg « " not found, starting with default Options"
      « std::endl;
00174                                   }
00175                                   okMROptions = true;
00176                                   break;
00177                           case 'O':
00178                                   cmd_line_mr_options.push_back(std::string(optarg));
00179                                   break;
00180                           case 'm':
00181                                   if( okMRText ) {
00182                                           MRErr() « "More than one minerule has been specified! "
00183                                                   "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00184
00185                                           std::cerr  « "More than one minerule has been specified! "
00186                                                   "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00187                                           exit(MR_ERROR_OPTION_PARSING);
00188                                   }
00189                                   mrtext=optarg;
00190                                   okMRText=true;
00191                                   mrOpts.setMineruleName(parseMineruleName(mrtext));
00192                                   mrOpts.setMineruleSourceName("MRFromSTDIN");
00193                                   break;
00194                           default:
00195                                   MRErr() « "Option not recognized:" « (char)opt « std::endl;
00196                                   std::cerr «  "Option not recognized:" « (char)opt « std::endl;
00197                                   exit(MR_ERROR_OPTION_PARSING);
00198                   }
00199           }
00200
00201           if(!okMRText) {
00202                   MRErr() « "No minerule has been specified (use -i or -m options)" « std::endl;
00203                   exit(MR_ERROR_NO_MINERULE_SPECIFIED);
00204           }
00205
00206           if(!okMROptions) {
00207                   if( FileUtils::fileExists(MineruleOptions::DEFAULT_FILE_NAME) ) {
00208                           mrOpts.readFromFile(MineruleOptions::DEFAULT_FILE_NAME);
00209                           MRLog() « "Options read from:" « MineruleOptions::DEFAULT_FILE_NAME «
      std::endl;
00210                           okMROptions = true;
00211                   }
00212
00213                   if( !cmd_line_mr_options.empty() ) {
00214                           for(std::vector<std::string>::const_iterator it=cmd_line_mr_options.begin();
      it!=cmd_line_mr_options.end(); ++it) {
00215                                   mrOpts.readFromString(*it);
00216                                   MRLog() « "Options set accordingly to command line paramenter:-O " «
      *it « std::endl;
00217                           }
00218                           okMROptions = true;
00219                   }
00220
```

```
00221
00222                 if(!okMROptions) {
00223                         MRErr() « "No option file specified or found.";
00224                         exit(MR_ERROR_NO_OPTIONFILE_SPECIFIED);
00225                 }
00226         }
00227 }
```

7.351.1.5 printHelp()

```
void printHelp (
             char * progName )
```

Definition at line 33 of file mrquery.cpp.

```
00033                         {
00034         std::cout « StringUtils::toBold("Usage:") « std::endl
00035                 « "   " « StringUtils::toBold(progName) « " [-c] [-i <mineruletextfile>] [-m
      '<mineruletext>'] " « std::endl
00036                 « "      -f <mineruleoptionfile> -O <optionlist> -v -h " « std::endl « std::endl;
00037
00038         std::cout
00039                 « StringUtils::toBold("   -c") « " -- disable colors in messages" « std::endl
00040                 « StringUtils::toBold("   -i") « " -- specify a file name containing the text of the
      minerule" « std::endl
00041                 « StringUtils::toBold("   -m") « " -- the argument is the text of the minerule" «
      std::endl
00042                 « StringUtils::toBold("   -f") « " -- specify a file name containing the Options to be
      used." « std::endl
00043                 « "         more than one is allowed and later ones override the options"« std::endl
00044                 « "         read from previous ones" « std::endl
00045                 « StringUtils::toBold("   -O") « " -- Allows to specify some options from the command
      line" « std::endl
00046                 « "         whether those options overrides or not the ones given by -f"«std::endl
00047                 « "         commands depends on which comes first" «std::endl
00048                 « StringUtils::toBold("   -v") « " -- version informations" « std::endl
00049                 « StringUtils::toBold("   -h") « " -- this message" « std::endl
00050                 « StringUtils::toBold(" exit codes:") « std::endl;
00051         for(int i=me_error_begin(); i<me_error_end(); i++) {
00052                 std::cout
00053                 «  "\t" « i « "\t- " « me_error_name((MineruleErrors) i) « std::endl;
00054         }
00055
00056         exit(0);
00057 }
```

7.351.1.6 printVersion()

```
void printVersion ( )
```

Definition at line 59 of file mrquery.cpp.

```
00059                         {
00060         std::cout « StringUtils::toBold("Minerule v:") « MR_VERSION « std::endl;
00061 }
```

7.352 mrquery.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
```

```
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <iostream>
00017 #include <string>
00018 #include <fstream>
00019 #include <getopt.h>
00020
00021 #include "minerule/Utils/StringUtils.hpp"
00022 #include "minerule/Utils/MineruleErrors.hpp"
00023 #include "minerule/Utils/MineruleLogs.hpp"
00024 #include "minerule/Utils/MineruleOptions.hpp"
00025 #include "minerule/Utils/FileUtils.hpp"
00026 #include "minerule/Optimizer/OptimizedMinerule.hpp"
00027 #include "minerule/Algorithms/Algorithms.hpp"
00028
00029 using namespace minerule;
00030
00031
00032 void
00033 printHelp(char* progName) {
00034         std::cout << StringUtils::toBold("Usage:") << std::endl
00035                   << "   " << StringUtils::toBold(progName) << " [-c] [-i <mineruletextfile>] [-m
      '<mineruletext>'] " << std::endl
00036                   << "      -f <mineruleoptionfile> -O <optionlist> -v -h " << std::endl << std::endl;
00037
00038         std::cout
00039                   << StringUtils::toBold("   -c") << " -- disable colors in messages" << std::endl
00040                   << StringUtils::toBold("   -i") << " -- specify a file name containing the text of the
      minerule" << std::endl
00041                   << StringUtils::toBold("   -m") << " -- the argument is the text of the minerule" <<
      std::endl
00042                   << StringUtils::toBold("   -f") << " -- specify a file name containing the Options to be
      used." << std::endl
00043                   << "          more than one is allowed and later ones override the options"<< std::endl
00044                   << "          read from previous ones" << std::endl
00045                   << StringUtils::toBold("   -O") << " -- Allows to specify some options from the command
      line" << std::endl
00046                   << "          whether those options overrides or not the ones given by -f"<<std::endl
00047                   << "          commands depends on which comes first" <<std::endl
00048                   << StringUtils::toBold("   -v") << " -- version informations" << std::endl
00049                   << StringUtils::toBold("   -h") << " -- this message" << std::endl
00050                   << StringUtils::toBold(" exit codes:") << std::endl;
00051         for(int i=me_error_begin(); i<me_error_end(); i++) {
00052                   std::cout
00053                   <<  "\t" << i << "\t- " << me_error_name((MineruleErrors) i) << std::endl;
00054         }
00055
00056         exit(0);
00057 }
00058
00059 void printVersion() {
00060         std::cout << StringUtils::toBold("Minerule v:") << MR_VERSION << std::endl;
00061 }
00062
00063
00064 /*
00065 * Tries to parse mrtext in order to find the name of
00066         * the current minerule.
00067  */
00068 std::string
00069 parseMineruleName(const std::string& mrtext) {
00070         std::string mrtextcopy = mrtext;
00071         std::string spacechars = " \t\n";
00072
00073         std::string mstringrule="mine rule ";
00074         std::string mstringsequence="mine sequence";
00075         std::string mstringitemset="mine itemset";
00076         std::string mstring;
00077
00078         size_t minerulepos;
00079         size_t startnamepos = std::string::npos;
00080         size_t endnamepos = std::string::npos;
00081
00082   // strlwr seems to be not standard and hence not
00083   // present in all systems. The following substitue it
00084   // (it is needed in order to implement case insensitivity)
00085         for(size_t i=0; i<mrtext.size(); i++) {
00086                   mrtextcopy[i]=tolower(mrtextcopy[i]);
00087         }
00088
00089         if(mrtextcopy.find(mstringrule,0)!=std::string::npos)
```

```
00090                 mstring=mstringrule;
00091         else if(mrtextcopy.find(mstringitemset,0)!=std::string::npos)
00092                 mstring=mstringitemset;
00093         else
00094                 mstring=mstringsequence;
00095
00096         minerulepos = mrtextcopy.find(mstring,0);
00097
00098         if(minerulepos!=std::string::npos)
00099                 startnamepos = mrtextcopy.find_first_not_of(spacechars ,
00100                         minerulepos+mstring.size());
00101
00102         if(startnamepos!=std::string::npos)
00103                 endnamepos = mrtextcopy.find_first_of(spacechars,startnamepos);
00104
00105
00106         if(endnamepos==std::string::npos ) {
00107     // notice we will be here, if any of the searches above fails
00108                 MRErr() « "Error in retrieving the minerule name from the "
00109                         « "text which define the minerule. I could not "
00110                         « "initialize, then, the text for %m parameter. "
00111                         « "I'm now switching to the default 'mrnameerror' mine rule name" « std::endl
00112                         « "NOTE: If the minerule main parser succeed in parsing the" « std::endl
00113                         « " minerule text, then, this message is a BUG and should be"«std::endl
00114                         « " reported!" « std::endl;
00115                 return "mrnameerror";
00116         }
00117
00118         return mrtext.substr(startnamepos, endnamepos-startnamepos);
00119 }
00120
00121
00122 void
00123 parseOptions(int argc, char** argv, MineruleOptions& mrOpts, std::string& mrtext) {
00124         int opt;
00125         bool okMRText=false;
00126         bool okMROptions=false;
00127         std::vector<std::string> cmd_line_mr_options;
00128
00129         while((opt=getopt(argc,argv,"chi:f:m:O:ev"))!=-1) {
00130                 switch(opt) {
00131                         case 'c':
00132                                 StringUtils::setColorsEnabled(false);
00133                                 break;
00134                         case 'h':
00135                                 printVersion();
00136                                 printHelp(argv[0]);
00137                                 exit(0);
00138                         case 'v':
00139                                 printVersion();
00140                                 exit(0);
00141                         case 'i':
00142                         {
00143                                 if(okMRText) {
00144                                         MRErr() « "More than one minerule has been specified! "
00145                                                 "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00146
00147                                         std::cerr  « "More than one minerule has been specified! "
00148                                                 "(you cannot specify more than one {-i,-m} options)" «
      std::endl;
00149                                         exit(MR_ERROR_OPTION_PARSING);
00150                                 }
00151                                 std::ifstream infile(optarg);
00152                                 if(!infile) {
00153                                         MRErr() « "Cannot open input file" « std::endl;
00154                                         std::cerr «  "Cannot open input file" « std::endl;
00155                                         exit(MR_ERROR_INPUT_FILE_NOT_FOUND);
00156                                 }
00157                                 std::string buf;
00158                                 mrtext = "";
00159                                 while( infile » buf ) {
00160                                         mrtext += " " + buf;
00161                                 }
00162
00163                                 okMRText=true;
00164                                 mrOpts.setMineruleName(parseMineruleName(mrtext));
00165                                 mrOpts.setMineruleSourceName(optarg);
00166                         }
00167                         break;
00168                         case 'f':
00169                                 if( FileUtils::fileExists(optarg) ) {
00170                                         mrOpts.readFromFile(optarg);
00171                                         MRLog() « "Options read from '" « optarg «"'." « std::endl;
00172                                 } else {
00173                                         MRLog() « optarg « " not found, starting with default Options"
      « std::endl;
```

```
00174                                   }
00175                                   okMROptions = true;
00176                                   break;
00177                           case 'O':
00178                                   cmd_line_mr_options.push_back(std::string(optarg));
00179                                   break;
00180                           case 'm':
00181                                   if( okMRText ) {
00182                                           MRErr() << "More than one minerule has been specified! "
00183                                                   "(you cannot specify more than one {-i,-m} options)" <<
      std::endl;
00184
00185                                           std::cerr  << "More than one minerule has been specified! "
00186                                                   "(you cannot specify more than one {-i,-m} options)" <<
      std::endl;
00187                                           exit(MR_ERROR_OPTION_PARSING);
00188                                   }
00189                                   mrtext=optarg;
00190                                   okMRText=true;
00191                                   mrOpts.setMineruleName(parseMineruleName(mrtext));
00192                                   mrOpts.setMineruleSourceName("MRFromSTDIN");
00193                                   break;
00194                           default:
00195                                   MRErr() << "Option not recognized:" << (char)opt << std::endl;
00196                                   std::cerr <<  "Option not recognized:" << (char)opt << std::endl;
00197                                   exit(MR_ERROR_OPTION_PARSING);
00198                   }
00199           }
00200
00201           if(!okMRText) {
00202                   MRErr() << "No minerule has been specified (use -i or -m options)" << std::endl;
00203                   exit(MR_ERROR_NO_MINERULE_SPECIFIED);
00204           }
00205
00206           if(!okMROptions) {
00207                   if( FileUtils::fileExists(MineruleOptions::DEFAULT_FILE_NAME) ) {
00208                           mrOpts.readFromFile(MineruleOptions::DEFAULT_FILE_NAME);
00209                           MRLog() << "Options read from:" << MineruleOptions::DEFAULT_FILE_NAME <<
      std::endl;
00210                           okMROptions = true;
00211                   }
00212
00213                   if( !cmd_line_mr_options.empty() ) {
00214                           for(std::vector<std::string>::const_iterator it=cmd_line_mr_options.begin();
      it!=cmd_line_mr_options.end(); ++it) {
00215                                   mrOpts.readFromString(*it);
00216                                   MRLog() << "Options set accordingly to command line paramenter:-O " <<
      *it << std::endl;
00217                           }
00218                           okMROptions = true;
00219                   }
00220
00221
00222                   if(!okMROptions) {
00223                           MRErr() << "No option file specified or found.";
00224                           exit(MR_ERROR_NO_OPTIONFILE_SPECIFIED);
00225                   }
00226           }
00227 }
00228
00229
00230 void execQuery(int argc, char** argv) {
00231           MineruleOptions& mrOpts =  MineruleOptions::getSharedOptions();
00232           std::string mrtext;
00233
00234           parseOptions(argc, argv, mrOpts, mrtext);
00235
00236           MRLogPush("Initialization...");
00237           MRLog() << "Minerule source text:" << std::endl;
00238           MRLog(mrtext);
00239           MRLogPop();
00240
00241           MRLogPush("Minerule system starting");
00242
00243           OptimizedMinerule optMR(mrtext);
00244           Algorithms::executeMinerule(optMR);
00245
00246           MRLogPop();
00247
00248           MRLogShowMeasurements();
00249
00250 }
00251
00252 int main (int argc, char *argv[])
00253 {
00254           try {
00255                   execQuery(argc, argv);
```

```
00256         } catch (mrdb::SQLException& e) {
00257                 MRErr() « "MRDB Exception:" « e.what() « std::endl;
00258                 exit(MR_ERROR_DATABASE_ERROR);
00259         } catch (MineruleException& e) {
00260                 MRErr() « e.what() « std::endl;
00261                 exit( e.getErrorCode() );
00262         } catch (std::exception& e) {
00263                 MRErr() « "Exception:" « e.what() « std::endl;
00264                 exit( MR_ERROR_UNKNOWN );
00265         }       catch(...) {
00266                 MRErr() « "An unknown exception has been thrown..." « std::endl;
00267                 exit( MR_ERROR_UNKNOWN );
00268         }
00269
00270         return 0;
00271 }
```

