Representing and querying now-relative relational medical data

Representing and querying now-relative relational medical data

Luca Anselma[1*], Luca Piovesan[2], Bela Stantic[3], Paolo Terenziani[2]

[1]*Dipartimento di Informatica, Università di Torino, Italy*

anselma@di.unito.it

[2]*Computer Science Institute, DISIT, Università del Piemonte Orientale, Alessandria, Italy*

{luca.piovesan|paolo.terenziani}@uniupo.it

[3]*Institute for Integrated and Intelligent Systems, Griffith University, Queensland, Australia*

b.stantic@griffith.edu.au

**Corresponding author*

Abstract

Temporal information plays a crucial role in medicine. Patients' clinical records are intrinsically temporal. Thus, in Medical Informatics there is an increasing need to store, support and query temporal data (particularly in relational databases), in order, for instance, to supplement decision-support systems. In this paper, we show that current approaches to relational data have remarkable limitations in the treatment of "now-relative" data (i.e., data holding at the current time), which can severely compromise their applicability in general, and specifically in the medical context, where "now-relative" data are essential to assess the current status of the patients. We propose a theoretically grounded and application-independent relational approach to cope with now-relative data (which can be paired, e.g., with different decision support systems) overcoming such limitations. We propose a new temporal relational representation, which is the first relational model coping with the temporal indeterminacy intrinsic in now-relative data. We also propose new temporal algebraic operators to query them, supporting the distinction between possible and necessary time, and Allen's temporal relations between data. We exemplify the impact of our approach, and study the theoretical and computational properties of the new representation and algebra.

Keywords: Temporal relational data; Relational model and algebra; Now-relative data.

1. Introduction

Most clinical data (e.g., patients' clinical records) are temporal. To be interpreted, patients' symptoms, laboratory test results and, more generally, all clinical data must be paired with the time when they hold in the real world (called *valid time* in the database literature and henceforth). Though different database management systems are adopted to manage clinical data, many of them are based on the relational model theory. The research has demonstrated that, to cope with time in the relational model, the simple addition of timestamped attributes (e.g., the START and END times) is not enough, since many complex problems must be tackled. As a consequence, starting from the '80s, a whole area of re-

search, the area of (relational) Temporal Databases (TDBs henceforth) has emerged. Such an area is very active: for instance, a cumulative bibliography published in 1998 refers more than 2000 papers [1], and in the recent Encyclopaedia of Database Systems [2] more than 90 entries have been dedicated to TDB notions. Many results in the area have been consolidated, e.g., in the TSQL2 approach [3]. In the medical area, several researchers have recognized the need of TSQL2-like temporal support. For instance, Musen et al. have implemented different versions (Chronus [4] and Chronus II [5]) of a subset of TSQL2 [3], focusing on valid time. As a final recognition of the relevance of the techniques in many application areas, in the last five years some commercial systems implemented some of the achievements of the TDB research (e.g., Oracle, starting from version 11g, TERADATA DATABASE 13.10, IBM DB2 10; see http://www.cs.arizona.edu/~rts/sql3.html). An additional discussion about the importance of adopting the TDB methodologies, and of its basic principles, is reported in Appendix.1 (which can be safely skipped by experts in the TDB methodology).

However, several open problems still remain. Some of them have been recently faced by some of the co-authors of this work, with specific attention to phenomena that have a deep impact in the medical domain, such as the telic/atelic distinction [6], periodically repeated data [7], proposal vetting [8], [9] and temporal indeterminacy [10]–[12]. In this paper, we continue such a line of research, facing a challenging open problem: devising a proper TDB approach to cope with "***now-relative***" facts such as "*John is in the Intensive Care Unit (ICU henceforth) from August 10 to* ***now***" (i.e., facts starting in the past and still *valid* until the current time, as in John's example).

The treatment of now-relative facts in databases is very important, since they are usually very frequent, and are likely to be accessed more frequently [13]. Focusing our attention to *medical databases*, the relevance of now-relative facts becomes even more evident. Now-relative data constitute, for instance, a relevant portion of patients' data, and are the most likely data to be accessed (through queries to the clinical database concerning the patients' records) during the diagnostic process, or while prescribing treatments to patients. In particular, during both diagnosis and treatment prescription, valid-time now-relative data are usually the most useful ones, since they describe the *current* status of the patient. As a consequence, the possibility of properly representing and querying now-relative facts is essential in the medical domain.

Our specific interest in the treatment of temporal data in general, and now-related data in particular, is related to our involvement in the GLARE (Guideline Acquisition, Representation, and Execution) project [14], [15]. GLARE has been started from 1997 in a long-term cooperation between the Department of Computer Science of the University of Eastern Piedmont Alessandria, Italy, and the ASU San Giovanni Battista in Turin (one of the largest hospitals in Italy). Besides supporting CIG acquisition, representation, storage and execution, GLARE is characterized by the adoption of advanced Artificial Intelligence and Temporal Database formal techniques to provide advanced supports for different tasks, including reasoning about temporal constraints, model-based verification, cost-benefits analysis. However, all such support could be achieved only through a strict integration of GLARE with the patients' database, and requires an adequate treatment of the valid time of patients' data. For such a reason, GLARE interacts with a (prototypical) relational DB which is based on the TDB principles, and we have progressively extended TDB methodologies to cope with "advanced" temporal phenomena (see [6], [7], [8], [10]).

When executing a guideline, and, in particular, when taking therapeutic or diagnostic decisions, the current state of the patient has to be assessed. As a consequence, GLARE has to access the patients' database, and retrieve the patient's "now-relative" data (i.e., the data valid at the current time), and, in many cases, also their valid time (to consider when

current data started to hold). Quite surprisingly, as exemplified in the next section, and discussed in the rest of the paper, a substantial extension to current TDB techniques is needed to achieve such an apparently "easy" task.

The paper is organized as follows. In Section 2, we propose an in-depth analysis of the problem. We start from a motivating example, addressing the limitations of the approaches in the literature, and then we clarify motivations, goals and methodology of our approach. In Section 3, which is the core of the paper, we present in technical detail our approach, presenting a new temporal relational data model and algebra to cope with now-relative facts. In Section 4, we discuss the main theoretical properties of our approach (Section 4.1) and we demonstrate, through an extensive experimental evaluation, that it is computationally efficient, in that it does not add significant overhead (in query answering) with respect to "traditional" TDB approaches (Section 4.2). In Section 5, we briefly show how our approach to now-relative facts can be extended to deal also *with transaction time*. Finally, Section 6 contains conclusions and a discussion on the treatment of now-related facts in Artificial Intelligence. Appendix 1 contains a general discussion about the main motivations, goals and methodology animating the TDB area of research, and can be useful as a background knowledge for non-experts in the area. Appendix 2 proposes further comparisons between our approach and the other ones in the literature. Finally, Appendix 3 reports the definition of some algebraic operators that, for the sake of brevity, we have omitted in Section 3.

Finally, before moving on, we want to emphasize that, despite our focus on the GLARE project, the methodology we propose in this paper is totally system- and domain-independent: it is, indeed, an extension of the general TDB methodology.

2. Coping with now-relative data in TDBs: problems to be faced and goals of our approach

We first introduce a motivating example (Section 2.1), and then abstract from it, to identify the problems to be faced, and the goals of our approach (Section 2.2). We then detail our advances with respect to the state of the art (Section 2.3), and sketch the methodology we adopt to achieve them (Section 2.4).