7.353 /Users/esposito/Software/minerule/src/Result/QueryResult.cpp File Reference

```
#include "minerule/Optimizer/OptimizerCatalogue.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Database/Connection.hpp"
```

Namespaces

- namespace minerule

7.354 QueryResult.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Optimizer/OptimizerCatalogue.hpp"
00017 #include "minerule/Utils/MineruleOptions.hpp"
00018 #include "minerule/Database/Connection.hpp"
00019
00020 namespace minerule {
00021
00022   void
00023   QueryResult::Iterator::readElems(int id, ItemSet& elems, mrdb::PreparedStatement* ps_elems) {
00024     ps_elems->setInt(1,id);
00025     mrdb::ResultSet* rs = ps_elems->executeQuery();
00026
00027     try {
00028       while( rs->next() ) {
00029         ItemType elem;
00030         elem.setSourceRowElement(*SourceRowElement::deserializeElementFromResultSet(rs,2));
00031
00032         elems.push_back(elem);
00033       }
00034     } catch (...) {
00035       delete rs;
00036       throw;
```

```
00037     }
00038
00039     delete rs;
00040   }
00041
00042
00043   void
00044   QueryResult::Iterator::init( const std::string& rulesTable,
00045   double support,
00046   double confidence ) {
00047     mrdb::Connection* mrdb_conn = MineruleOptions::getSharedOptions().getMRDB().getMRDBConnection();
00048     Connection connection;
00049     connection.setOutTableName(rulesTable);
00050
00051     state=mrdb_conn->createStatement();
00052     std::string query =
00053     "SELECT bodyId, headId, supp, con "
00054     "FROM " + connection.getTableName(Connection::RulesTable) + " "
00055     "WHERE supp>=" + Converter(support).toString() + " AND "
00056     "con>="+Converter(confidence).toString();
00057
00058     body_elems = mrdb_conn->prepareStatement(
00059     "SELECT * FROM "+ connection.getTableName(Connection::BodiesTable) +
00060     " WHERE id=?");
00061
00062     head_elems = mrdb_conn->prepareStatement(
00063     "SELECT * FROM "+ connection.getTableName(Connection::HeadsTable) +
00064     " WHERE id=?");
00065
00066
00067     MRDebug() « "QueryResult::Iterator, the filter query is:" « query « std::endl;
00068
00069     rs_rules = state->executeQuery(query);
00070   }
00071
00072   bool QueryResult::Iterator::next() {
00073     return rs_rules->next();
00074   }
00075
00076   void QueryResult::Iterator::getRule( Rule& r ) {
00077     ItemSet* body = new ItemSet();
00078     ItemSet* head = new ItemSet();
00079     size_t bid, hid;
00080
00081     bid = rs_rules->getInt(1);
00082     hid = rs_rules->getInt(2);
00083
00084     readElems( bid, *body, body_elems );
00085     readElems( hid, *head, head_elems );
00086
00087     r.setBody(body);
00088     r.setHead(head);
00089     r.setSupport(rs_rules->getDouble(3));
00090     r.setConfidence(rs_rules->getDouble(4));
00091     r.setBodyId(bid);
00092     r.setHeadId(hid);
00093   }
00094
00095 } // namespace
```

7.355 /Users/esposito/Software/minerule/src/Result/RuleFormatter.cpp File Reference

```
#include "minerule/Result/RuleFormatter.hpp"
#include <iomanip>
#include <sstream>
```

Namespaces

- namespace minerule

7.356 RuleFormatter.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Result/RuleFormatter.hpp"
00017 #include <iomanip>
00018 #include <sstream>
00019
00020
00021 namespace minerule {
00022         std::string RuleFormatter::quote(const std::string& elem) {
00023                 return "["+elem+"]";
00024         }
00025
00026         std::string RuleFormatter::quoteElems(const ItemSet& elems) {
00027                 std::string result;
00028
00029                 ItemSet::const_iterator it=elems.begin();
00030
00031                 if(it!=elems.end()) {
00032                         result = quote(it->asString());
00033                         it++;
00034                 }
00035
00036                 for(; it!=elems.end(); it++) {
00037                         result +=  ", " + quote(it->asString());
00038                 }
00039
00040                 return result;
00041         }
00042
00043
00044         std::string
00045         SimpleRuleFormatter::formatRule(const Rule& rule, bool includeSuppConf) {
00046                 std::stringstream out;
00047
00048                 out     << std::setw(_fieldWidths.body) << quoteElems(rule.getBody())
00049                         << _bhSep
00050                         << std::left << std::setw(_fieldWidths.head) << quoteElems(rule.getHead());
00051
00052                 if(includeSuppConf) {
00053                         out << std::right
00054                                 << " "
00055                                 << std::setw(_fieldWidths.supp) << rule.getSupport()
00056                                 << " "
00057                                 << std::setw(_fieldWidths.conf) << rule.getConfidence();
00058                 }
00059
00060                 return out.str();
00061         }
00062
00063
00064         void
00065         SimpleRuleFormatter::printRule(const Rule& rule) {
00066                 if( suppressLog() ) {
00067                         std::cout << formatRule(rule)  << std::endl ;
00068                 } else {
00069                         MRLog() << formatRule(rule)  << std::endl;
00070                 }
00071         }
00072
00073
00074 } // namespace
```

7.357 /Users/esposito/Software/minerule/src/Result/RulesMatcher.cpp File Reference

```
#include "minerule/Result/RulesMatcher.hpp"
```

Namespaces

- namespace minerule

7.358 RulesMatcher.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Result/RulesMatcher.hpp"
00017
00018 namespace minerule {
00019         bool RulesMatcher::match( const Rule& r, const RuleTransaction<RuleSetType>& t ) {
00020                 std::set< ItemType > bodySet;
00021                 std::set< ItemType > headSet;
00022
00023                 for( RuleSetType::const_iterator it = t.begin(); it!=t.end(); ++it ) {
00024                         bodySet.insert(it->first);
00025                         headSet.insert(it->second);
00026                 }
00027
00028                 const ItemSet& body = r.getBody();
00029                 for( ItemSet::const_iterator bodyIt=body.begin(); bodyIt!=body.end();++bodyIt) {
00030                         if(bodySet.find(*bodyIt) == bodySet.end())
00031                                 return false;
00032                 }
00033
00034                 const ItemSet& head = r.getHead();
00035                 for( ItemSet::const_iterator headIt=head.begin(); headIt!=head.end();++headIt) {
00036                         if(headSet.find(*headIt) == headSet.end())
00037                                 return false;
00038                 }
00039
00040                 return true;
00041         }
00042
00043         bool RulesMatcher::match( const Rule& r, const ItemTransaction<ItemSetType>& bodySet, const
      ItemTransaction<ItemSetType>& headSet ) {
00044                 const ItemSet& body = r.getBody();
00045                 for( ItemSet::const_iterator it=body.begin(); it!=body.end();++it) {
00046                         if(bodySet.find(*it) == bodySet.end())
00047                                 return false;
00048                 }
00049
00050                 const ItemSet& head = r.getHead();
00051                 for( ItemSet::const_iterator it=head.begin(); it!=head.end();++it) {
00052                         if(headSet.find(*it) == headSet.end())
00053                                 return false;
00054                 }
00055
00056                 return true;
00057         }
00058 }
```

7.359 /Users/esposito/Software/minerule/src/Utils/AlgorithmTypes.cpp File Reference

```
#include "minerule/Utils/AlgorithmTypes.hpp"
```

Namespaces

- namespace minerule

Functions

- const std::string & minerule::miningTaskToString (MiningTasks mt)
- MiningTasks minerule::stringToMiningTask (const std::string &s)
- std::string minerule::buildStringWithListOfMiningTasks ()
- const std::string & minerule::stringWithListOfMiningTasks ()
- const std::string & minerule::algorithmTypeToString (AlgorithmTypes t)
- AlgorithmTypes minerule::stringToAlgorithmType (const std::string &s)
- std::string minerule::buildStringWithListOfAlgorithmTypes ()
- std::string minerule::stringWithListOfAlgorithmTypes ()

7.360 AlgorithmTypes.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/AlgorithmTypes.hpp"
00017
00018
00019 namespace minerule {
00020   static const std::string taskNames[] = {
00021                                           "MineRules",
00022                                           "MineItemsets",
00023                                           "MineSequences" };
00024
00025
00026
00027   static const std::string algoNames[] = {
00028                                          "None",
00029                                          "BFSWithGidsNoCross",
00030                                          "BFSWithGidsAndCross",
00031                                          "Care",
00032                                          "ConstrainedItemsets"};
00033
00034
00035   const std::string& miningTaskToString(MiningTasks mt) {
00036     return taskNames[mt];
00037   }
00038
00039
00040   MiningTasks stringToMiningTask(const std::string& s) {
00041     MiningTasks task;
```

```
00042     for(task=MTMineRules; task<MTEnd; task=static_cast<MiningTasks>(task+1) ) {
00043       if( taskNames[task]==s )
00044                   return task;
00045     }
00046 
00047 
00048     throw MineruleException( MR_ERROR_INTERNAL, "Cannot convert "+s+" to a valid MiningTask" );
00049   }
00050 
00051   std::string buildStringWithListOfMiningTasks() {
00052     std::string l;
00053     MiningTasks task;
00054 
00055     for(task=MTMineRules; task<MTEnd; task=static_cast<MiningTasks>(task+1)) {
00056       if(task!=MTMineRules)
00057                   l+=",";
00058       l+=taskNames[task];
00059     }
00060 
00061     return l;
00062   }
00063 
00064   const std::string& stringWithListOfMiningTasks() {
00065     static std::string l = buildStringWithListOfMiningTasks();
00066 
00067     return l;
00068   }
00069 
00070 
00071 
00072 
00073   const std::string& algorithmTypeToString(AlgorithmTypes t) {
00074     return algoNames[t];
00075   }
00076 
00077   AlgorithmTypes stringToAlgorithmType(const std::string& s) {
00078     AlgorithmTypes algoType;
00079     for( algoType=ATNone; algoType<ATEnd; algoType=static_cast<AlgorithmTypes>(algoType+1) ) {
00080       if(algoNames[algoType]==s)
00081                   return algoType;
00082         }
00083 
00084     throw MineruleException( MR_ERROR_INTERNAL, "Cannot convert "+s+" to a valid AlgorithmType" );
00085   }
00086 
00087   std::string buildStringWithListOfAlgorithmTypes() {
00088     std::string l;
00089     AlgorithmTypes algoType;
00090 
00091     for(algoType=ATNone; algoType<ATEnd; algoType=static_cast<AlgorithmTypes>(algoType+1)) {
00092       if(algoType!=ATNone)
00093                   l+=",";
00094 
00095       l+=algoNames[algoType];
00096     }
00097 
00098     return l;
00099   }
00100 
00101   std::string stringWithListOfAlgorithmTypes() {
00102     return buildStringWithListOfAlgorithmTypes();
00103   }
00104 };
```

7.361 /Users/esposito/Software/minerule/src/Utils/Bitstring.cpp File Reference

```
#include "minerule/Utils/Bitstring.hpp"
```

Namespaces

- namespace minerule

Functions

- std::ostream & minerule::operator<< (std::ostream &out, const BitString &bs)
- std::istream & minerule::operator>> (std::istream &in, BitString &bs)

7.362 Bitstring.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/Bitstring.hpp"
00017
00018 namespace minerule {
00019
00020
00021
00022 static int NBITS[] = {
00023 0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1,
00024 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 1, 2,
00025 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3,
00026 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 1, 2, 2, 3,
00027 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3,
00028 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 2, 3, 3, 4, 3, 4,
00029 4, 5, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5,
00030 6, 4, 5, 5, 6, 5, 6, 6, 7, 1, 2, 2, 3, 2, 3, 3, 4,
00031 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4, 5, 3,
00032 4, 4, 5, 4, 5, 5, 6, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4,
00033 4, 5, 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5,
00034 6, 5, 6, 6, 7, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5,
00035 4, 5, 5, 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5,
00036 6, 6, 7, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6,
00037 6, 7, 4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8
00038 };
00039
00040 /*************************************************************
00041  Body module : bitstring.cpp
00042  Code generated by Object Domain script
00043      Copyright 95
00044 *************************************************************/
00045
00046
00047 int BitString::intersections = 0;
00048 /*************************************************************************
00049    BitString::init
00050
00051 *************************************************************************/
00052 void BitString::init(int nbits)
00053 {
00054 n = nbits;
00055 Nw = (n == 0) ? 0 : (n-1) / Nb;
00056 for (int i=0; i<=Nw; i++) bits.insert(bits.end(),(Bits)0);
00057 }
00058
00059 /*************************************************************************
00060    BitString::BitString
00061
00062 *************************************************************************/
00063  BitString::BitString() { init(0); }
00064
00065  BitString::BitString(int nbits)
00066 {
00067 init(nbits);
00068 }
00069
00070 BitString::BitString(const BitString& bs) {
00071 n = bs.n;
00072 Nw = bs.Nw;
```

```
00073 for (int i=0; i<=Nw; i++) bits.insert(bits.end(),bs.bits[i]);
00074 }
00075
00076 /*************************************************************************
00077   BitString BitString::set
00078
00079 *************************************************************************/
00080 BitString&  BitString::set()
00081 {
00082 for (int i=0; i<=Nw; i++) bits[i] = ~(Bits)0;
00083 return *this;
00084 }
00085
00086 /*************************************************************************
00087   BitString BitString::set
00088
00089 *************************************************************************/
00090 BitString&  BitString::set(int i, boolean value)
00091 {
00092 if (n <= i) _Xran(i);
00093 if (value) bits[i/Nb] |= (Bits)1 « i%Nb;
00094 else bits[i/Nb] &= ~((Bits)1 « i%Nb);
00095 return *this;
00096 }
00097
00098 /*************************************************************************
00099   BitString BitString::reset
00100
00101 *************************************************************************/
00102 BitString&  BitString::reset()
00103 {
00104 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00105 return *this;
00106 }
00107
00108 /*************************************************************************
00109   BitString BitString::reset
00110
00111 *************************************************************************/
00112 BitString&  BitString::reset(int i)
00113 {
00114 return set(i, false);
00115 }
00116
00117 /*************************************************************************
00118   BitString BitString::clear
00119
00120 *************************************************************************/
00121 BitString&  BitString::clear()
00122 {
00123 for (int i=0; i<=Nw; i++) bits[i] = (Bits)0;
00124 return *this;
00125 }
00126
00127 /*************************************************************************
00128   BitString BitString::clear
00129
00130 *************************************************************************/
00131 BitString&  BitString::clear(int i)
00132 {
00133 return set(i, false);
00134 }
00135
00136 /*************************************************************************
00137   BitString BitString::invert
00138
00139 *************************************************************************/
00140 BitString&  BitString::invert()
00141 {
00142 for (int i=0; i<=Nw; i++) bits[i] = ~bits[i];
00143 return *this;
00144 }
00145
00146 /*************************************************************************
00147   BitString BitString::invert
00148
00149 *************************************************************************/
00150 BitString&  BitString::invert(int i)
00151 {
00152 if (n <= i) _Xran(i);
00153 bits[i/Nb] ^= (Bits)1 « i%Nb;
00154 return *this;
00155 }
00156
00157 /*************************************************************************
00158   BitString BitString::operator|=
00159
```

```
00160 ***************************************************************************/
00161 BitString&  BitString::operator|=(const BitString& bs)
00162 {
00163 if (n < bs.n) {
00164         for (; n < (Nw+1)*Nb && n < bs.n; n++) reset(n);
00165         n = bs.n;
00166 }
00167 if (Nw < bs.Nw) {
00168         for (; Nw <= bs.Nw; ) { Nw++; bits.insert(bits.end(), (Bits)0); }
00169 }
00170 for (int i=0; i<=bs.Nw; i++) bits[i] |= bs.bits[i];
00171 return *this;
00172 }
00173
00174 /***************************************************************************
00175   BitString BitString::operator^=
00176
00177 ***************************************************************************/
00178 BitString&  BitString::operator^=(const BitString& bs)
00179 {
00180 if (n < bs.n) {
00181         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n,bs.test(n));
00182         n = bs.n;
00183 }
00184 if (Nw < bs.Nw) {
00185         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), bs.bits[Nw]); }
00186 }
00187 for (int i=0; i<=bs.Nw; i++) bits[i] ^= bs.bits[i];
00188 return *this;
00189 }
00190
00191 /***************************************************************************
00192   BitString BitString::operator&=
00193
00194 ***************************************************************************/
00195 BitString&  BitString::operator&=(const BitString& bs)
00196 {
00197 /*
00198 if (n < bs.n) {
00199         for (; n < (Nw+1)*Nb && n < bs.n; n++) set(n);
00200         n = bs.n;
00201 }
00202 if (Nw < bs.Nw) {
00203         for (; Nw < bs.Nw; ) { Nw++; bits.insert(bits.end(), ~(char)0); }
00204         n = bs.n;
00205 }
00206 for (int i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00207 */
00208 if (Nw < bs.Nw) {
00209         for (int i=0; i<=Nw; i++) bits[i] &= bs.bits[i];
00210 } else {
00211         int i;
00212         for (i=0; i<=bs.Nw; i++) bits[i] &= bs.bits[i];
00213         for (; i<=Nw; i++) bits[i] = (Bits)0;
00214 }
00215 intersections++;
00216 return *this;
00217 }
00218
00219 /***************************************************************************
00220   BitString BitString::operator&
00221
00222 ***************************************************************************/
00223 BitString  BitString::operator&(const BitString& bs1)
00224 {
00225         BitString bs;
00226         int min = Nw < bs1.Nw ? Nw : bs1.Nw;
00227         for (int i=0; i<=min; i++) bs.bits.insert(bs.bits.end(),bits[i] & bs1.bits[i]);
00228         bs.Nw = min;
00229         bs.n = n < bs1.n ? n : bs1.n;
00230         return bs;
00231 }
00232
00233 /***************************************************************************
00234   BitString BitString::operator=
00235
00236 ***************************************************************************/
00237 BitString&  BitString::operator=(const BitString& bs)
00238 {
00239 bits.erase(bits.begin(),bits.end());
00240 n = bs.n;
00241 Nw = bs.Nw;
00242 for (int i=0; i<=Nw; i++) bits.insert(bits.end(),bs.bits[i]);
00243 //for (int i=0; i<=Nw && i<=bs.Nw; i++) bits[i] = bs.bits[i];
00244 return *this;
00245 }
00246
```

```
00247 /*****************************************************************************
00248   BitString BitString::operator==
00249
00250 *****************************************************************************/
00251 boolean BitString::operator==(const BitString& bs)
00252 {
00253 boolean eq = n == bs.n;
00254 for (int i=0; i<Nw && eq; i++) eq = bits[i] == bs.bits[i];
00255 for (int i=(Nw-1)*Nb; i<n && eq; i++) eq = test(i) == bs.test(i);
00256 return eq;
00257 }
00258
00259 /*****************************************************************************
00260   BitString BitString::operator!=
00261
00262 *****************************************************************************/
00263 boolean BitString::operator!=(const BitString& bs)
00264 {
00265 return !operator==(bs);
00266 }
00267
00268 /*****************************************************************************
00269   void BitString::_Xran
00270
00271 *****************************************************************************/
00272 void BitString::_Xran(int i)
00273 {
00274 //std::cerr « "BitString index out of range: " « i « std::endl;
00275 int newNw = i/Nb;
00276 for (int j=Nw+1; j<=newNw; j++) bits.insert(bits.end(),(Bits)0);
00277 n = i+1; Nw = newNw;
00278 }
00279
00280 /*****************************************************************************
00281   int BitString::count
00282
00283 *****************************************************************************/
00284 int BitString::count(boolean what) const
00285 {
00286 int i,j, k=0;
00287 for (i=0; i<=Nw; i++) {
00288 //      Bits b = bits[i];
00289 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b»=1;}
00290         for (j=0; j<sizeof(Bits); j++)
00291         k += NBITS[*(((unsigned char*)&bits[i])+j)];
00292 }
00293 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00294 return k;
00295 }
00296
00297 /*****************************************************************************
00298   bool BitString::moreThan
00299
00300 *****************************************************************************/
00301 bool BitString::moreThan(double threshold) const
00302 {
00303 int i,j, k=(int)(threshold);
00304 for (i=0; i<=Nw && k >= 0; i++) {
00305 //      Bits b = bits[i];
00306 //      for (int j=0; j<Nb && b!=0; j++) { k+=b&0x01; b»=1;}
00307         for (j=0; j<sizeof(Bits); j++)
00308         k -= NBITS[*(((unsigned char*)&bits[i])+j)];
00309 }
00310 //for (i=0, k=0; i<n; i++) if (test(i) == what) k++;
00311 return k<0;
00312 }
00313
00314 /*****************************************************************************
00315   std::ostream BitString::operator«
00316
00317 *****************************************************************************/
00318 std::ostream& operator«(std::ostream& out, const BitString& bs)
00319 {
00320 for (int i=0; i<bs.n; i++) out « (bs.test(i) ? '1' : '0');
00321 return out;
00322 }
00323
00324 /*****************************************************************************
00325   istream BitString::operator»
00326
00327 *****************************************************************************/
00328 std::istream& operator»(std::istream& in, BitString& bs)
00329 {
00330 std::ios::iostate St = std::ios::goodbit;
00331 boolean Chg = false;
00332 int C = in.rdbuf()->sgetc();
00333 bs.reset();
```

```
00334 while (C != EOF && C == ' ') C = in.rdbuf()->snextc();
00335 for (size_t M = 0; true ; C = in.rdbuf()->snextc(), ++M)
00336 {
00337         if (C == EOF) {St |= std::ios::eofbit; break; }
00338         else if (C != '0' && C != '1') break;
00339         //else if (_X.allocation() <= _X.length())
00340         //      {_St |= ios::failbit;
00341         //      break; }
00342         else { bs.set(M,C=='1'); Chg = true; }
00343 }
00344 //if (!_Chg)  _St |= ios::failbit;
00345 in.setstate(St);
00346 return (in);
00347 }
00348
00349
00350 //MARCO
00351
00352 void BitString::serialize(char* serialized,int* start)
00353 {int i;
00354
00355  memcpy(&(serialized[(*start)]),&Nw,sizeof(int));
00356         //cout<<int(serialized[(*start)])<<" "<<std::endl;
00357  //cout<<"\"";
00358  (*start) += sizeof(int);
00359  memcpy(&(serialized[(*start)]),&n,sizeof(int));
00360  (*start) += sizeof(int);
00361         for (i=0;i<=Nw;i++)
00362         {
00363          memcpy(&(serialized[(*start)]),&(bits[i]),sizeof(Bits));
00364          //cout<<(int)(serialized[(*start)])<<" ";
00365                 (*start) += sizeof(Bits);
00366         }
00367 }
00368
00369 void BitString::unserialize(char* serialized,int* start)
00370 {int i;
00371  char tmpchar;
00372
00373  memcpy(&Nw,&(serialized[(*start)]),sizeof(int));
00374         //cout<<int(serialized[(*start)])<<" "<<std::endl;
00375  //cout<<"\"";
00376  (*start) += sizeof(int);
00377  memcpy(&n,&(serialized[(*start)]),sizeof(int));
00378  (*start) += sizeof(int);
00379  for (i=bits.size();i<=Nw;i++) bits.insert(bits.end(),(Bits)0);
00380         for (i=0;i<=Nw;i++)
00381         {
00382          memcpy(&tmpchar,&serialized[(*start)],sizeof(Bits));
00383          //cout<<(int)(serialized[(*start)])<<" ";
00384          bits[i] = tmpchar;
00385          (*start) += sizeof(Bits);
00386         }
00387 // setNBit(numbit);
00388 }
00389
00390 void BitString::print(std::ostream& out)
00391 {int i,dim;
00392
00393  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00394  {
00395  dim=(*this).length();
00396  for (i=0;i<dim;i++)
00397  {
00398   if ((i%8==0)&&(i!=0)) out<<"-";
00399   out<<test(i);
00400  }
00401  }
00402 }
00403
00404 void BitString::print()
00405 {int i,dim;
00406
00407  if (DB_LEVEL_ALL<=DEBUG_LEVEL)
00408  {
00409  dim=(*this).length();
00410  for (i=0;i<dim;i++)
00411  {
00412   if ((i%8==0)&&(i!=0)) std::cout<<"-";
00413   std::cout<<test(i);
00414  }
00415  }
00416 }
00417
00418 } // end namespace
```

7.363 /Users/esposito/Software/minerule/src/Utils/Constraints.cpp File Reference

```
#include "minerule/Algorithms/Bodymap.hpp"
#include "minerule/Utils/Constraints.hpp"
```

Namespaces

- namespace minerule

7.364 Constraints.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Algorithms/Bodymap.hpp"
00017 #include "minerule/Utils/Constraints.hpp"
00018
00019 namespace minerule {
00020
00021
00022
00023 bool SumLessThan::check (BodyMap& itemMap, std::vector<ItemType>& items) {
00024         int sum = 0;
00025         for(std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum < value; i++)
00026                 sum += itemMap[*i].attribute[attributeIndex].minValue();
00027         return sum < value;
00028 }
00029
00030 bool SumLessThan::check (Transaction& items) {
00031         int sum = 0; int j = 0;
00032         for (Transaction::iterator i = items.begin(); i != items.end() && sum < value; i++)
00033                 sum += items.values[j++];
00034                 //sum += items.values[attributeIndex][j++];
00035         return sum < value;
00036 }
00037
00038 bool SumLessEq::check (BodyMap& itemMap, std::vector<ItemType>& items) {
00039         int sum = 0;
00040         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00041                 sum += itemMap[*i].attribute[attributeIndex].minValue();
00042         return sum <= value;
00043 }
00044
00045 bool SumLessEq::check (Transaction& items) {
00046         int sum = 0; int j = 0;
00047         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00048                 sum += items.values[j++];
00049                 //sum += items.values[attributeIndex][j++];
00050         return sum <= value;
00051 }
00052
00053 bool SumGreaterThan::check (BodyMap& itemMap, std::vector<ItemType>& items) {
00054         int sum = 0;
00055         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00056                 sum += itemMap[*i].attribute[attributeIndex].maxValue();
00057         return sum > value;
00058 }
00059
```

```
00060 bool SumGreaterThan::check (Transaction& items) {
00061         int sum = 0; int j = 0;
00062         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00063                 sum += items.values[j++];
00064                 //sum += items.values[attributeIndex][j++];
00065         return sum > value;
00066 }
00067
00068 bool SumGreaterEq::check (BodyMap& itemMap, std::vector<ItemType>& items) {
00069         int sum = 0;
00070         for (std::vector<ItemType>::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00071                 sum += itemMap[*i].attribute[attributeIndex].maxValue();
00072         return sum >= value;
00073 }
00074
00075 bool SumGreaterEq::check (Transaction& items) {
00076         int sum = 0; int j = 0;
00077         for (Transaction::iterator i = items.begin(); i != items.end() && sum <= value; i++)
00078                 sum += items.values[j++];
00079                 //sum += items.values[attributeIndex][j++];
00080         return sum >= value;
00081 }
00082
00083 } //end namespace
```