2.1 Coping with now-relative data: an example of the problem

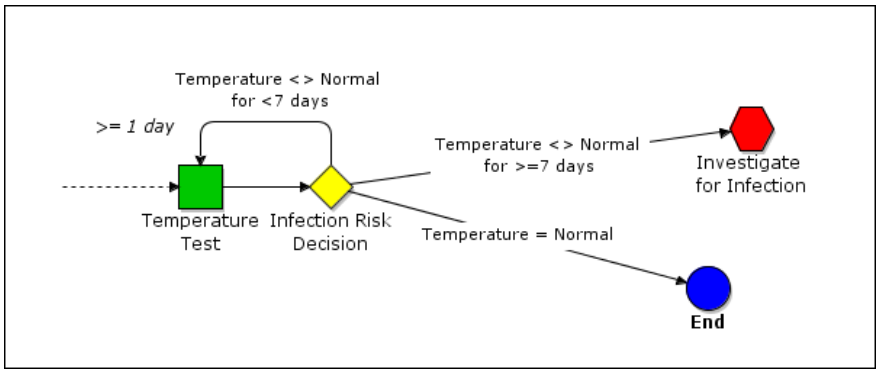
To motivate our approach, we consider a concrete example in which GLARE has to interact with the patients' database to access now-related data, and we show what are the results that we would obtain if we paired GLARE with (the implementations of) the current temporal relational database methodologies. We show that none of them would provide the proper result, which is indeed obtained only by the methodology we will illustrate in the rest of the paper.

In Fig. 1, we show a fragment of a CIG for the treatment of post-operative patients that manifested fever some days after the surgery. Fever can suggest the presence of an infection. Thus, there is the need of monitoring the patient's temperature and investigating for infections in case in which ***the fever persists for at least 7 days***. Action "Temperature Test" in the figure is a query action to check such a condition. In GLARE, query actions retrieve data from the patients' database. If data are not present in the database, the system waits until they become available.

In this example, we suppose that fact (f_1) holds:

(f_1) *Stephen manifested moderate fever starting from day May 21 to now (day May 22).*

Fig. 1. Part of a guideline for the treatment of post-surgical patients modelled in GLARE.



Fact f_1 is now-relative: it started to hold on May 21, holds on May 22 (we suppose that in such a day the fact has also been inserted in the database) and it may persist further on. In the example, we suppose that the decision "Infection Risk Decision" of the CIG is evaluated on day May 28 and that, at that time, the database still contains the tuple t_1, modelling f_1, unchanged. To take such a decision, GLARE must query the database, to retrieve the ***valid time*** of the fact "Stephen has fever", and (**Q1**) to check whether Stephen's fever *lasted at least seven days* or not.

Q1) When did Stephen have fever?

In the following, we show the result that we could obtained by pairing GLARE with the main TDB approaches in the literature coping (directly or indirectly) with now-relative facts, namely Clifford et al. [16], MIN, MAX and NULL [13], POINT [17], [18], and ORACLE 12c (as a representative of commercial systems).

CLIFFORD ET AL.: The approach by Clifford et al. [16] is a milestone, since it introduced the semantics of "now" in TDBs. Roughly speaking, it represents the end of the valid time of a now-relative tuple by the variable "now" (indeed, also now-relative variables are considered by [16]; see the discussion in Section 2.3). To answer queries, such a variable is set to the current time. In our example, at time May 28, given a database containing the fact f_1, the valid time of the fact provided by the approach by Clifford et al. [16] is [May 21, May 29).

NULL, MIN, MAX and POINT approaches: MIN, MAX and NULL approaches [13] support now-relative tuples by representing NOW as a special value for the valid-time end, i.e., the minimum chronon (MIN approach) or the maximum chronon (MAX approach) allowed by the database or the NULL value (NULL approach). In these approaches, NOW does not receive any specific support in the query language: following the seminal approach by Clifford et al. [16] at query time the special value is replaced with the current time. More recently, the POINT approach has been proposed [17], [18] and it outperforms the MAX, MIN and NULL approaches. In this approach, the end of the valid time of now-relative tuples is set to be equal to the starting time, but this is just a technical device to support Clifford et al.'s semantics in an efficient way. Given such approaches, on May 28, given a database containing the fact f_1, the valid time of the fact is again [May 21, May 29).

ORACLE 12c: Oracle database version 12c supports valid time. However, it does not explicitly cope with now-relative data; it only allows users to set the end and start valid times to NULL in order to represent facts valid at all time values [19]. Such a feature could be used to model now-relative facts. In such an approach (see Table 1) the fact "Stephen has fever" is valid forever (starting from May 21).

In Table 1, we summarize, for each approach, how it represents the temporal part of the now-relative fact (f_1) (the representation of our approach will be widely discussed in Section 3.1 of this paper), the valid time of such a fact at time RT = May 28, and the answer to a query (Q1) asking who had at least seven days of fever.

TABLE 1. Comparison of different approaches on a concrete example

Approach	Representation	Valid Time of Fever at RT = May 28	Result of query Q1 (RT=May 28)
Oracle 12c	<… | May 21, NULL>	[May 21, c_{max})	{<Stephen, Temperature, Moderate>}
NULL	<… | May 21, NULL>	[May 21, May 29)	{<Stephen, Temperature, Moderate>}
MAX	<… | May 21, c_{max}>	[May 21, May 29)	{<Stephen, Temperature, Moderate>}
MIN	<… | May 21, c_{min}>	[May 21, May 29)	{<Stephen, Temperature, Moderate>}
POINT	<… | May 21, May 21>	[May 21, May 29)	{<Stephen, Temperature, Moderate>}
Clifford	<… | May 21, now>	[May 21, May 29)	{<Stephen, Temperature, Moderate>}
Our approach	<… | May 21, May 23, c_{max}, $-\infty$, $+\infty$>	POSS: [May 21, c_{max}) CERT: [May 21, May 23)	{ }

As Table 1 clearly shows, in Oracle 12c the fact "Stephen has fever" is valid forever (starting from May 21). The approaches NULL, MAX and MIN and POINT (as well as the approach by Clifford et al.) give the same result: at time May 28, given a database containing the fact f_1, the valid time of the fact is [May 21, May 29). In all such cases, since the valid time of the fever is at least seven days, the tuple <Stephen, Temperature, Moderate> is given as output by the TDB. As a consequence of the TDB answer, GLARE would suggest user-physicians to investigate for an infection. However, it is fundamental to notice that such a suggestion is ***not supported*** by the data available in the database.

Indeed, given the fact that (f_1) has been asserted (and inserted in the database) on May 22, we can be certain that, on May 21 and May 22, Stephen had a moderate fever. But in no way the fact (f_1) can grant that the moderate fever persisted in the next days (and, in particular, until May 28). Stephen's conditions could have changed in the meanwhile, but the database may not have been updated. For instance, it may be the case that Stephen's fever had ended on May 26, but this change had not been reported yet in the database. As a consequence, if GLARE were paired with any of the above approaches (i.e., Oracle 12c, NULL, MIN, MAX, POINT and Clifford's one), the infection investigation might be erroneously suggested (and executed on the patient). On the other hand, as we will show in the rest of the paper, with our methodology, the TDB would provide GLARE with the correct answer: given the information in the database, it is not certain only that Stephen had fever from May 21 to May 22 (CERT valid time), but it is also possible that the fever persisted further on (POSS valid time). Thus, the certain duration of fever is just two days, while it may have persisted further on. GLARE may interpret such an answer as a missing information, and it might ask for an up-to-date value of the data before suggesting a decision, thus providing the correct recommendation.