7.365 /Users/esposito/Software/minerule/src/Utils/FileUtils.cpp File Reference

```
#include "minerule/Utils/FileUtils.hpp"
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
```

Namespaces

- namespace minerule

7.366 FileUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/FileUtils.hpp"
00017 #include <sys/types.h>
00018 #include <sys/stat.h>
00019 #include <unistd.h>
00020
00021 namespace minerule {
00022         bool FileUtils::fileExists(const std::string& fname) {
00023                 struct stat trash;
00024                 if( stat(fname.c_str(),&trash)==0 )
00025                         return true;
00026                 else
00027                         return false;
00028         }
00029 }
```

7.367 /Users/esposito/Software/minerule/src/Utils/MineruleErrors.cpp File Reference

```
#include "minerule/Utils/MineruleErrors.hpp"
#include <assert.h>
```

Namespaces

- namespace minerule

Functions

- const char * minerule::me_error_name (MineruleErrors me)
- int minerule::me_error_begin ()
- int minerule::me_error_end ()

Variables

- const char * minerule::MineruleErrorsNames [ ]

7.368 MineruleErrors.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/MineruleErrors.hpp"
00017 #include <assert.h>
00018
00019 namespace minerule {
00020
00021 const char* MineruleErrorsNames[] = {
00022   "MR_ERROR_NO_ERROR",
00023   "MR_ERROR_UNKNOWN",
00024   "MR_ERROR_INTERNAL",
00025   "MR_ERROR_INPUT_FILE_NOT_FOUND",
00026   "MR_ERROR_OUTPUT_FILE_PROBLEM",
00027   "MR_ERROR_NO_MINERULE_SPECIFIED",
00028   "MR_ERROR_NO_OPTIONFILE_SPECIFIED",
00029   "MR_ERROR_OPTION_CONFIGURATION",
00030   "MR_ERROR_MINERULE_ALREADY_EXISTS",
00031   "MR_ERROR_DATABASE_ERROR",
00032   "MR_ERROR_OPTION_PARSING",
00033   "MR_ERROR_MINERULETEXT_PARSING",
00034   "MR_ERROR_CATALOGUE_ERROR",
00035   "MR_ERROR_OPTIMIZER_ERROR",
00036   "MR_ERROR_INSTALLATION_PROBLEM",
00037   "MR_ERROR_SAFETY_PROBLEM"
00038 };
00039
```

```
00040 const char* me_error_name(MineruleErrors me) {
00041   assert( me>=me_error_begin() && me<me_error_end() );
00042   return MineruleErrorsNames[me];
00043 }
00044
00045 int me_error_begin() {
00046   return MR_ERROR_NO_ERROR;
00047 }
00048
00049 int me_error_end() {
00050   return (int)MR_ERROR_SAFETY_PROBLEM + 1;
00051 }
00052
00053
00054 } // namespace
```

7.369 /Users/esposito/Software/minerule/src/Utils/MineruleException.cpp File Reference

```
#include "minerule/Utils/MineruleException.hpp"
#include "minerule/Utils/StringUtils.hpp"
#include <iostream>
```

Namespaces

- namespace minerule

7.370 MineruleException.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/MineruleException.hpp"
00017 #include "minerule/Utils/StringUtils.hpp"
00018
00019 #include <iostream>
00020
00021 namespace minerule {
00022
00023         #undef MineruleException
00024
00025         MineruleException::MineruleException(std::string sourceFile, int sourceLine,  size_t errCode,
      std::string msg) : message(msg), file(sourceFile), line(sourceLine), errorCode(errCode) {
00026                                 formatMessage();
00027         }
00028
00029         void MineruleException::formatMessage() _NOEXCEPT {
00030                 std::stringstream ss;
00031
00032                 ss « "\n\t" « StringUtils::toBold("class:") « "\tMineruleException " « std::endl
00033                         « StringUtils::toBold("\tsource:\t") « file « ":" « line « std::endl
00034                         « StringUtils::toBold("\tcode:\t")« errorCode
00035                         « " - " « me_error_name((MineruleErrors) errorCode) « std::endl
00036                         « StringUtils::toBold("\tmessage:") « StringUtils::toBoldRed("[");
```

```
00037
00038                 // std::vector<std::string>* chunks = StringUtils::splitToLength(message, 70);
00039                 // ss « StringUtils::join(*chunks, "\n");
00040                 ss « message;
00041                 ss « StringUtils::toBoldRed("]");
00042
00043                 // delete chunks;
00044
00045                 formattedMessage = ss.str();
00046         }
00047
00048         const char* MineruleException::what() const _NOEXCEPT {
00049                 return formattedMessage.c_str();
00050         }
00051
00052         const char* MineruleException::unformattedMessage() const _NOEXCEPT {
00053       return message.c_str();
00054         }
00055
00056 }
```

7.371 /Users/esposito/Software/minerule/src/Utils/MineruleLogs.cpp File Reference

```
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include "minerule/Utils/StringUtils.hpp"
```

Namespaces

- namespace minerule

Functions

- void minerule::logRespectingMaxLogLength (std::ostream &(*logger)(void), size_t indentLen, const std↩ ::string &msg)
- std::ostream & minerule::MRLog ()
- void minerule::MRLog (const std::string &msg)
- std::ostream & minerule::MRErr ()
- void minerule::MRErr (const std::string &msg)
- std::ostream & minerule::MRWarn ()
- void minerule::MRWarn (const std::string &msg)
- std::ostream & minerule::MRDebug ()
- void minerule::MRDebug (const std::string &msg)
- void minerule::MRLogPush (const std::string &descr)
- void minerule::MRLogPop ()
- void minerule::MRErrPush (const std::string &descr)
- void minerule::MRErrPop ()
- void minerule::MRWarnPush (const std::string &descr)
- void minerule::MRWarnPop ()
- void minerule::MRDebugPush (const std::string &descr)
- void minerule::MRDebugPop ()
- void minerule::MRLogStartMeasuring (const std::string &description)
- void minerule::MRLogStopMeasuring (const std::string &description)
- void minerule::MRLogShowMeasurements ()

Variables

- const size_t MAX_LOG_LENGTH = 70

7.371.1 Variable Documentation

7.371.1.1 MAX_LOG_LENGTH

```
const size_t MAX_LOG_LENGTH = 70
```

Definition at line 20 of file MineruleLogs.cpp.

7.372 MineruleLogs.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/MineruleLogs.hpp"
00017 #include "minerule/Utils/MineruleOptions.hpp"
00018 #include "minerule/Utils/StringUtils.hpp"
00019
00020 const size_t MAX_LOG_LENGTH = 70;
00021
00022 namespace minerule {
00023
00024         void logRespectingMaxLogLength(std::ostream& (*logger)(void), size_t indentLen, const
       std::string& msg ) {
00025
00026                 std::vector<std::string>* chunks = StringUtils::splitToLength(msg,
       MAX_LOG_LENGTH-indentLen);
00027
00028                 std::vector<std::string>::const_iterator it= chunks->begin();
00029                 if( it != chunks->end() ) {
00030                         logger() << *it << std::endl;
00031                 }
00032
00033                 ++it;
00034
00035                 for( ; it!=chunks->end(); ++it ) {
00036                         logger() << StringUtils::toGreen("  +") << *it << std::endl;

00037                 }
00038
00039                 delete chunks;
00040         }
00041
00042
00043     std::ostream& MRLog() {
00044       return MineruleOptions::getSharedOptions().getLogStream();
00045     }
00046
00047         void MRLog(const std::string& msg) {
00048                 size_t indentLength =
       MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().getIndentLen();
00049                 logRespectingMaxLogLength(MRLog, indentLength ,msg);
```

```
00050         }
00051
00052     std::ostream& MRErr() {
00053       return MineruleOptions::getSharedOptions().getErrStream();
00054     }
00055
00056         void MRErr(const std::string& msg) {
00057                 size_t indentLength =
      MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().getIndentLen();
00058                 logRespectingMaxLogLength(MRErr, indentLength, msg);
00059         }
00060
00061
00062     std::ostream& MRWarn() {
00063       return MineruleOptions::getSharedOptions().getWarnStream();
00064     }
00065
00066         void MRWarn(const std::string& msg) {
00067             size_t indentLength =
      MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().getIndentLen();
00068                 logRespectingMaxLogLength(MRWarn, indentLength, msg);
00069         }
00070
00071
00072     std::ostream& MRDebug() {
00073       return MineruleOptions::getSharedOptions().getDebugStream();
00074     }
00075
00076         void MRDebug(const std::string& msg) {
00077         size_t indentLength =
      MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().getIndentLen();
00078                 logRespectingMaxLogLength(MRDebug, indentLength, msg);
00079         }
00080
00081
00082
00083     void MRLogPush(const std::string& descr) {
00084       MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().push(descr);
00085     }
00086
00087     void MRLogPop() {
00088       MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().pop();
00089     }
00090
00091     void MRErrPush(const std::string& descr) {
00092       MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().push(descr);
00093     }
00094
00095     void MRErrPop() {
00096       MineruleOptions::getSharedOptions().getErrStreamObj().getLogger().pop();
00097     }
00098
00099     void MRWarnPush(const std::string& descr) {
00100       MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().push(descr);
00101     }
00102
00103     void MRWarnPop() {
00104       MineruleOptions::getSharedOptions().getWarnStreamObj().getLogger().pop();
00105     }
00106
00107     void MRDebugPush(const std::string& descr) {
00108       MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().push(descr);
00109     }
00110
00111     void MRDebugPop() {
00112       MineruleOptions::getSharedOptions().getDebugStreamObj().getLogger().pop();
00113     }
00114
00115
00116     void MRLogStartMeasuring(const std::string& description) {
00117         MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().startMeasuring(description);
00118     }
00119     void MRLogStopMeasuring(const std::string& description) {
00120         MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().stopMeasuring(description);

00121     }
00122
00123         void MRLogShowMeasurements() {
00124                 MineruleOptions::getSharedOptions().getLogStreamObj().getLogger().logMeasurements();
00125         }
00126 }
```

7.373 /Users/esposito/Software/minerule/src/Utils/MineruleOptions.cpp File Reference

```
#include "minerule/Utils/MineruleOptions.hpp"
#include <iostream>
#include <fstream>
#include "minerule/Utils/OptionParserLib.hpp"
```

Namespaces

- namespace minerule

Variables

- const std::string MINERULE_OPTIONS_PARSING_ERROR

7.373.1 Variable Documentation

7.373.1.1 MINERULE_OPTIONS_PARSING_ERROR

```
const std::string MINERULE_OPTIONS_PARSING_ERROR
```

Initial value:

```
=
    "Parsing Error while parsing MineruleOptions"
```

Definition at line 22 of file MineruleOptions.cpp.

7.374 MineruleOptions.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/MineruleOptions.hpp"
00017 #include <iostream>
00018 #include <fstream>
00019
00020 //#include <malloc.h>
00021
00022 const std::string MINERULE_OPTIONS_PARSING_ERROR =
```

```
00023     "Parsing Error while parsing MineruleOptions";
00024
00025 #include "minerule/Utils/OptionParserLib.hpp"
00026
00027 namespace minerule {
00028 const std::string MineruleOptions::DEFAULT_FILE_NAME("options.txt");
00029
00030 MineruleOptions MineruleOptions::sharedOptions;
00031
00032 MineruleOptions::~MineruleOptions() {
00033   std::map<std::string, MRLogger *>::iterator it;
00034   for (it = knownStreams.begin(); it != knownStreams.end(); it++) {
00035     std::ostream *ostr = NULL;
00036
00037     if (it->first != "<stdout>" && it->first != "<stderr>") {
00038       ostr = it->second->getStream();
00039     }
00040     delete it->second;
00041     if (ostr != NULL)
00042       delete ostr;
00043   }
00044 }
00045
00046 OptionBase &MineruleOptions::subclassForName(const std::string &oclass) {
00047   if (oclass == "mrdb")
00048     return getMRDB();
00049   else if (oclass == "safety")
00050     return getSafety();
00051   else if (oclass == "miningalgorithms")
00052     return getMiningAlgorithms();
00053   else if (oclass == "optimizations")
00054     return getOptimizations();
00055   else if (oclass == "logstream")
00056     return getLogStreamObj();
00057   else if (oclass == "errstream")
00058     return getErrStreamObj();
00059   else if (oclass == "warnstream")
00060     return getWarnStreamObj();
00061   else if (oclass == "debugstream")
00062     return getDebugStreamObj();
00063   else if (oclass == "parsers")
00064     return getParsers();
00065   else
00066     return OptionBase::subclassForName(oclass);
00067 }
00068
00069 void MineruleOptions::init() {
00070   miningAlgorithms.getRulesMiningAlgorithms()
00071       .getPartitionBase()
00072       .setRowsPerPartition(300000); /*300.000*/
00073   miningAlgorithms.getRulesMiningAlgorithms()
00074       .getPartitionWithClusters()
00075       .setRowsPerPartition(300000); /*300.000*/
00076   miningAlgorithms.getRulesMiningAlgorithms().getFPGrowth().setAlgoType(
00077       MineruleOptions::MiningAlgorithms::RulesMiningAlgorithms::FPGrowth::
00078           Original);
00079
00080   optimizations.setTryOptimizations(false);
00081
00082   MRLogger *stdlog = new MRLogger(std::cout);
00083   MRLogger *errlog = new MRLogger(std::cerr);
00084
00085   logStream.setLogger(*stdlog);
00086   errStream.setLogger(*errlog);
00087   warnStream.setLogger(*stdlog);
00088   debugStream.setLogger(*errlog);
00089
00090   parsers.setLogFILE("/dev/null");
00091
00092   knownStreams["<stdout>"] = stdlog;
00093   knownStreams["<stderr>"] = errlog;
00094
00095   setMineruleSourceName("default-mr-name");
00096   setMineruleName("default-mr-name");
00097
00098   ready = true;
00099 }
00100
00101 void MineruleOptions::readFromFile(std::string filename) {
00102   FILE *file = fopen(filename.c_str(), "r");
00103
00104   try {
00105     if (file == NULL) {
00106       throw MineruleException(MR_ERROR_INPUT_FILE_NOT_FOUND,
00107                               "Cannot open file:" + filename);
00108     }
00109     initializeOptionsFromFile(*this, file);
```

```
00110     mrdb_db.resetConnection();
00111   } catch (MineruleException &e) {
00112     if (file != NULL)
00113       fclose(file);
00114
00115     throw;
00116   } catch (mrdb::SQLException &e) {
00117     if (file != NULL)
00118       fclose(file);
00119
00120     throw;
00121   }
00122
00123   fclose(file);
00124 }
00125
00126 void MineruleOptions::readFromString(const std::string &str) {
00127   initializeOptionsFromString(*this, str);
00128
00129   mrdb_db.resetConnection();
00130 }
00131
00132 std::ostream &MineruleOptions::saveOptions(std::ostream &os) const {
00133   os « "# Options related to the MRDB connection" « std::endl;
00134   os « "mrdb::{" « std::endl « "  +name=" « getMRDB().getName() « std::endl
00135      « "  +username=" « getMRDB().getUsername() « std::endl
00136      « "  +password=" « getMRDB().getPassword() « std::endl
00137      « "  +cacheWrites=" « Converter(getMRDB().getCacheWrites()).toString()
00138      « std::endl « "# dbms allows one to specify the underlying dbms, "
00139                    "supported dbms are presently" « std::endl
00140      « "# mysql and postgres" « std::endl « "  +dbms=" « getMRDB().getDBMS()
00141      « std::endl « "}" « std::endl « std::endl;
00142
00143   os « "# Options related to data safety issues" « std::endl;
00144   os « "safety::{" « std::endl
00145      « "# if the following option is set to 'True', then the" « std::endl
00146      « "# system will delete old results whenever a new minerule" « std::endl
00147      « "# having the same name of an old one is inserted. Otherwise"
00148      « std::endl « "# the system will report an error message and exit."
00149      « std::endl « "  +overwriteHomonymMinerules="
00150      « Converter(getSafety().getOverwriteHomonymMinerules()).toString()
00151      « std::endl « "# if overwriteHomonymMinerules is set to True, then the"
00152      « std::endl
00153      « "# following option decides whether the system should delete"
00154      « std::endl « "# also the minerules for which the result depends on the "
00155      « std::endl « "# deleted one. If the option is set to True, those "
00156      « std::endl « "# minerule will be deleted as well, otherwise the system "
00157      « std::endl « "# with halt reporting an error." « std::endl
00158      « "  +allowCascadeDeletes="
00159      « Converter(getSafety().getAllowCascadeDeletes()).toString() « std::endl
00160      « "}" « std::endl « std::endl;
00161
00162   os « "# Options related to mining algorithms" « std::endl;
00163   os « "miningalgorithms::{" « std::endl;
00164   os « "  #options for configuring rule mining algorithms" « std::endl;
00165   os « "  rulesmining::{" « std::endl;
00166   os « "    +preferredAlgorithm="
00167      « algorithmTypeToString(getMiningAlgorithms()
00168                                .getRulesMiningAlgorithms()
00169                                .getPreferredAlgorithm()) « std::endl
00170      « std::endl;
00171   os « "    # Options related to PartitionBase algorithm" « std::endl;
00172   os « "    partitionbase::{" « std::endl « "      +rowsPerPartition="
00173      « getMiningAlgorithms()
00174            .getRulesMiningAlgorithms()
00175            .getPartitionBase()
00176            .getRowsPerPartition() « std::endl « "    }" « std::endl
00177      « std::endl;
00178
00179   os « "    # Options related to PartitionWithClusters algorithm" « std::endl;
00180   os « "    partitionwithclusters::{" « std::endl
00181      « "      +rowsPerPartition="
00182      « getMiningAlgorithms()
00183            .getRulesMiningAlgorithms()
00184            .getPartitionWithClusters()
00185            .getRowsPerPartition() « std::endl « "    }" « std::endl
00186      « std::endl;
00187
00188   std::string algoType;
00189   if (getMiningAlgorithms()
00190         .getRulesMiningAlgorithms()
00191         .getFPGrowth()
00192         .getAlgoType() ==
00193       MiningAlgorithms::RulesMiningAlgorithms::FPGrowth::Original)
00194     algoType = "Original";
00195   else
00196     algoType = "SingleReorder";
```

```
00197
00198   os « "    # Options related to FPGrowth algorithms" « std::endl;
00199   os « "    fpgrowth::{" « std::endl « "      +algoType=" « algoType
00200      « std::endl « "    }" « std::endl;
00201   os « "  }" « std::endl;
00202
00203   os « " itemsetsmining::{" « std::endl;
00204   os « "    +preferredAlgorithm="
00205      « algorithmTypeToString(getMiningAlgorithms()
00206                                  .getItemsetsMiningAlgorithms()
00207                                  .getPreferredAlgorithm()) « std::endl
00208      « std::endl;
00209   os « " }" « std::endl;
00210   os « "}" « std::endl « std::endl;
00211
00212   std::string optimizations;
00213   if (getOptimizations().getTryOptimizations())
00214     optimizations = "True";
00215   else
00216     optimizations = "False";
00217   std::string incrAlgorithm;
00218   switch (getOptimizations().getIncrementalAlgorithm()) {
00219   case Optimizations::ConstructiveAlgo:
00220     incrAlgorithm = "constructive";
00221     break;
00222   case Optimizations::DestructiveAlgo:
00223     incrAlgorithm = "destructive";
00224     break;
00225   case Optimizations::AutochooseIncrAlgo:
00226     incrAlgorithm = "Auto";
00227     break;
00228   }
00229
00230   os « "# Options related to Optimizations" « std::endl;
00231   os « "optimizations::{" « std::endl
00232      « "  +enableOptimizations=" « optimizations « std::endl
00233      « "# If set to True, this option will disable the detection of dominant"
00234      « std::endl « "# queries (this imply also that the system will not try "
00235                    "to find equivalent" « std::endl
00236      « "# queries, since they are a particular case of dominance)" « std::endl
00237      « "  +avoidDominanceDetection="
00238      « Converter(getOptimizations().getAvoidDominanceDetection()).toString()
00239      « std::endl « "# If set to True this option will make the optimizer to "
00240      « std::endl
00241      « "# consider equivalent queries as if they were dominant ones"
00242      « std::endl « "# (i.e., it will call an incremental algorithm instead of"
00243      « std::endl « "# dealing with the equivalence)." « std::endl
00244      « "  +avoidEquivalenceDetection="
00245      « Converter(getOptimizations().getAvoidEquivalenceDetection()).toString()
00246      « std::endl « "# If set to True the optimizer will not try to find "
00247      « std::endl
00248      « "# a combinations of previous queries equivalent to the current one."
00249      « std::endl
00250      « "# Notice that the search for combination may be a slow process"
00251      « std::endl « "  +avoidCombinationDetection="
00252      « Converter(getOptimizations().getAvoidCombinationDetection()).toString()
00253      « std::endl « "# The following option allows the user to specify how a "
00254      « std::endl
00255      « "# particular incremental algorithm  have to be chosen. The"
00256      « std::endl « "# following values are allowed:{Constructive,Destructive,"
00257      « std::endl « "# Auto}" « std::endl
00258      « "# Constructive and Destructive force the corresponding " « std::endl
00259      « "# algorithm to be chosen. " « std::endl
00260      « "# Auto leaves the choice to the optimizer." « std::endl
00261      « "  +incrementalAlgorithm=" « incrAlgorithm « std::endl
00262      « "# Options related to the query combinator algorithm" « std::endl
00263      « "  combinator::{" « std::endl
00264      « "# amount of time the search for a combination is allowed to run "
00265      « std::endl « "    +timeOut="
00266      « Converter(getOptimizations().getCombinator().getTimeOutThreshold())
00267            .toString() « std::endl « "# Max number of disjuncts. It is the "
00268                                      "number of disjuncts that is considered"
00269      « std::endl « "# during the search. Notice that increasing this number "
00270                    "has a strong impact" « std::endl
00271      « "# on the dimension of the search space." « std::endl
00272      « "    +maxDisjuncts="
00273      « Converter(long(getOptimizations().getCombinator().getMaxDisjuncts()))
00274            .toString() « std::endl « "# Max number of queries. Max number "
00275                                      "of distinct queries the user allows to"
00276      « std::endl « "# be combined in the result. Formulae with a larger "
00277                    "number of queries are" « std::endl
00278      « "# penalized in the evaluation function." « std::endl
00279      « "    +maxQueries="
00280      « Converter(long(getOptimizations().getCombinator().getMaxQueries()))
00281            .toString() « std::endl « "# Max distinct predicates. Max number "
00282                                      "of distinct predicates that the user"
00283      « std::endl « "# allows. This afflict the response time: the time spent "
```

```
00284                     "in assessing each" « std::endl
00285      « "# formula grows exponentially fast with the number of predicates."
00286      « std::endl « "    +maxDistinctPredicates="
00287      « Converter(
00288            long(getOptimizations().getCombinator().getMaxDistinctPredicates()))
00289            .toString() « std::endl « "  }" « std::endl « "}" « std::endl
00290      « std::endl;
00291   os « "# Options related to the parsing algorithms" « std::endl;
00292   os « "parsers::{" « std::endl;
00293   os « "# Parsers log stream, valid names are:" « std::endl;
00294   os « "#   <stdout>, <stderr> and any writeable file." « std::endl;
00295   os « " +logfile=/dev/null" « std::endl;
00296   os « "# The following four options allows to set constraint on" « std::endl
00297      « "# cardinalities of elements which appears in the body/head"
00298      « std::endl
00299      « "# part of rules. The constraints set here 'win' on the ones"
00300      « std::endl
00301      « "# in minerules (i.e., if you say '1..n' as BODY in your minerule"
00302      « std::endl
00303      « "# but set it to 1..5 here, than 1..5 will be used instead."
00304      « std::endl;
00305   os « " +minBodyElems=" « getParsers().getBodyCardinalities().getMin()
00306      « std::endl;
00307   os « " +maxBodyElems=" « getParsers().getBodyCardinalities().getMax()
00308      « std::endl;
00309   os « " +minHeadElems=" « getParsers().getHeadCardinalities().getMin()
00310      « std::endl;
00311   os « " +maxHeadElems=" « getParsers().getHeadCardinalities().getMax()
00312      « std::endl;
00313   os « "}" « std::endl « std::endl;
00314
00315   os « "# Options related to streams, note that they are commented."
00316      « std::endl
00317      « "# the reason is that the following conresponds to default "
00318      « std::endl « "# settings instead of the actual ones." « std::endl
00319      « "# In specifying the stream parameter, you can:" « std::endl
00320      « "# 1) Specify a file path" « std::endl
00321      « "# 2) Specify <stdout>,<stderr> in order to specify the standard"
00322      « std::endl « "#    output and the standard error respectively"
00323      « std::endl « "# 3) Specify a file path including the %m and %i symbols"
00324      « std::endl
00325      « "# In case 3) %m is expandend to the current minerule name as"
00326      « std::endl
00327      « "# it appear in the minerule text, %i is expanded to the value"
00328      « std::endl « "# of the -i parameter if any, to 'mr' otherwise"
00329      « std::endl
00330      « "# The loglevel option allows you to select how deep the log nesting"
00331      « std::endl « "# can grow. The default (loglevel==100) means: \"grow as "
00332                  "much as needed\"" « std::endl
00333      « "# (no function in minerule will ever nest logs more than a few "
00334        "levels)." « std::endl « "# If set to 0, then the log is actually "
00335                                 "suppressed (the first level is 1)."
00336      « std::endl
00337      « "# Otherwise, if set to n, only the first n levels will be displayed."
00338      « std::endl « "#" « std::endl « "# logstream::{" « std::endl
00339      « "#    +stream=<stdout> " « std::endl « "#    +loglevel=100"
00340      « std::endl « "# }" « std::endl « "# errstream::{" « std::endl
00341      « "#    +stream=<stderr> " « std::endl « "#    +loglevel=100"
00342      « std::endl « "# }" « std::endl « "# warnstream::{" « std::endl
00343      « "#    +stream=<stdout> " « std::endl « "#    +loglevel=100"
00344      « std::endl « "# }" « std::endl « "# debugstream::{" « std::endl
00345      « "#    +stream=<stderr> " « std::endl « "#    +loglevel=100"
00346      « std::endl « "# }" « std::endl;
00347
00348   return os;
00349 }
00350
00351 void MineruleOptions::Parsers::setLogFILE(const std::string &fname) {
00352   clearStream();
00353   logfile = fopen(fname.c_str(), "w");
00354   if (logfile == NULL)
00355     throw MineruleException(
00356         MR_ERROR_OUTPUT_FILE_PROBLEM,
00357         std::string("Error while parsing options,I've tried to open file:") +
00358             fname + std::string(", but an error occurred") +
00359             std::string(" the reason is:\n") + strerror(errno));
00360 }
00361
00362 void MineruleOptions::Parsers::setLogOnStdout() { logfile = stdout; }
00363
00364 void MineruleOptions::Parsers::setLogOnStderr() { logfile = stderr; }
00365
00366 void MineruleOptions::Mrdb::setOption(
00367     const std::string &name,
00368     const std::string &value) {
00369   if (name == "name")
00370     setName(value);
```