The above example clearly shows that our approach outperforms the other approaches in the literature in the treatment of now-relative data[1] (a more detailed comparison is provided in Appendix 2, after the full presentation of our approach), and exemplifies the impact of such a new feature on clinical decision making. In the next subsection, we gen-

[1] A digression is worth here, to further emphasize the advantage of adopting a TDB approach like the one we propose. Several current hospital databases, lacking TDB support, simply store the starting time of facts (even of *durative* facts, like *fever*), relying on physicians' background knowledge and inferential capabilities for the evaluation of the persistence of such facts (i.e., of their *valid time*). Such an approach could not work in the context proposed by our example, in which the valid time of a fact (and, in particular, its duration) must be considered, and such a value must be automatically provided by the TDB to the GLARE decision support system (with no possibility of human intervention or inference). This remark substantiates the general claim reported in Appendix 1 that TDB approaches (and our approach in particular) provide users with a better and higher-level support to deal with temporal data (and, in particular, with *durative* temporal data: consider *Principle 1* in Appendix 1), relieving users, e.g., from the burden of inferring the ending time and\or duration of durative facts.

eralize from this example, pointing out the problem underlying the treatment of "now" done by the approaches in the literature, and highlighting the limitation of such approaches, and the goals of our approach.

2.2 Coping with now-relative data: the core problem

In this section, we abstract from the specific example above, trying to show why the TDB approaches in the literature fail managing the valid time of now-relative facts, and the treatment of now-relative data imposes additional challenges to "traditional" TDB techniques (consider, e.g., the widely known TSQL2 approach [3]). In "traditional" TDBs, the fact that

Example 1. $Fact_1$ holds from vt_s to vt_e (where $vt_s \leq vt_e$)

is usually represented through the tuple $<Fact_1|vt_s,vt_e>$, which associates the tuple describing $Fact_1$ with the starting point vt_s of its valid time (usually represented by the temporal attribute VT_s) and the ending point vt_e of its valid time (usually represented by the temporal attribute VT_e)[2]. The meaning of such a representation is that the fact $Fact_1$ holds at the time units $\{vt_s, vt_s+1, \ldots, vt_e\}$. Thus, the TDB can correctly answer any query about when $Fact_1$ holds.

However, let us consider a now-relative fact, i.e., a fact that is asserted to hold "now". For instance, in Example 2 we suppose that $Fact_1$ has started to hold at time vt_s, and has been asserted at time vt_a (e.g., at time vt_a a user has said: "$Fact_1$ has started to hold at time vt_s, and still holds ***now***").

Example 2. $Fact_1$ holds from vt_s to "now" (asserted at time vt_a; $vt_s \leq vt_a$).

The representation of such a fact in a TDB is difficult, since "now" looks like a variable, and standard relational DBMSs do not support variables. Let us suppose, for the sake of brevity, that a special symbol (say "*") is used to represent "now".[3]

We can thus associate a tuple representing $Fact_1$ with the starting point $VT_s=vt_s$ and the ending point VT_e = "*" (i.e., the tuple $<Fact_1 \mid vt_s,"*">$). However, the meaning of such a representation is now quite complex: the fact $Fact_1$ certainly holds at the times $\{vt_s, vt_s+1, \ldots, vt_a\}$, and *may* continue to hold at later times[4]. Thus, there is some degree of ***temporal indeterminacy***: we do not know how long $Fact_1$ will persist after vt_a. Such a degree of indeterminacy makes query answering more complex and subtle. Consider, e.g., a time t_Q later than vt_a, and suppose that one asks the query Q2 (to the TDB containing the tuple $<Fact_1, vt_s,"*">$):

Q2) Does $Fact_1$ hold at time t_Q, and what is the overall valid time of $Fact_1$ (at time t_Q)?

All the approaches coping with now-relative facts in the TDB literature (with the only exception of Clifford et al. [16], discussed later) would give the same answer:

A2) $Fact_1$ holds at time t_Q, and its valid time is the time interval $[vt_s, t_Q]$.

This result is motivated by the fact that such approaches assume that, as long as a now-relative fact is in the database, it is true also in the outside world. However, though operationally effective, such an assumption in general is ***not logically correct***. The now-relative fact may have stopped to hold in the outside world, but the database might not have been up-

---

[2] This representation is motivated by the basic ***Principle (1)*** discussed in Appendix 1.

[3] Indeed, as discussed in Section 2.1 above, and illustrated in Table 1, several different approaches have been proposed, including the representation of "now" with the NULL value or with the minimum or maximum possible date supported by the DBMS [13], or imposing that the ending time is equal to the start time (POINT approach [17], [18]) or through a variable, in the semantic approach by Clifford et al [16].

[4] Notice that now-relative facts may also be facts that start to hold at the assertion time (consider, e.g., the assertion: "$Fact_1$ starts to hold now", asserted at a given time vt_a). This is simply a special case of the general example in Example 2, in which vt_s is equal to vt_a. In such a case, the intended meaning is that $Fact_1$ certainly holds at time vt_a, and may continue to hold at later times.

dated yet to record such a change. Consider again, e.g., the example in Section 2.1 above: from the fact that Stephen had moderate fever from May 21 to now, asserted (and inserted into the database) on May 22, we cannot be certain that Stephen has still moderate fever, e.g., on May 28. It might have happened that fever has ceased in the meanwhile, and no one has soon updated the tuple in the database. To better explain this phenomenon, the notion of ***latency*** should be explicitly introduced:

Definition (Latency). Latency is the span of time that occurs between the time (say t_{stop}) when a now-relative fact stops to hold in the modelled world, and the time when the corresponding TDB tuple is updated (to put the t_{stop} value at the place of "*" as the value of the temporal attribute VT_e)[5].

Saying that a now-relative tuple <$Fact_1$ | vt_s,"*"> has *latency* equal to *zero* means that it is *guaranteed* that, as soon as $Fact_1$ will stop to hold in the real world, its corresponding tuple in the TDB will be updated (it does not matter by whom this update will be performed; e.g., it may be done by the system administrator, or by the user who has inserted the tuple). Since with zero latency we know that, as soon as $Fact_1$ will become false in the real world, the TDB tuple will be modified, from the fact that at time t_Q the tuple <$Fact_1$ | vt_s,"*"> is still in the TDB we can *correctly* infer that at time t_Q $Fact_1$ does still hold in the real world. Thus, given the fact in Example 2, A2 ***may be*** a correct answer to the query Q2. Notice, however, that ***the answer A2 is correct just in case zero latency can be assumed*** (see the discussion below about unknown latency).

This means that, ***although implicitly***, ***all current TDB approaches*** (e.g., the NULL, MIN, MAX and POINT approaches [13], [17]) ***assume that the latency is exactly zero***. Indeed, they assume that, as soon as a change happens in the real world, now-relative tuples are updated. Of course, such an assumption is unrealistic in the medical domain: what clinical record can assume to be instantaneously updated with respect to the (data concerning the) state of the patient? No one can grant, for example, that each (even small) change in the blood pressure of a patient is instantaneously recorded into her/his clinical record (unless the patient is continuously monitored).