```
00371   else if (name == "username")
00372     setUsername(value);
00373   else if (name == "password")
00374     setPassword(value);
00375   else if (name == "cacheWrites")
00376     setCacheWrites(Converter(value).toBool());
00377   else if (name == "dbms")
00378     setDBMS(value);
00379   else {
00380     std::cerr « "Error while parsing options, expecting an mrdb option in:"
00381              « std::endl
00382              « "{name, userName, password, cacheWrites,dbms} and: "
00383              « std::endl « "\"" « name « "\" found." « std::endl;
00384     throw MineruleException(MR_ERROR_OPTION_PARSING,
00385                             MINERULE_OPTIONS_PARSING_ERROR);
00386   }
00387 }
00388
00389 void MineruleOptions::Safety::setOption(
00390     const std::string &name,
00391     const std::string &value) {
00392   try {
00393     if (name == "overwriteHomonymMinerules") {
00394       setOverwriteHomonymMinerules(Converter(value).toBool());
00395     } else if (name == "allowCascadeDeletes") {
00396       setAllowCascadeDeletes(Converter(value).toBool());
00397     } else {
00398       throw MineruleException(MR_ERROR_OPTION_PARSING,
00399                               "Expected a value in "
00400                               "{overwriteHomonymMinerules,allowCascadeDeletes},"
00401                               " but " +
00402                                   value + " found.");
00403     }
00404   } catch (MineruleException &e) {
00405     std::cerr
00406         « "Parsing error while parsing a safety option(given option name:"
00407         « name « " given option value:" « value « "). The reason for"
00408         « " the error is:" « e.what() « std::endl;
00409     throw MineruleException(MR_ERROR_OPTION_PARSING,
00410                             MINERULE_OPTIONS_PARSING_ERROR);
00411   }
00412 }
00413
00414 void MineruleOptions::MiningAlgorithms::RulesMiningAlgorithms::PartitionBase::
00415     setOption(const std::string &name,
00416               const std::string &value) {
00417   if (name == "rowsPerPartition") {
00418     unsigned int rpp = stringToLong(value, name);
00419     setRowsPerPartition(rpp);
00420   } else {
00421     std::cerr « "Error while parsing options, expecting a partition option in:"
00422              « std::endl « "{rowsPerPartition} and: " « std::endl « "\""
00423              « name « "\" found." « std::endl;
00424     throw MineruleException(MR_ERROR_OPTION_PARSING,
00425                             MINERULE_OPTIONS_PARSING_ERROR);
00426   }
00427 }
00428
00429 void MineruleOptions::MiningAlgorithms::RulesMiningAlgorithms::
00430     PartitionWithClusters::setOption(
00431         const std::string &name,
00432         const std::string &value) {
00433   if (name == "rowsPerPartition") {
00434     unsigned int rpp = stringToLong(value, name);
00435     setRowsPerPartition(rpp);
00436   } else {
00437     std::cerr « "Error while parsing options, expecting a partition option in:"
00438              « std::endl « "{rowsPerPartition} and: " « std::endl « "\""
00439              « name « "\" found." « std::endl;
00440     throw MineruleException(MR_ERROR_OPTION_PARSING,
00441                             MINERULE_OPTIONS_PARSING_ERROR);
00442   }
00443 }
00444
00445 void MineruleOptions::MiningAlgorithms::RulesMiningAlgorithms::FPGrowth::
00446     setOption(const std::string &name,
00447               const std::string &value) {
00448   if (name == "algoType") {
00449     if (value == "Original") {
00450       setAlgoType(Original);
00451     } else if (value == "SingleReorder") {
00452       setAlgoType(SingleReorder);
00453     } else {
00454       std::cerr
00455           « "Error while parsing options, expecting a fpgrowth::algoType "
00456           « " value in:" « std::endl
00457           « "{Original,SingleReorder} and: " « std::endl « "\"" « value
```

```
00458           « "\" found." « std::endl;
00459       throw MineruleException(MR_ERROR_OPTION_PARSING,
00460                               MINERULE_OPTIONS_PARSING_ERROR);
00461     }
00462   } else {
00463     std::cerr « "Error while parsing options, expecting a fpgrowth option in:"
00464               « std::endl « "{algoType} and: " « std::endl « "\"" « name
00465               « "\" found." « std::endl;
00466     throw MineruleException(MR_ERROR_OPTION_PARSING,
00467                             MINERULE_OPTIONS_PARSING_ERROR);
00468   }
00469 }
00470
00471 void MineruleOptions::MiningAlgorithms::RulesMiningAlgorithms::setOption(
00472     const std::string &name,
00473     const std::string &value) {
00474   if (name == "preferredAlgorithm") {
00475     try {
00476       setPreferredAlgorithm(stringToAlgorithmType(value));
00477     } catch (MineruleException e) {
00478       std::cerr
00479           « "Error while parsing options, expecting an algorithm option in {"
00480           « stringWithListOfAlgorithmTypes() « "}, but \"" « value
00481           « "\" found." « std::endl;
00482       throw MineruleException(MR_ERROR_OPTION_PARSING,
00483                               MINERULE_OPTIONS_PARSING_ERROR);
00484     }
00485   } else {
00486     std::cerr « "Error while parsing options, expecting a miningalgorithms "
00487                 "option in:" « std::endl
00488               « "{preferredAlgorithm} and: " « std::endl « "\"" « name
00489               « "\" found." « std::endl;
00490     throw MineruleException(MR_ERROR_OPTION_PARSING,
00491                             MINERULE_OPTIONS_PARSING_ERROR);
00492   }
00493 }
00494
00495 void MineruleOptions::MiningAlgorithms::ItemsetsMiningAlgorithms::setOption(
00496     const std::string &name,
00497     const std::string &value) {
00498   if (name == "preferredAlgorithm") {
00499     try {
00500       setPreferredAlgorithm(stringToAlgorithmType(value));
00501     } catch (MineruleException e) {
00502       std::cerr
00503           « "Error while parsing options, expecting an algorithm option in {"
00504           « stringWithListOfAlgorithmTypes() « "}, but \"" « value
00505           « "\" found." « std::endl;
00506       throw MineruleException(MR_ERROR_OPTION_PARSING,
00507                               MINERULE_OPTIONS_PARSING_ERROR);
00508     }
00509   } else {
00510     std::cerr « "Error while parsing options, expecting a miningalgorithms "
00511                 "option in:" « std::endl
00512               « "{preferredAlgorithm} and: " « std::endl « "\"" « name
00513               « "\" found." « std::endl;
00514     throw MineruleException(MR_ERROR_OPTION_PARSING,
00515                             MINERULE_OPTIONS_PARSING_ERROR);
00516   }
00517 }
00518
00519 void MineruleOptions::MiningAlgorithms::SequencesMiningAlgorithms::setOption(
00520     const std::string &name,
00521     const std::string &value) {
00522   if (name == "preferredAlgorithm") {
00523     try {
00524       setPreferredAlgorithm(stringToAlgorithmType(value));
00525     } catch (MineruleException e) {
00526       std::cerr
00527           « "Error while parsing options, expecting an algorithm option in {"
00528           « stringWithListOfAlgorithmTypes() « "}, but \"" « value
00529           « "\" found." « std::endl;
00530       throw MineruleException(MR_ERROR_OPTION_PARSING,
00531                               MINERULE_OPTIONS_PARSING_ERROR);
00532     }
00533   } else {
00534     std::cerr « "Error while parsing options, expecting a miningalgorithms "
00535                 "option in:" « std::endl
00536               « "{preferredAlgorithm} and: " « std::endl « "\"" « name
00537               « "\" found." « std::endl;
00538     throw MineruleException(MR_ERROR_OPTION_PARSING,
00539                             MINERULE_OPTIONS_PARSING_ERROR);
00540   }
00541 }
00542
00543 void MineruleOptions::Optimizations::setOption(
00544     const std::string &name,
```

```
00545     const std::string &value) {
00546   if (name == "enableOptimizations") {
00547     if (value == "True") {
00548       setTryOptimizations(true);
00549     } else if (value == "False") {
00550       setTryOptimizations(false);
00551     } else {
00552       std::cerr « "Error while parsing options, expecting an "
00553                « "optimizations::tryOptimizations value in " « std::endl
00554                « "{True,False} and: " « std::endl « "\"" « value
00555                « "\" found." « std::endl;
00556       throw MineruleException(MR_ERROR_OPTION_PARSING,
00557                               MINERULE_OPTIONS_PARSING_ERROR);
00558     }
00559   } else if (name == "incrementalAlgorithm") {
00560     if (value == "Constructive") {
00561       setIncrementalAlgorithm(Optimizations::ConstructiveAlgo);
00562     } else if (value == "Destructive") {
00563       setIncrementalAlgorithm(Optimizations::DestructiveAlgo);
00564     } else if (value == "Auto") {
00565       setIncrementalAlgorithm(Optimizations::AutochooseIncrAlgo);
00566     } else {
00567       std::cerr « "Error while parsing options, expecting an "
00568                   "optimizations::incrementalAlgorithm value" « std::endl
00569                « "in {Constructive,Destructive,Auto}, but \"" + value +
00570                      "\" found." « std::endl;
00571       throw MineruleException(MR_ERROR_OPTION_PARSING,
00572                               MINERULE_OPTIONS_PARSING_ERROR);
00573     }
00574   } else if (name == "avoidDominanceDetection") {
00575     try {
00576       setAvoidDominanceDetection(Converter(value).toBool());
00577     } catch (MineruleException &e) {
00578       std::cerr
00579           « "Error while parsing avoidDominanceDetection option, expecting "
00580             "a value in {True, False}, but " « value « "found" « std::endl;
00581       throw MineruleException(MR_ERROR_OPTION_PARSING,
00582                               MINERULE_OPTIONS_PARSING_ERROR);
00583     }
00584   } else if (name == "avoidEquivalenceDetection") {
00585     try {
00586       setAvoidEquivalenceDetection(Converter(value).toBool());
00587     } catch (MineruleException &e) {
00588       std::cerr
00589           « "Error while parsing avoidEquivalenceDetection option, expecting "
00590             "a value in {True, False}, but " « value « "found" « std::endl;
00591       throw MineruleException(MR_ERROR_OPTION_PARSING,
00592                               MINERULE_OPTIONS_PARSING_ERROR);
00593     }
00594   } else if (name == "avoidCombinationDetection") {
00595     try {
00596       setAvoidCombinationDetection(Converter(value).toBool());
00597     } catch (MineruleException &e) {
00598       std::cerr
00599           « "Error while parsing avoidCombinationDetection option, expecting "
00600             "a value in {True, False}, but " « value « "found" « std::endl;
00601       throw MineruleException(MR_ERROR_OPTION_PARSING,
00602                               MINERULE_OPTIONS_PARSING_ERROR);
00603     }
00604   } else {
00605     std::cerr
00606         « "Error while parsing options, expecting an optimization option in:"
00607         « std::endl « "{enableOptimizations, incrementalAlgorithm, "
00608                       "avoidEquivalenceDetection"
00609         « ", tryOptimizationThruCombination} and: " « std::endl « "\""
00610         « name « "\" found." « std::endl;
00611     throw MineruleException(MR_ERROR_OPTION_PARSING,
00612                             MINERULE_OPTIONS_PARSING_ERROR);
00613   }
00614 }
00615
00616 void MineruleOptions::Optimizations::Combinator::setOption(
00617     const std::string &name,
00618     const std::string &value) {
00619
00620   if (name == "timeOut")
00621     setTimeOutThreshold(Converter(value).toDouble());
00622   else if (name == "maxDisjuncts")
00623     setMaxDisjuncts(Converter(value).toLong());
00624   else if (name == "maxQueries")
00625     setMaxQueries(Converter(value).toLong());
00626   else if (name == "maxDistinctPredicates")
00627     setMaxDistinctPredicates(Converter(value).toLong());
00628   else {
00629     std::cerr « "Error while parsing options, expecting a combinator option i:"
00630              « std::endl
00631              « "{timeOut,maxDisjuncts,maxQueries,maxDistinctPredicates}, but"
```

```
00632               « std::endl « "\"" « name « "\" found." « std::endl;
00633
00634     throw MineruleException(MR_ERROR_OPTION_PARSING,
00635                             MINERULE_OPTIONS_PARSING_ERROR);
00636   }
00637 }
00638
00639 void MineruleOptions::OutStream::setOption(
00640     const std::string &name,
00641     const std::string &value) {
00642   // we need to modify value, hence we made a copy of it
00643   // and use it in the rest of the procedure
00644   std::string valueCopy = value;
00645   std::map<std::string, MRLogger *> &knownStreams =
00646       getSharedOptions().knownStreams;
00647
00648   if (name == "stream") {
00649     size_t markerPos;
00650     if ((markerPos = valueCopy.find("%i")) != valueCopy.npos) {
00651       valueCopy.erase(markerPos, 2);
00652       valueCopy.insert(markerPos, options->getMineruleSourceName());
00653     }
00654
00655     if ((markerPos = valueCopy.find("%m")) != valueCopy.npos) {
00656       valueCopy.erase(markerPos, 2);
00657       valueCopy.insert(markerPos, options->getMineruleName());
00658     }
00659
00660     if (knownStreams.find(valueCopy) == knownStreams.end()) {
00661       std::ostream *ostrptr(new std::ofstream(valueCopy.c_str()));
00662
00663       if (!*ostrptr) {
00664         std::cerr « "Error while parsing options, failure while trying to"
00665                  « std::endl « "open in output the following file "
00666                  « valueCopy « std::endl
00667                  « "Possible values for the stream options are:"
00668                  « "{<stdin>,<stdout>,anyStringRepresentingAFileName}"
00669                  « std::endl « "Note you can also use file names having the "
00670                                "modifier %i or %m inside." « std::endl
00671                  « "In that case %i is expandended into the current value of "
00672                     "the -i parameter" « std::endl
00673                  « "(provided that the -i parameter is used instead of the "
00674                     "-m one, otherwise, the" « std::endl
00675                  « "default 'mr' value is assumed). Analogously, the %m "
00676                     "parameter will be" « std::endl
00677                  « "substituted with the current minerule name (as it appear "
00678                     "in the minerule text" « std::endl;
00679         throw MineruleException(MR_ERROR_OPTION_PARSING,
00680                                 MINERULE_OPTIONS_PARSING_ERROR);
00681       }
00682
00683       MRLogger *logger = new MRLogger(*ostrptr);
00684       knownStreams[valueCopy] = logger;
00685     }
00686
00687     setLogger(*knownStreams[valueCopy]);
00688   } else if (name == "loglevel") {
00689     try {
00690       size_t level = Converter(valueCopy).toLong();
00691       setLogLevel(level);
00692     } catch (MineruleException &e) {
00693       throw MineruleException(
00694           MR_ERROR_OPTION_PARSING,
00695           "Error parsing options while converting loglevel value `" +
00696               valueCopy + "' to int.");
00697     }
00698   } else {
00699     std::cerr « "Error while parsing options, expecting a stream option in:"
00700              « std::endl « "{stream} and: " « std::endl « "\"" « name
00701              « "\" found." « std::endl;
00702     throw MineruleException(MR_ERROR_OPTION_PARSING,
00703                             MINERULE_OPTIONS_PARSING_ERROR);
00704   }
00705 }
00706
00707 void MineruleOptions::Parsers::setOption(
00708     const std::string &name,
00709     const std::string &value) {
00710   if (name == "logfile") {
00711     if (value == "<stdout>")
00712       setLogOnStdout();
00713     else if (value == "<stderr>")
00714       setLogOnStderr();
00715     else
00716       setLogFILE(value);
00717   } else if (name == "minBodyElems") {
00718     int min = stringToLong(value, name);
```