The only exception in literature is the *semantic* approach by Clifford et al. [16], who extend the zero-latency approaches by assuming that latency is *exact* and *known*. Therefore, Clifford et al. cover not only the case in which changes to now-relative facts are recorded as soon as they happen, but also proactive and retroactive updates. However, for each specific piece of now-relative data, Clifford et al. assume that it is exactly specified what its latency is. For instance, considering the blood pressure example, one may know that each change of value is recorded ***exactly*** one hour after it occurs in the real world (notably, also proactive updates are managed by Clifford et al. – e.g., the salary of a physician is updated exactly one month *before* the actual change occurs). However, also this assumption is not realistic in many domains, and, in particular, in the ***medical domain***, where, in general,

[5] Until Section 5 in this paper we do not deal with ***transaction time*** (for the sake of brevity, but also because it is not essential in many medical applications). However, we think that it is important to clearly distinguish between latency and transaction time. Transaction time is the database time, i.e., the time when tuples are inserted/deleted/updated in the database. For instance, if the user has inserted a tuple representing "John is in ICU from day 11 to NOW" into the TDB on day 16, we have that the transaction tuple of such a tuple is 16 (more precisely, 16 is the starting point of its transaction time). The ***latency*** of such a tuple may be totally ***independent*** of such a transaction time. For instance, it may be zero (if someone guarantees that s/he will update the now-relative fact in the database as soon as John will be dismissed from ICU), or an exact or indeterminate value, or even unknown, in case there is no guarantee at all about when the update of the "now" value will be performed (with respect to the time when John will be dismissed).

(i) one can just specify a minimum and maximum delay between changes of now-relative facts in the world and their recording in the database (i.e., a range for latency, between a minimum and a maximum value),

(ii) or, in the most general case, one cannot make any assumption on when changes regarding now-relative facts will be recorded in the database (i.e., latency is totally unknown)[6].

Of course, the different assumptions on latency (latency equal to zero, exact known latency, range of values for latency, or unknown latency) have a deep impact on the semantics of now-relative facts, and, consequently, on the database answers to user queries concerning now-relative facts.

Let us consider, in particular, case (ii) above, where one has no guarantee about latency, i.e., about the span of time between the time when $Fact_1$ will stop in the real world and the time when the TDB now-relative tuple will be updated accordingly. In such a case, we say that the latency is ***unknown***. If latency is unknown, the fact that the tuple <$Fact_1$ | vt_s, "*"> is still in the TDB at time t_Q ***does not add any evidence about the persistence of $Fact_1$ at that time***. Maybe $Fact_1$ has finished to hold at time vt_a+1 (or at any later time vt_a+k, where $vt_a+k \leq t_Q$), but *no one has updated the tuple* <$Fact_1$ | vt_s,"*"> in the TDB until time t_Q (included). We can just be *certain* that $Fact_1$ holds from vt_s to vt_a, while it is *possible* that $Fact_1$ persists later on. Thus, in case the latency of the tuple representing Example 2 is unknown, the answer to query **Q2** would be **A2'** below.

A2') $Fact_1$ *may* hold at time t_Q, but it is ***not** certain*. Its *certain* valid time is [vt_s, vt_a] while it is possible that $Fact_1$ holds also after vt_a.

In general, we believe that in most real applications, and in particular in most medical applications, it is unrealistic to expect to have *any guarantee about latency*. Therefore, we think that the case of ***unknown latency*** is the most important. Notably, in this paper, we propose the ***first*** TDB approach which can deal with such an extremely important case. A second important case is the one in which, though it is not possible to grant that the update of now-relative facts will be performed soon (zero latency), it is possible to grant that such an update will be executed within a pre-specified range of time (e.g., within two days from the change). This is realistic, e.g., in cases in which specific data explicitly impose such a constraint, as in Example 3.

Example 3. Nonsteroidal anti-inflammatory drugs (NSAIDs) increase the risk of developing gastrointestinal disorders. For this reason, the therapies requiring the administrations of NSAIDs usually prescribe also the co-administration of gastroprotective drugs, which must be discontinued within a few days from the end of the NSAID therapy. For instance, indomethacin is a NSAID drug administered to treat musculoskeletal disorders and arthritis, and it is usually associated with ranitidine (a gastroprotective drug). To guarantee that ranitidine is discontinued in an adequate span of time after indomethacin has been discontinued, the latency of update of a now-relative fact modelling the administration of indomethacin must be at most three days, starting from the day in which the indomethacin treatment is discontinued.

In such a case, that we term "***interval latency***", latency is ***partially indeterminate***, ranging between a ***minimum*** and a ***maximum value*** (e.g., from zero to three days in the example above). Indeed, for the sake of generality, and to show that our approach can easily treat also the cases treated by the TDB approaches in the literature, we also consider zero

[6] Indeed, Jensen & Snodgrass [20] have explicitly defined ***general*** temporal relations as relations where "*there are no restrictions on the interrelations of, or correlations between, the transaction and the valid timestamps of an item*". This means that, in such relations, no assumption can be made on the value of latency (i.e., on the time when changes in the real world are recorded into the database; indeed, the *orthogonality* of valid time and transaction time is one of the basic principles of temporal databases – see, e.g., TSQL2 [3] and BCDM [21]).

latency (as the MIN, MAX, NULL, and Clifford et al.'s approaches), and exact latency (as Clifford et al.). Thus, in the (unlikely) situation in which we have some guarantee about when now-relative tuples are modified, we work as the other approaches in the literature (which, on the other hand, cannot correctly manage unknown and interval latency).
In summary, our approach copes with all the possible cases of latency:

- ***zero latency,***
- ***exact latency,***
- ***interval latency***,
- ***unknown latency***.

In such a way, our approach supports a complete treatment of all kinds of now-relative facts, significantly advancing the state of the art, and providing a sound ground for medical applications (such as, e.g., GLARE), and in general, for queries regarding now-relative data.

Our approach emerges from all the other TDB approaches to now-relative facts also because we are the only ones coping with ***bounds*** on *now-relative data*. While in general the persistence of now-relative facts may be totally unknown, in the medical domain it is quite frequent to know a maximum bound for their persistency. There are many different clinical situations in which a bound for the future persistence of a clinical data can be assumed, and it has to be managed. One example regards the results of (cyclically repeated) laboratory tests. Although the valid time of a single laboratory test result is punctual (i.e., the result is valid only in the time in which the test takes place), a sequence of tests with normal results produces a belief of "stability" of the patient's condition whose duration depends (obviously) on the specific parameters under evaluation and on the length of the sequence of normal results. This fact can be represented by the introduction of an upper bound (in the future) for the time of validity of the stability of patients' conditions, as in Example 4.

Example 4. Mary is treated with warfarin. Mary's blood coagulation is "stable" from April 4 (the day in which the last INR test has been performed) to now (asserted on April 10). Stability is assumed at most to May 4.

Comment to Example 4. Warfarin is an anticoagulant drug usually administered for the treatment of venous thrombosis. However, the administration of anticoagulants such as warfarin affects the blood coagulation status (increasing the risk of bleedings). For this reason, after the beginning of the warfarin treatment, the blood coagulation status of a patient cannot be considered "stable" and needs to be frequently monitored. The most significant measure to monitor the blood coagulation status is the INR (International Normalized Ratio). Several guidelines (among them [22]) suggest to start evaluating the INR every two days immediately after the beginning of the treatment. In case the tests produce abnormal INR values, the anticoagulant dosage can be changed and further tests must be performed again every two (or less) days. On the other hand, in case the INR values result normal at each evaluation, the time between tests can be progressively incremented[7] to a week, a month and finally to 12 weeks. This is clearly justified by the "belief" of stability given by the past stable results: each time a normal value is observed, it increases the belief of stability in the future, and the next test can be performed later in time. In our example, we consider a patient treated with warfarin, Mary, whose previous INR tests have been all normal. Supposing that Mary's last examination has been performed on April 4, the physician assumes that Mary's blood coagulation status may be "stable" for a month (i.e., the upper bound for the "stability" assumption is May 4).

[7] It is worth stressing that the rate of increment does not depend only on the results of the past tests. In general, it depends also on the patient's medical status and it is evaluated case by case by a physician.

Notice that Example 4 reflects a quite general situation in the medical context, regarding many different patient's parameters (consider, for instance, glycaemia –for the treatment of diabetic patients– or blood pressure –for hypertension).