```
00719     setMinBodyElems(min);
00720   } else if (name == "maxBodyElems") {
00721     int max = stringToLong(value, name);
00722     setMaxBodyElems(max);
00723   } else if (name == "minHeadElems") {
00724     int min = stringToLong(value, name);
00725     setMinHeadElems(min);
00726   } else if (name == "maxHeadElems") {
00727     int max = stringToLong(value, name);
00728     setMaxHeadElems(max);
00729   } else {
00730     std::cerr « "Error while parsing options, expecting a parser option in:"
00731               « std::endl « "{logfile, minBodyElems, maxBodyElems, "
00732                             "minHeadElems, maxHeadElems} and: " « std::endl
00733               « "\"" « name « "\" found." « std::endl;
00734     throw MineruleException(MR_ERROR_OPTION_PARSING,
00735                             MINERULE_OPTIONS_PARSING_ERROR);
00736   }
00737 }
00738
00739 } // namespace
```

7.375 /Users/esposito/Software/minerule/src/Utils/MRLogger.cpp File Reference

```
#include <cassert>
#include "minerule/Utils/MRLogger.hpp"
#include "minerule/Utils/StringUtils.hpp"
```

Namespaces

- namespace minerule

7.376 MRLogger.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include <cassert>
00017
00018 #include "minerule/Utils/MRLogger.hpp"
00019 #include "minerule/Utils/StringUtils.hpp"
00020
00021 namespace minerule {
00022
00023   const std::string MRLogger::START_SEPARATOR="+";
00024   const std::string MRLogger::CONT_SEPARATOR="|";
00025   const std::string MRLogger::END_SEPARATOR="`--";
00026   std::ofstream MRLogger::nullLog("/dev/null");
00027
00028
00029  std::string MRLogger::evalTimeMemInfo(const LogInfo& li) const {
00030     std::stringstream ss;
00031     ss«"[cpu:"
```

```
00032       « getCpuSecs(li,logStack.front())
00033       «" d-cpu:"
00034       « getCpuSecs(li,logStack.back())
00035       «" time:"
00036       « getTimeSecs(li,logStack.front())
00037       «" d-time:"
00038       « getTimeSecs(li,logStack.back())
00039       «"]";
00040     return ss.str();
00041   }
00042
00043   double MRLogger::getCurrentCpuSecs() const {
00044     LogInfo curInfo("");
00045     return getCpuSecs(curInfo,logStack.front());
00046   }
00047
00048   double MRLogger::getCurrentCpuDelta() const {
00049     LogInfo curInfo("");
00050     return getCpuSecs(curInfo,logStack.back());
00051   }
00052
00053   double MRLogger::getCurrentTimeSecs() const {
00054     LogInfo curInfo("");
00055     return getTimeSecs(curInfo,logStack.front());
00056   }
00057
00058   double MRLogger::getCurrentTimeDelta() const {
00059     LogInfo curInfo("");
00060     return getTimeSecs(curInfo,logStack.back());
00061   }
00062
00063   MRLogger::MRLogger(void) : indentInset("  "), os(NULL), logLevel(100), curLogLevel(1) {
00064     updateIndentString();
00065   }
00066
00067   MRLogger::MRLogger(std::ostream& ostr) : indentInset("  "), os(NULL), logLevel(100), curLogLevel(1)
      {
00068     setStream(ostr);
00069     updateIndentString();
00070   }
00071
00072   void MRLogger::setStream(std::ostream& ostr)  {
00073     os=&ostr;
00074   };
00075
00076   MRLogger::~MRLogger(void) {
00077     while(!logStack.empty()) {
00078       logStack.pop_back();
00079     }
00080   }
00081
00082   void MRLogger::updateIndentString() {
00083     LogStack::const_iterator it;
00084     indentString="";
00085     for(it=logStack.begin(); it!=logStack.end(); it++) {
00086       indentString+=it->indent;
00087     }
00088   }
00089
00090   void MRLogger::push(const std::string& descr) {
00091     curLogLevel++;
00092     if(curLogLevel>logLevel)
00093       return;
00094
00095     assert(os!=NULL);
00096     indent();
00097     *os«std::endl;
00098
00099    std::string insetSep;
00100     if(logStack.empty())
00101       insetSep="";
00102     else
00103       insetSep=indentInset;
00104
00105     LogInfo li(insetSep+CONT_SEPARATOR);
00106     logStack.push_back(li);
00107
00108     indent();
00109     *os«insetSep«"+"« StringUtils::toBold(descr) «std::endl;
00110     updateIndentString();
00111   }
00112
00113
00114
00115   void MRLogger::pop() {
00116     if(logStack.empty()) return;
00117
```

```
00118     if(curLogLevel>logLevel) {
00119       curLogLevel--;
00120       return;
00121     } else {
00122       curLogLevel--;
00123     }
00124     assert(os!=NULL);
00125     LogInfo curInfo("");
00126     std::string timeMemInfo=evalTimeMemInfo(curInfo);
00127 
00128     logStack.pop_back();
00129     updateIndentString();
00130     indent();
00131 
00132     if(!logStack.empty())
00133       *os«indentInset«END_SEPARATOR« StringUtils::toBold(timeMemInfo) «std::endl;
00134     else
00135           *os«END_SEPARATOR«StringUtils::toBold(timeMemInfo)«std::endl;
00136 
00137     indent();
00138     *os«std::endl;
00139   }
00140 
00141   void MRLogger::logMeasurement(const std::string& description, const MeasurementInfo& data) {
00142           log() « StringUtils::toGreen("tag: ") « description « StringUtils::toGreen(" cpu time:") «
      data.totCpu « StringUtils::toGreen(" time:") « data.totTime « std::endl;
00143   }
00144 
00145   void MRLogger::logMeasurements() {
00146           push("Showing measured execution times:");
00147 
00148                 for( Measurements::const_iterator it=measurements.begin(); it!=measurements.end();
      ++it ) {
00149                   logMeasurement(it->first, it->second);
00150           }
00151 
00152           pop();
00153   }
00154 
00155 
00156 } // minerule
```

7.377 /Users/esposito/Software/minerule/src/Utils/OptionParser.ypp File Reference

7.378 OptionParser.ypp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 /*
00017   ----------------------------------------------------------------------------
00018   Minerule Option Parser
00019   ----------------------------------------------------------------------------
00020 
00021   A parser for the minerule option file. Minerule options are defined inside
00022   (possibly nested) option namespaces. Each namespace is a subclass of OptionBase.
00023   Each OptionBase subclass defines the sub-namespace and the options it supports.
00024   The root namespace is defined in 'MineruleOptions_implementations/root.h'.
00025 
00026   The syntax for accessing a namespace is:
00027           namespace::
00028   after a namespace declaration one can instantly declare an option or open a context
00029   for subsequent operations.
```

```
00030   The syntax for setting an option is:
00031         +optionName=optionValue
00032   Putting both things together:
00033           namespace::+optionName=optionValue
00034   sets option 'optionName' in namespace 'namespace' to the value 'optionValue'.
00035   Nested namespace are simply appended one to another:
00036           ns1::ns2::ns3::option=optionValue
00037   sets  'option' in namespace ns3 which is a sub-namespace of ns2 which is sub-namespace of ns1.
00038
00039   The parser supports an additional syntax that simplifies setting many options in the
00040   same namespace:
00041   namespace::{
00042           ....
00043   }
00044   everything in the braces are interpreted in the context of 'namespace'.
00045
00046   *NOTE*
00047   The syntax namespace1::namespace2::+option=optionValue is meant to be used to set options
00048   from the command line.
00049   The syntax
00050         namespace1::{
00051                 namespace2::{
00052                         +option=optionValue
00053                 }
00054         }
00055   is meant for option files. This is important to know because using the first syntax in an option
00056   file will not work since the parser does *not* support more than one option set using the first
00057   syntax (this is ok using command line options since a new instance of the parser will be
00058   instantiated for each option).
00059 */
00060
00061 %{
00062 #define YYSTYPE std::string
00063
00064 #include <iostream>
00065 #include "minerule/Utils/OptionParserLib.hpp"
00066
00067
00068 using namespace minerule;
00069
00070 extern int yylex();
00071 extern int yyerror(char const*);
00072
00073 %}
00074
00075 %error-verbose
00076 %locations
00077
00078 %token STRING
00079 %token QUOTEDSTRING
00080 %token EQSIM
00081 %token NSSEP
00082 %token STARTOPT
00083 %token LEFTBRACE
00084 %token RIGHTBRACE
00085
00086 %%
00087
00088 option_list: option option_list
00089         |option
00090         ;
00091
00092 option: STARTOPT STRING EQSIM  STRING  {
00093                 setOption($2,$4);
00094             }
00095         | LEFTBRACE option_list RIGHTBRACE {
00096                  popOptionClassFromContext();
00097             }
00098     | nsdecl_list option
00099         ;
00100
00101 nsdecl: STRING NSSEP {
00102                 pushOptionClassIntoContext($1);
00103             }
00104         ;
00105
00106 nsdecl_list: nsdecl nsdecl_list
00107         | nsdecl
00108
```

7.379 /Users/esposito/Software/minerule/src/Utils/OptionParserLib.cpp File Reference

```
#include "minerule/Utils/OptionParserLib.hpp"
#include "Utils/OptionParser_lexer.cpp"
```

Namespaces

- namespace minerule

Functions

- void OP_switch_to_buffer (YY_BUFFER_STATE new_buffer)
- void OP_delete_buffer (YY_BUFFER_STATE buffer)
- YY_BUFFER_STATE OP_scan_string (const char *str)
- void minerule::initializeOptionsFromFile (MineruleOptions &mrOpts, FILE *file)
- void minerule::initializeOptionsFromString (MineruleOptions &mrOpts, std::string str)
- void minerule::pushOptionClassIntoContext (const std::string &oclass)
- void minerule::popOptionClassFromContext ()
- void minerule::setOption (const std::string &name, const std::string &value)

Variables

- std::vector< MineruleOptions::OptionBase * > minerule::context

7.379.1 Function Documentation

7.379.1.1 OP_delete_buffer()

```
void OP_delete_buffer (
            YY_BUFFER_STATE buffer )
```

7.379.1.2 OP_scan_string()

```
YY_BUFFER_STATE OP_scan_string (
            const char * str )
```

7.379.1.3 OP_switch_to_buffer()

```
void OP_switch_to_buffer (
             YY_BUFFER_STATE new_buffer )
```

7.380 OptionParserLib.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/OptionParserLib.hpp"
00017 #include "Utils/OptionParser_lexer.cpp"
00018
00019 //extern void OPparse();
00020 //extern FILE* OPin;
00021 //struct YY_BUFFER_STATE;
00022
00023 void OP_switch_to_buffer( YY_BUFFER_STATE new_buffer );
00024 void OP_delete_buffer( YY_BUFFER_STATE buffer );
00025 YY_BUFFER_STATE OP_scan_string(const char *str);
00026 namespace minerule {
00027
00028
00029   std::vector<MineruleOptions::OptionBase*> context;// this maintain the context, i.e., it
00030                                     // sais to which "Option Class" new options
00031                                     // belong
00032
00033   void initializeOptionsFromFile(MineruleOptions& mrOpts, FILE* file) {
00034     YY_BUFFER_STATE buf = yy_new_buffer(file,YY_BUF_SIZE);
00035     OP_switch_to_buffer(buf);
00036     context.push_back(&mrOpts);
00037     OPparse();
00038     OP_delete_buffer(buf);
00039   }
00040
00041   void initializeOptionsFromString(MineruleOptions& mrOpts,std::string str) {
00042     YY_BUFFER_STATE buf = OP_scan_string(str.c_str());
00043     context.push_back(&mrOpts);
00044     OPparse();
00045     OP_delete_buffer(buf);
00046   }
00047
00048   void pushOptionClassIntoContext(const std::string& oclass)  {
00049     /* Here we have to push_back OptionBase pointers */
00050     assert(context.size()>0);
00051     context.push_back( &context.back()->subclassForName(oclass) );
00052   }
00053
00054   void popOptionClassFromContext() {
00055     assert(context.size()>=1);
00056     context.pop_back();
00057   }
00058
00059   void setOption(const std::string& name, const std::string& value) {
00060     assert(context.size()>=1);
00061
00062     context.back()->setOption(name,value);
00063   }
00064
00065
00066
00067
00068
00069
00070
00071
```

```
00072
00073
00074
00075 } // namespace
```

7.381 /Users/esposito/Software/minerule/src/Utils/Progress.cpp File Reference

```
#include "minerule/Utils/Progress.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/Utils/MineruleOptions.hpp"
#include <unistd.h>
#include <signal.h>
```

Namespaces

- namespace minerule

Functions

- void minerule::abort_handler (int s)
- void minerule::install_abort_handler ()
- void minerule::angledSpeenWheelUpdateHandler (int n, int count, int num)
- void minerule::barSpeenWheelUpdateHandler (int n, int count, int num)
- void minerule::tickNumberUpdateHandler (int n, int count, int num)
- void minerule::clockHandsUpdateHandler (int n, int count, int num)

7.382 Progress.cpp

Go to the documentation of this file.

```
00001 #include "minerule/Utils/Progress.hpp"
00002 #include "minerule/Utils/MineruleLogs.hpp"
00003 #include "minerule/Utils/MineruleOptions.hpp"
00004 #include <unistd.h>
00005 #include <signal.h>
00006
00007 namespace minerule {
00008
00009 Progress::UpdateHandler Progress::defaultHandler_(clockHandsUpdateHandler);
00010
00011 void abort_handler(int s){
00012         std::string msg = "Ctrl-C or Ctrl-Z caught \e[?25h";
00013         write(1, msg.c_str(), msg.size());
00014   exit(1);
00015 }
00016
00017
00018 void install_abort_handler() {
00019         static bool handlerInstalled = false;
00020         if(handlerInstalled) {
00021                 return;
00022         }
00023
00024   struct sigaction sigIntHandler;
00025
00026   sigIntHandler.sa_handler = abort_handler;
00027   sigemptyset(&sigIntHandler.sa_mask);
00028   sigIntHandler.sa_flags = 0;
00029
```

```
00030    handlerInstalled = !sigaction(SIGINT, &sigIntHandler, NULL);
00031 }
00032
00033 bool Progress::handleStartStop(std::ostream& logger, int n) {
00034         install_abort_handler();
00035
00036         if(n == -1) {
00037                 MRLog() « "  ";
00038                 logger « "\e[?251";
00039                 return true;
00040         }
00041
00042         if(n == -2) {
00043                 logger « std::endl;
00044                 logger « "\e[?25h";
00045                 return true;
00046         }
00047
00048         return false;
00049 }
00050
00051 void angledSpeenWheelUpdateHandler(int n, int count, int num) {
00052   static unsigned int step = 0;
00053   static char spinWheel[] = { '^', '>', 'v', '<' };
00054
00055         static std::ostream& logger =
00056                 *MineruleOptions::getSharedOptions()
00057                         .getLogStreamObj()
00058                         .getLogger()
00059                         .getStream();
00060
00061         if(Progress::handleStartStop(logger, n))
00062                 return;
00063
00064   if((count+n) % num) {
00065     logger « "\b" « spinWheel[step++ % 4];
00066                 logger.flush();
00067   }
00068 }
00069
00070 void barSpeenWheelUpdateHandler(int n, int count, int num) {
00071   static unsigned int step = 0;
00072   static char spinWheel[] = { '|', '/', '-', '\\' };
00073
00074         static std::ostream& logger =
00075                 *MineruleOptions::getSharedOptions()
00076                         .getLogStreamObj()
00077                         .getLogger()
00078                         .getStream();
00079
00080         if(Progress::handleStartStop(logger, n))
00081                 return;
00082
00083   if((count+n) % num) {
00084     logger « "\b" « spinWheel[step++ % 4];
00085                 logger.flush();
00086   }
00087 }
00088
00089 void tickNumberUpdateHandler(int n, int count, int num) {
00090         if(n<0) {
00091                 return;
00092         }
00093
00094         if((count+n) % num == 0) {
00095                 MRLog() « count+n « std::endl;
00096         }
00097 }
00098
00099 void clockHandsUpdateHandler(int n, int count, int num) {
00100   static unsigned long int step = 0;
00101   static std::string spinWheel[] =
00102           { "", "", "", "",
00103                 "", "", "", "",
00104                 "", "", "", "" };
00105
00106         static std::ostream& logger =
00107                 *MineruleOptions::getSharedOptions()
00108                         .getLogStreamObj()
00109                         .getLogger()
00110                         .getStream();
00111
00112         if(Progress::handleStartStop(logger, n)) {
00113                 return;
00114         }
00115
00116   if((count+n) % num == 0) {
```

```
00117     logger « "\b\b" « spinWheel[step++ % 12] « " ";
00118                 logger.flush();
00119   }
00120 }
00121
00122
00123
00124 }
```

7.383 /Users/esposito/Software/minerule/src/Utils/SQLUtils.cpp File Reference

```
#include <map>
#include <memory>
#include "minerule/Utils/SQLUtils.hpp"
#include "minerule/Utils/MineruleLogs.hpp"
#include "minerule/mrdb/Connection.hpp"
#include "minerule/mrdb/DatabaseMetaData.hpp"
```

Namespaces

- namespace minerule

7.384 SQLUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016
00017 #include <map>
00018 #include <memory>
00019
00020 #include "minerule/Utils/SQLUtils.hpp"
00021 #include "minerule/Utils/MineruleLogs.hpp"
00022 #include "minerule/mrdb/Connection.hpp"
00023 #include "minerule/mrdb/DatabaseMetaData.hpp"
00024
00025 namespace minerule {
00026
00027 SQLUtils::Type
00028 SQLUtils::getType(mrdb::Types::SQLType type) {
00029   switch (type) {
00030   case mrdb::Types::BIGINT:
00031   case mrdb::Types::DECIMAL:
00032   case mrdb::Types::DOUBLE:
00033   case mrdb::Types::FLOAT:
00034   case mrdb::Types::INTEGER:
00035   case mrdb::Types::NUMERIC:
00036   case mrdb::Types::REAL:
00037   case mrdb::Types::SMALLINT:
00038   case mrdb::Types::TINYINT:
00039     return Numeric;
00040   case mrdb::Types::CHAR:
```

```
00041   case mrdb::Types::VARCHAR:
00042   case mrdb::Types::LONGVARCHAR:
00043     return String;
00044   case mrdb::Types::DATE:
00045   case mrdb::Types::TIME:
00046   case mrdb::Types::TIMESTAMP:
00047     return DateTime;
00048   case mrdb::Types::BINARY:
00049   case mrdb::Types::LONGVARBINARY:
00050   case mrdb::Types::VARBINARY:
00051     return Binary;
00052   case mrdb::Types::BIT:
00053     return Bit;
00054   default:
00055     throw MineruleException(MR_ERROR_INTERNAL,
00056                             "Unexpected kind of SQL data type:" +
00057                                 Converter(type).toString());
00058   }
00059 }
00060
00061 std::string SQLUtils::quote(const std::string &str) {
00062   // We have to substitute each "'" with "''" in str (since
00063   // in sql values cannot contain single "'".
00064   std::string rest = str;
00065   std::string result = "";
00066
00067   while (rest != "") {
00068     size_t p = rest.find("'");
00069     if (p == rest.npos) {
00070       result += rest;
00071       rest = "";
00072     } else {
00073       result += rest.substr(0, p);
00074       result += "''";
00075       rest = rest.substr(p + 1, rest.length() - p - 1);
00076     }
00077   }
00078
00079   return "'" + result + "'";
00080 }
00081
00082 void SQLUtils::removeHeadBodyFromAttrName(std::string &str) {
00083   size_t pos;
00084   if ((pos = str.find("HEAD.")) != str.npos) {
00085     str.erase(pos, 5);
00086   }
00087   if ((pos = str.find("BODY.")) != str.npos) {
00088     str.erase(pos, 5);
00089   }
00090 }
00091
00092 // It seems that the following code does not work any more
00093 // A new implementation is given below, unfortunately it is
00094 // not as clean and efficient as the one below.
00095 SQLUtils::Type SQLUtils::getType(mrdb::Connection *connection,
00096                                  const std::string &tabName,
00097                                  std::string colName) {
00098   removeHeadBodyFromAttrName(colName);
00099
00100   try {
00101     mrdb::DatabaseMetaData *dbmd = connection->getMetaData();
00102
00103     return getType(dbmd->getColumnType(tabName, colName));
00104   } catch (mrdb::SQLException &e) {
00105     throw MineruleException(MR_ERROR_DATABASE_ERROR,
00106                             std::string("Cannot access to the database metadata the reason is:") +
00107                                 e.what());
00108   }
00109
00110   throw MineruleException(MR_ERROR_DATABASE_ERROR,
00111                           "Cannot access to the database metadata");
00112 }
00113
00114 // SQLUtils::Type SQLUtils::getType(mrdb::Connection *connection,
00115 //                                  const std::string &tabName,
00116 //                                  std::string colName)
00117 //                                  {
00118 //   removeHeadBodyFromAttrName(colName);
00119 //
00120 //   try {
00121 //     typedef std::pair<std::string, std::string> TabColName;
00122 //     typedef std::map<TabColName, SQLUtils::Type> TypeCatalogue;
00123 //     static TypeCatalogue typeCatalogue;
00124 //
00125 //     TypeCatalogue::const_iterator it;
00126 //     TabColName tabColName(tabName, colName);
00127 //     it = typeCatalogue.find(tabColName);
```

```
00128 //
00129 //     if (it != typeCatalogue.end())
00130 //       return it->second;
00131 //
00132 //     MRDebug("Checking mrdb type for column:" + colName + " of table " +
00133 //             tabName);
00134 //     // we are now going to load the catalogue
00135 //     mrdb::DatabaseMetaData *dbmd = connection->getMetaData();
00136 //     std::auto_ptr<mrdb::ResultSet> rs(dbmd->getColumns(
00137 //         "", // mysql does not support catalogs, what about the others?
00138 //         "", // no schema patterns
00139 //         tabName,
00140 //         "")); // this is the matter of the problem, the library bugs when
00141 //         this
00142 //               // is not ""
00143 //
00144 //     while (rs->next()) {
00145 //       typeCatalogue[TabColName(tabName, rs->getString(4))] =
00146 //           getType((mrdb::Types::SQLType)rs->getShort(5));
00147 //     }
00148 //
00149 //     // now we should find the columns in the catalogue!
00150 //     it = typeCatalogue.find(tabColName);
00151 //
00152 //     if (it != typeCatalogue.end())
00153 //       return it->second;
00154 //
00155 //   } catch (mrdb::SQLException &e) {
00156 //     throw MineruleException(MR_ERROR_DATABASE_ERROR,
00157 //                             "Cannot access to the database metadata"
00158 //                             "the reason is:" +
00159 //                                 e.what());
00160 //   }
00161 //
00162 //   throw MineruleException(MR_ERROR_DATABASE_ERROR,
00163 //                           "Cannot access to the database metadata. More "
00164 //                           "precisely, cannot determine type of column `" +
00165 //                               colName + "' of table `" + tabName + "'");
00166 // }
00167
00168 } // namespace
```

7.385 /Users/esposito/Software/minerule/src/Utils/StringUtils.cpp File Reference

```
#include "minerule/Utils/StringUtils.hpp"
```

Namespaces

- namespace minerule

7.386 StringUtils.cpp

Go to the documentation of this file.

```
00001 //   Minerule - a sql-like language for datamining
00002 //   Copyright (C) 2013 Roberto Esposito (esposito@di.unito.it)
00003 //
00004 //   This program is free software: you can redistribute it and/or modify
00005 //   it under the terms of the GNU General Public License as published by
00006 //   the Free Software Foundation, either version 3 of the License, or
00007 //   (at your option) any later version.
00008 //
00009 //   This program is distributed in the hope that it will be useful,
00010 //   but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 //   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00012 //   GNU General Public License for more details.
00013 //
00014 //   You should have received a copy of the GNU General Public License
```

```
00015 //   along with this program.  If not, see <http://www.gnu.org/licenses/>.
00016 #include "minerule/Utils/StringUtils.hpp"
00017
00018 namespace minerule {
00019         bool StringUtils::enableColors = true;
00020
00021         const std::string StringUtils::BOLD    = "\x1B[1m";
00022         const std::string StringUtils::RED             = "\x1B[31m";
00023         const std::string StringUtils::GREEN  = "\x1B[32m";
00024         const std::string StringUtils::YELLOW        = "\x1B[33m";
00025         const std::string StringUtils::BLUE    = "\x1B[34m";
00026         const std::string StringUtils::WHITE  = "\x1B[37m";
00027         const std::string StringUtils::CLOSE  = "\x1B[0m";
00028
00029         std::vector<std::string>* StringUtils::splitToLength(const std::string& str, size_t out_len) {
00030                 std::vector<std::string>* result = new std::vector<std::string>;
00031                 if(str.size() <= out_len) {
00032                         result->push_back(str);
00033                         return result;
00034                 }
00035
00036                 size_t len = str.size();
00037                 size_t cur_pos = 0;
00038                 while( cur_pos < len ) {
00039                         std::string chunk;
00040                         size_t new_pos = std::min( cur_pos + out_len, len );
00041
00042
00043                         size_t last_space = str.find_last_of(" ", new_pos);
00044                         if( last_space != std::string::npos &&  // found something AND
00045                                         last_space > cur_pos &&
          // still moving forward AND
00046                                         last_space + 10 >= new_pos &&                   // not too far
     back w.r.t. from new_pos
00047                                         !(cur_pos == len))
                                      // not at the end of the string
00048                                 new_pos = last_space+1;
00049
00050                         result->push_back(str.substr(cur_pos, new_pos-cur_pos));
00051                         cur_pos = new_pos;
00052                 }
00053
00054                 return result;
00055         }
00056
00057         std::vector<std::string> StringUtils::split(const std::string& str, const std::string& sep) {
00058                 std::vector<std::string> result;
00059                 size_t pos = 0;
00060                 size_t size = str.size();
00061                 while( pos < size ) {
00062                         size_t found_pos = str.find(sep,pos);
00063                         std::string cur_piece = str.substr(pos,found_pos);
00064                         pos+=cur_piece.size();
00065
00066                         result.push_back(cur_piece);
00067
00068                         if( pos < size ) { // if something where found, we move beyond the sep string.
00069                                 pos+=sep.size();
00070                         }
00071                 }
00072
00073                 return result;
00074         }
00075
00076         std::string StringUtils::join(const std::vector<std::string>& vec, const std::string& sep) {
00077                 std::string result;
00078                 for(std::vector<std::string>::const_iterator it=vec.begin(); it!=vec.end(); ++it) {
00079                         if(it!=vec.begin()) { result += sep; }
00080                         result += *it;
00081                 }
00082
00083                 return result;
00084         }
00085
00086 }
```