2.3 Overcoming the limitations of current TDB approaches to now-relative data

Despite this importance, the problem of dealing with "now" has attracted only a limited attention within the temporal database community, and even less attention in the field of commercial databases. Already SQL-92 introduced the construct CURRENT_TIMESTAMP to model "now". However, it could only be used in the queries, while one could not store it in a SQL column as a value for the ending time of a now-relative tuple. The user was forced to store a specific time, which is clearly problematic and prone to errors (see [16]). To solve this problem, many subsequent approaches have introduced a variable, such as "now" (other symbols have been used, e.g., "–", "∞", "@" and "until-changed" [23]) as its ending time, leading to "*variable*" databases [23]. However, variable databases require a significant departure from the "consensus" relational model since they are not supported by the domain of SQL1999 values [24]. More recent research approaches, already considered in the example in Section 2.1, have identified four different ways to implement the value "now" in a standard relational database, using (i) NULL, (ii) the smallest timestamp (MIN approach), (iii) the largest timestamp (MAX approach), or (iv) degenerate zero-point intervals (POINT approach) (see, e.g., [17]). Recent commercial approaches provide temporal support to valid and transaction time, mostly based on the TSQL2 "consensus" research result [3] (in particular, TSQL2 provides, e.g., the theoretical basis of Oracle, starting from version 11g, TERADATA DATABASE 13.10, IBM DB2 10; see http://www.cs.arizona.edu/~rts/sql3.html), and some of them (like Oracle 12c and SQL Server 2016) also support the NULL or the MAX values to cope with valid-time tuples. However, such approaches still have three main limitations:

(1) they propose models to store data, but no temporal algebra to query them (except the POINT approach [18]),

(2) they implicitly assume zero-latency, or, at most, exact and known latency [16], and

(3) they cannot cope with bounds on the future persistence of now-relative facts.

All the above limitations are very relevant in the medical field. The impact of issue (1) is clear. As discussed above, in medical applications, now-relative facts need to be often selected and retrieved, e.g., for diagnostic or therapeutic purposes, through queries. However, without a clear and explicit relational algebra, there is no formal specification on how now-relative data are managed by query answering. The impact of limitation (2) is even more important, and has been widely discussed in Sections 2.1 and 2.2 above. Also, in at the end of Section 2.2, we have discussed and exemplified the relevance of "bounds" on now-relative data.

Before moving on, we end the comparisons with the related literature by considering our previous work in the treatment of ***now-relative data***. In the proceedings of AIME'15, we have published a short paper addressing the importance of coping with *unknown latency*, and proposing a new data model (which is properly enclosed in the one we present in this paper) and the definition of Cartesian product operating on it [25]. In [26] we have then extended such a preliminary work to cope also with *zero* and *exact latency*, and to consider also *transaction time*. We have proposed an extended data model (which is properly included in the one we present in this paper), and an extension of Codd's algebraic operators to cope with it. In this paper, we further extend the approach in [26] to many respects:

(1) We add the possibility of coping also with "***interval latency***", which is important in several medical contexts (see the discussion in Section 2.2). To achieve such an extension, we had to extend the definition of the data model in [26] and to modify the definition of Codd's algebraic operators in [26].

(2) We have added the ***temporal selection operators*** (Section 3.3 and Appendix 3), which are very important for medical applications, and were missing in [26].

(3) Finally, in [26] the approach was purely theoretical and domain-independent. We have almost completely changed it, to demonstrate the impact and application of our approach in the context of Medical Informatics, and to make our paper suitable for readers in the Medical Informatics area.

(In summary, Sections 3.1, 3.2 and 4.1 are an extension of the Sections 3.1 and 4.2 in [26] to consider interval latency, while the rest of the paper is new).

3. Materials and methods

To devise our approach, we follow a methodology that is commonly used in the TDB area, and that we followed in our previous works in the area (to cope with the telic/atelic distinction [6], periodically repeated data [7], proposal vetting [8] and temporal indeterminacy [10]): we propose (i) a new data model to represent now-relative data, and (ii) a new relational algebra to query it. Our goal is not just to conceive an *ad-hoc* solution, but to propose a *theoretically sound* framework which is *implementable* on top of current technologies (in particular, we consider TSQL2 [3], which provides the theoretical basis of current commercial temporal relational DBMS; see [27]) and *interoperable* with current frameworks, to grant that previous data can be maintained in the new framework. To guarantee the achievement of such objectives, we prove that our data model is a *consistent extension* of TSQL2's one (which, in turn, is a consistent extension of SQL), and that our algebra is *reducible* to the standard relational one. Finally, we also show through experimental evaluations that, despite of its expressiveness and generality, our approach does not add any significant overhead to "standard" TDB approaches.

3.1 Data Model

We first propose a compact 1NF representation for now-relative tuples, considering *valid time* (*transaction time* [3] will be briefly considered in Section 5). As regards the definition of the temporal domain, the large majority of TDB approaches simply notice that, although time is continuous, it is commonly represented in a discrete way (by splitting the timeline on the basis of a basic granularity). Additionally, although time may be thought to be infinite in the future, a bound can be imposed in practical real-world applications. As a consequence, the temporal ontology commonly used in TDBs [3], [21] assumes that time is discrete, linearly ordered and isomorphic to a subset of the integers. We adhere to such a general assumption, though our approach is largely independent of it. For simplicity, we also assume a single granularity (*days* in the examples). We denote by c_{max} the maximum chronon in the temporal domain.

Definition: Chronon. The chronon is the basic time unit. The chronon domain T^C, also called timeline, is the totally ordered set of chronons $\{\ldots, c_i, \ldots, c_j, \ldots\}$, with $c_i < c_j$ as $i < j$. ■

Possibly now-relative tuples (*pn-tuples*) and relations (*pn-relations*) are represented as below.

Definition: pn-tuple and pn-relation. Given a schema $(A_1, \ldots, A_n)$ where each A_i represents a non-temporal attribute on the domain D_i, a pn-relation r^{pn} is an instance of the schema $(A_1,\ldots,A_n|VT_s,VT_a,VT_e,\Delta_m,\Delta_M)$ defined over the domain $D_1\times \ldots \times D_n \times T^C \times T^C \times T^C \times (\mathbb{Z} \cup \{-\infty,'NR'\}) \times (\mathbb{Z} \cup \{+\infty,'NR'\})$, where T^C is the domain of *chronons*, VT_s represents the beginning of valid time, VT_a the assertion time (i.e., the time when the fact (tuple) is asserted by the user), VT_e the future bound for 'now' (the value c_{max} is used if no bound must be modelled), and Δ_m and Δ_M the minimum and maximum latency respectively. Δ_m and Δ_M assume integer values, as well as the values $-\infty$ for Δ_m and $+\infty$ for Δ_M, and 'NR' (not relevant). For each instance of the schema, $VT_s \leq VT_a \leq VT_e$ and $\Delta_m \leq \Delta_M$. Each tuple $x = (a_1, \ldots, a_n | vt_s, vt_a, vt_e, d_m, d_M) \in r^{pn}$ is termed a pn-tuple. In particular, (i) d_m=d_M when the latency is exact, (ii)

$d_m = -\infty$ and $d_M = +\infty$ represent the case in which the latency is totally unknown, (iii) $d_m = d_M = NR$ when the tuple is not now-relative.

It is worth noticing that, considering other problems, also Johnston and Weis [28] have pointed out that, besides *valid* (called *effective*) time and *transaction* time, also ***assertion*** time should be considered. *Transaction* (insertion) time may be different from the *assertion* time (e.g., the tuple may be asserted, and then inserted later in the database). Also, *assertion time* is different from Clifford et al.'s *reference time*. For instance, the data in Example 3, which is valid (*valid* time) from April 4 to NOW, can be asserted (*assertion* time) on April 10, physically inserted into the TDB (*transaction* time) on April 12 and the database can then be inspected considering, e.g., April 25 as ***reference*** time (see below).

Here we introduce some examples and then illustrate how they can be represented in a pn-relation as pn-tuples.

Example 5. John suffered from severe headache from August 19 to August 24. In addition, he also manifested high fever from August 20 to August 24 and since August 20 he still manifests neck stiffness. He has been visited on August 25 when such symptoms have been reported.

Example 6. Tom is suffering from abdominal pain since July 3 and suffered from severe nausea from August 1 to August 3. On August 6 he has been visited and such symptoms have been reported. On August 7 Tom did an abdominal x-ray and on August 9 he did an endoscopy; we assume that the results of such exams are valid for one month.

Example 7. Bill is manifesting fatigue since August 1, cough since August 10 and chest pain since August 15. During the medical examination, on August 25, Bill still exhibited the symptoms, and asserted them to the physician. In addition, Bill is suffering from high fever since August 22 and he has been hospitalized on the same day of the medical examination for a suspected pneumonia. He is recovered in ICU, and his fever is constantly monitored (and fever's values automatically recorded in the hospital database).

Example 8. Stephen underwent a surgical operation and then he was discharged. Some days later (on May 21) he manifested moderate fever. To monitor the possibility of an infection, the patient reports to the physician a change of his conditions within one day.

We assume that data of Examples 5, 6, 7, 8 are stored in a TDB following the approach described in this paper. Such examples, at the granularity of days, are expressed in our model as shown in the relations PSYMPTOM and PEVIDENCE in Fig. 2 and Fig. 3.

In our model, tuples that are not now-relative can still be represented, using the convention introduced in Property 1 in Section 4.1. In our approach, each now-relative tuple has its own latency, stating the "guarantee" about when such tuples will be updated in response of changes in the modeled world (notice that, in case of non-now-relative tuples, latency does not apply, and assumes the vale "NR" – for "Not Relevant"). Notice that, in the medical context, latency may be due both to the delay between the time when changes in the patient's status happen and the time when the patient reports them (e.g., in the case of outpatients), and, mainly, to the delay (due to physicians and/or hospital operators) in the actual update of the now-relative tuples in the database after such a change is reported. As discussed in Section 2, in general, in the medical context, there is no guarantee on when now-relative tuples will be updated (in response to a change in the world): thus, "unknown" latency is the general case. In the following, we assume "unknown" latency for *neck stiffness* in Example 5, for all the symptoms and examinations in Example 6 (except *nausea*, which is not now-relative, so that its latency is "NR"), and for all the symptoms in Example 7. In the case of Example 7, we assume that

the patient is in ICU and the patient's temperature is constantly monitored and the database values automatically updated, so that the latency of tuple 4 in the relation PEVIDENCE is assumed to be zero. Finally, we assume that, in the case of Example 8, we have the guarantee that the patient's data are recorded within two days after the time when the patient reports them. Thus, in this case (since reports may be a day after the change), the latency of the tuple is a range of possible values: [-3,0].

Fig. 2 Pn-relation PSYMPTOM.

ROWID	Patient	Symptom	VT_s	VT_a	VT_e	Δ_m	Δ_M
1	John	severe headache	Aug 19	Aug 25	Aug 25	NR	NR
2	John	neck stiffness	Aug 20	Aug 26	c_{max}	$-\infty$	$+\infty$
3	Tom	abdominal pain	Jul 3	Aug 7	c_{max}	$-\infty$	$+\infty$
4	Tom	nausea	Aug 1	Aug 4	Aug 4	NR	NR
5	Bill	chest pain	Aug 15	Aug 26	c_{max}	$-\infty$	$+\infty$
6	Bill	fatigue	Aug 1	Aug 26	c_{max}	$-\infty$	$+\infty$
7	Bill	cough	Aug 10	Aug 26	c_{max}	$-\infty$	$+\infty$

Fig. 3 Pn-relation PEVIDENCE.

ROWID	Patient	Examination	Outcome	VT_s	VT_a	VT_e	Δ_m	Δ_M
1	Stephen	temperature	Moderate	May 21	May 22	c_{max}	-3	0
2	Tom	abdominal x-ray	Negative	Aug 7	Aug 8	Sept 8	$-\infty$	$+\infty$
3	Tom	endoscopy	Gastritis	Aug 9	Aug 10	Sept 10	$-\infty$	$+\infty$
4	Bill	temperature	High	Aug 22	Aug 26	c_{max}	0	0
5	John	temperature	High	Aug 20	Aug 25	Aug 25	NR	NR
6	Frank	INR	Low	Jun 15	Jun 16	Jun 16	NR	NR

Data are shown in the relations PSYMPTOM and PEVIDENCE in Fig. 2 and Fig. 3. For instance, intuitively speaking, row 1 of Fig. 2 represents the fact "John suffered from severe headache from August 19 to August 24" of Example 5. The fact "Tom is suffering from abdominal pain since July 3", asserted by the physician during the visit on August 6, is represented in row 3 of Fig. 2. The latency of updates of this tuple is unknown (modeled with $\Delta_m = -\infty$ and $\Delta_M = +\infty$). On the other hand, the fact of Example 6 "Bill is suffering from high fever since August 22" (asserted on August 25) is modeled in row 4 of relation PEVIDENCE shown in Fig. 3. In this example, we assume that the temperature is constantly monitored and its value is automatically updated in the database, so that latency is zero (modeled with $\Delta_m = 0$ and $\Delta_M = 0$). Finally, the fact "Stephen manifested moderate fever on May 21" of Example 8 is represented in row 1 of relation PEVIDENCE. In this case, given the assumptions discussed above, latency is $\Delta_m = -3$ and $\Delta_M = 0$ days.

Conventions. We choose to adopt intervals $[VT_s,VT_a)$ and $[VT_a,VT_e)$ closed to the left and open to the right.

Notice that pn-relations can include heterogeneous types of tuples: any tuple, independently of the others, may be now-relative or not.

Notice also that some tuples may have future bounds (for VT_e). For instance, if we assume that the result of the abdominal x-ray examination of Tom has a validity of one month, and the examination has been performed on August 7, we can represent the end of its valid time as VT_e = September 8, as in row 2 of relation PEVIDENCE.

As a further example, in the first row of Fig. 3 we have shown our representation of fact (f_1): *Stephen manifested moderate fever starting from day May 21 to now (day May 22).*

Our representation models now-relative facts in a compact and implicit way. To provide the *semantics* of now-relative tuples, following Clifford et al. [16] we adopt a ***reference time*** (RT), «*to represent the relationship between a temporal database and the "real world" time at which it is viewed*» [16, p. 180]. Notice that RT is different from the *transaction* time (see [16] and above).

Definition: semantics of a pn-tuple. Given a reference time *t* and a pn-relation r^{pn}, which is an instance of the schema ($A_1,\ldots,A_n|VT_s,VT_a,VT_e,\Delta_m,\Delta_M$), each pn-tuple $x = (a_1, \ldots, a_n \,|vt_s, vt_a, vt_e, d_m, d_M) \in r^{pn}$ has the following semantics: at reference time *t*, *x* certainly holds between vt_s and $\min(\max(t + d_m, vt_a - 1), vt_e - 1)$ and possibly holds between $\max(t + d_m + 1, vt_a)$ and $vt_e - 1$.

For instance, at the *reference time* August 25, we can represent as shown in Table 2 the semantics of the facts regarding Tom in Example 5 reported in the relations in Fig. 2 and Fig. 3. The semantics is represented as a disjunction of time intervals, to show the alternative ending times of now-relative facts. For instance, Tom has had abdominal pain certainly from July 3 to August 6, but his symptom can persist until c_{max}, as modeled by the other intervals of the disjunction.

TABLE 2. Semantics of the facts regarding Tom in Example 5

Symptom/Examination	**Valid Time**
abdominal pain	[Jul 3, Aug 7) *or* [Jul 3, Aug 8) *or* … *or* [Jul 3, c_{max})
nausea	[Aug 1, Aug 4)
abdominal x-ray	[Aug 7, Aug 8) or [Aug 7, Aug 9) *or* … *or* [Aug 7, Sept 8)
endoscopy	[Aug 9, Aug 10) or [Aug 9, Aug 11) *or* … *or* [Aug 9, Sept 10)

3.2 Relational algebra: extending Codd's operator

Our relational algebra operators directly operate on the implicit 1NF representation above and are consistent with the underlying semantics. In the following, we provide our temporal extension to Codd's relational algebra operators [29]. As, e.g., in BCDM [21] and TSQL2 [3] to grant *reducibility* [3], temporal extensions operate as Codd's operators on the non-temporal attributes. Additionally, as, e.g., in TSQL2 non-temporal selection, projection and union do not directly operate on valid time. On the other hand, as in BCDM and TSQL2, our Cartesian product manages the non-temporal attributes $A_1,\ldots,A_n$, $B_1,\ldots,B_n$ in the standard (i.e., non temporal) way and evaluates the intersection of the temporal parts of the paired tuples. The function *interpr* basically makes explicit the semantics of the data model. As in Das and Musen's approach to indeterminate time [4], we propose two algebraic operators for difference: the certain difference $-_t^{\text{pn-cert}}$, returning only certain times, and the possible difference $-_t^{\text{pn-poss}}$, returning all possible times. The certain difference uses the *interpr* function to determine the end of the certain part of the valid time. The definition of difference requires specific attention. As already pointed out by in BCDM, for each tuple $t \in r^{pn}$, the times of all the tuples $t_1, \ldots, t_k \in s^{pn}$ that are *value-equivalent* to it (i.e., equal as regards the values of the non-temporal attributes) must be subtracted from the time of t. In the definitions in Fig. 4 the uniqueness operator ∃! identifies *all* and *only* the tuples $x_1'', \ldots, x_k'' \in s^{pn}$ value-equivalent to x'. The operator –* repeatedly applies the binary difference operator to remove elements of the second set from each element in the first set. When evaluating certain difference, we consider only the "certain" valid time for the minuend (so that the end of its valid time is *interpr(x'[VT_a], x'[VT_e], x'[Δ_m],t)*) and "possible" valid times for subtrahends (so that the end of their valid time is VT_e). When evaluating possible difference, only

certain times (determined through the *interpr()* function) of the subtrahends are deleted from the possible time of the minuend. Each element of $-^*$ has the form $<vt_s,vt_a,vt_e,d_m,d_M>$, where $vt_e=vt_a$. Binary difference between two elements $<vt^1_s,vt^1_a,vt^1_e,d^1_m,d^1_M>$ and $<vt^2_s,vt^2_a,vt^2_e,d^2_m,d^2_M>$ is computed as follows: the standard difference between the two time intervals $[vt^1_s,vt^1_e)$ and $[vt^2_s,vt^2_e)$ is performed. Zero, one or two intervals are provided as output. Let VT_set be the set of such intervals; for each interval $[vt_s,vt_e)\in$VT_set, $\{<vt_s,vt_e,vt_e,NR,NR>\}$ is added to the set of results.

Notation. In Fig. 4 and in the following, we assume that t[X] represents the value of the attribute X in the tuple t.

Fig. 4. Definition of relational algebra operators.

Definition 1. Interpr. $\text{interpr}(vt_a, vt_e, \Delta, t) = \min(\max(t+\Delta+1, vt_a), vt_e)$

Definition 2. Cartesian product. Given two pn-relations r^{pn} and s^{pn} defined on the schemas R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and S: (B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M) respectively and a reference time t, the Cartesian product $r^{pn} \times_t^{pn} s^{pn}$ has schema (A_1, …, A_n, B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M) and it is defined as follows:

$$r^{pn} \times_t^{pn} s^{pn} = \{x \setminus \exists x' \in r^{pn} \wedge \exists x'' \in s^{pn}\ x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge x[B_1, \ldots, B_m]= x''[B_1, \ldots, B_m] \wedge$$
$$x[VT_s]=\max(x'[VT_s], x''[VT_s]) \wedge x[VT_e]=\min(x'[VT_e], x''[VT_e]) \wedge$$
$$x[VT_a]=\max(\min(\text{interpr}(x'[VT_a], x'[VT_e], x'[\Delta_m], t), \text{interpr}(x''[VT_a], x''[VT_e], x''[\Delta_m], t)), x[VT_s]) \wedge$$
$$x[\Delta_m] = (\text{if } x[VT_a]=x[VT_e] \text{ then NR else } -\infty) \wedge x[\Delta_M] = (\text{if } x[VT_a]=x[VT_e] \text{ then NR else } +\infty)\}$$

Definition 3. Certain difference. Given two pn-relations r^{pn} and s^{pn} defined on the same schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, the cert_difference $r^{pn} -_t^{pn\text{-}nec} s^{pn}$ has schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and is defined as follows:

$$r^{pn} -_t^{pn\text{-}nec} s^{pn} = \{x \setminus (\exists x' \in r^{pn} \wedge \neg\exists x'' \in s^{pn}\ x'[A_1, \ldots, A_n]=x''[A_1, \ldots, A_n] \wedge x=x') \vee$$
$$(\exists x' \in r^{pn} \wedge \exists! x_1'', \ldots, x_k'' \in s^{pn}\ x'[A_1, \ldots, A_n]= x_1''[A_1, \ldots, A_n], = \ldots = x_k''[A_1, \ldots, A_n] \wedge$$
$$x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge t'= \text{interpr}(x'[VT_a], x'[VT_e], x'[\Delta_m], t) \wedge$$
$$x[VT_s, VT_a, VT_e, \Delta_m, \Delta_M] \in (\{<x'[VT_s], t', t', x'[\Delta_m], x'[\Delta_M]>\} -^* \{<x_1''[VT_s], x_1''[VT_e], x_1''[VT_e], x_1''[\Delta_m], x_1''[\Delta_M]>, \ldots,$$
$$<x_k''[VT_s], x_k''[VT_e], x_k''[VT_e], x_k''[\Delta_m], x_k''[\Delta_M]>\})\}$$

Definition 4. Possible difference. Given two pn-relations r^{pn} and s^{pn} defined on the same schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, the poss_difference $r^{pn} -_t^{pn\text{-}poss} s^{pn}$ has schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and is defined as follows:

$$r^{pn} -_t^{pn\text{-}poss} s^{pn} = \{x \setminus (\exists x' \in r^{pn} \wedge \neg\exists x'' \in s^{pn}\ x'[A_1, \ldots, A_n]=x''[A_1, \ldots, A_n] \wedge x=x') \vee$$
$$(\exists x' \in r^{pn} \wedge \exists! x_1'', \ldots, x_k'' \in s^{pn}\ x'[A_1, \ldots, A_n]= x_1''[A_1, \ldots, A_n], = \ldots = x_k''[A_1, \ldots, A_n] \wedge$$
$$x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge t_1''= \text{interpr}(x_1''[VT_a], x_1''[VT_e], x_1''[\Delta_m], t) \wedge \ldots \wedge t_k''= \text{interpr}(x_k''[VT_a], x_k''[VT_e], x_k''[\Delta_m], t) \wedge$$
$$x[VT_s, VT_a, VT_e, \Delta_m, \Delta_M] \in (\{<x'[VT_s], x'[VT_e], x'[VT_e], x'[\Delta_m], x'[\Delta_M]>\} -^* \{<x_1''[VT_s], t_1'', t_1'', x_1''[\Delta_m], x_1''[\Delta_M]>, \ldots,$$
$$<x_k''[VT_s], t_k'', t_k'', x_k''[\Delta_m], x_k''[\Delta_M]>\})\}$$

Definition 5. Union, projection and (non-temporal) selection. Given two pn-relations r^{pn} and s^{pn} defined on the proper schema

$$r^{pn} \cup_t^{pn} s^{pn} = \{x \setminus x \in r^{pn} \vee x \in s^{pn}\}$$
$$\pi_{X\ t}^{pn}(r^{pn}) = \{x \setminus \exists x' \in r^{pn}\ x[X] = x'[X]\}$$
$$\sigma_{P\ t}^{pn}(r^{pn}) = \{x \setminus x \in r^{pn} \wedge P(x)\}.$$

Codd's operators constitute the "core" algebra: any algebra is said to *complete* any query language that was as expressive as Codd's set of five relational algebraic operators: relational union ($\cup$), relational difference ($-$), selection (σ_P), projection (π_X), and Cartesian product ($\times$) [29]. However, to facilitate medical applications, several additions to such a

"core" are very important. In the following, we propose some of them (but we envision further additions in our future work).

3.3 Temporal selection operators

One of the advances of our approach to now-relative data is that we explicitly cope with their intrinsic temporal indeterminacy, distinguishing between certain and possible valid times. Such pieces of information are very important for physicians and medical informatics applications such as GLARE. In particular, it is very important to rise queries to select tuples that satisfy some temporal condition. The range of possible temporal conditions is too wide to be explored in just one paper. Here, we just aim at providing physicians with a core of temporal selection operators, suitable to support some of the most common conditions. In particular, we consider temporal selection operators to select tuples whose valid time:

- holds at a specific time point ($\boldsymbol{\sigma^{at}_t}$) or at some time before it ($\boldsymbol{\sigma^{<}_t}$), or at some time after it ($\boldsymbol{\sigma^{>}_t}$),
- has a given duration ($\boldsymbol{\sigma^{dur}_t}$),
- is in any of the temporal relation of Allen's algebra [30] with the valid time of another tuple (i.e., before, after, meets, met by, starts, started by, during, contains, finishes, finished by, equals),
- intersects or is disjoint from the valid time of another tuple ($\boldsymbol{\sigma^{INTERSECTS}_t}$, $\boldsymbol{\sigma^{DISJOINT}_t}$).

Notably, in the case on now-relative data, our approach supports three different views of the valid time: one may want to focus only on "certain" time, or also on "possible" time (**poss**). And, in the case of data with interval latency, one may further distinguish between the "certain" time obtained if the minimum latency is assumed (**cmin**), or the time obtained if the maximum latency is assumed (**cmax**). To support each one of the "views", we define a *poss*, a *cmin* and a *cmax* version for each one of the operators. For the sake of brevity, in Fig. 5 we report the three versions of the $\boldsymbol{\sigma^{point}_t}$, $\boldsymbol{\sigma^{<}_t}$, and $\boldsymbol{\sigma^{>}_t}$ operators (Definition 7), and only the *cmin* version of the others. The other versions are reported in Appendix 3.

Fig. 5. Temporal selection algebraic operators.

Definition 7. Temporal selection σ^{at}, Before-point $\sigma^{<}_t$, After-point $\sigma^{<}_t$. Given a pn-relation r^{pn} defined on the schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M), a time t', and a reference time t, $\sigma^{at\text{-}cmin}_t(r^{pn},t')$, $\sigma^{at\text{-}max}_t(r^{pn},t')$, $\sigma^{at\text{-}poss}_t(r^{pn},t')$, $\sigma^{<cmin}_t(r^{pn},t')$, $\sigma^{<cmax}_t(r^{pn},t')$, $\sigma^{<cposs}_t(r^{pn},t')$, $\sigma^{<cmin}_t(r^{pn},t')$, $\sigma^{<cmin}_t$, $\sigma^{<cmin}_t(r^{pn},t')$ have schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and are defined as follows:

$\sigma^{at\text{-}cmin}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' \in [x[VT_s], \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m], t))\}$

$\sigma^{at\text{-}cmax}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' \in [x[VT_s], \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M], t))\}$

$\sigma^{at\text{-}poss}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' \in [x[VT_s], x[VT_e])\}$

$\sigma^{<cmin}_t(r^{pn},t') = \sigma^{<cmax}_t(r^{pn},t') = \sigma^{<poss}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge x[VT_s] < t'\}$

$\sigma^{>cmin}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m], t)\text{-}1\ \}$

$\sigma^{>cmax}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M], t)\text{-}1\ \}$

$\sigma^{>poss}_t(r^{pn},t') = \{x \setminus x \in r^{pn} \wedge t' < x[VT_e]\text{-}1\ \}$

Definition 8. Temporal-duration selection $\sigma^{dur\text{-}cmin}_t$. Given a pn-relation r^{pn} defined on the schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, $\sigma^{dur\text{-}cmin}_t(r^{pn}, Op, dur)$, where $Op \in \{<,>,=,\leq,\geq,\neq\}$ and *dur* is a duration, has schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and is defined as follows:

$\sigma^{dur\text{-}cmin}_t(r^{pn}, Op, dur) = \{x \setminus x \in r^{pn} \wedge ((\text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) - x[VT_s])\ Op\ dur)\}$

Definition 9. Temporal-relation selection $\sigma^{\text{OP-cmin}}_{t}$ (here OP is a placeholder for one of the temporal relations below) Given two pn-relations r^{pn} and s^{pn} defined on the schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and S: (B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, $\sigma^{OP\text{-}cmin}{}_t(r^{pn})$ has schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and is defined as follows:

$\sigma^{BEFORE\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} \text{ interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) < y[VT_s]\}$

$\sigma^{MEETS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} \text{ interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) = y[VT_s]\}$

$\sigma^{OVERLAPS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} x[VT_s] < y[VT_s] \wedge y[VT_s] < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t)\text{-}1 \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\}$

$\sigma^{STARTS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} x[VT_s] = y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\}$

$\sigma^{DURING\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} x[VT_s] > y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\}$

$\sigma^{FINISHES\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} x[VT_s] > y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) = \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\}$

$\sigma^{EQUALS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} x[VT_s] = y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t) = \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\}$

$\sigma^{AFTER\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} \text{ interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) < x[VT_s]\}$

$\sigma^{METBY\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} \text{ interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) = x[VT_s]\}$

$\sigma^{OVERLAPPEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} y[VT_s] < x[VT_s] \wedge x[VT_s] < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t)\text{-}1 \wedge \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t)\}$

$\sigma^{STARTEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} y[VT_s] = x[VT_s] \wedge \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t)\}$

$\sigma^{CONTAINS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} y[VT_s] > x[VT_s] \wedge \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t)\}$

$\sigma^{FINISHEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn} y[VT_s] > x[VT_s] \wedge \text{interpr}(y[VT_a], y[VT_e], y[\Delta_m],t) = \text{interpr}(x[VT_a], x[VT_e], x[\Delta_m],t)\}$

$\sigma^{DISJOINT\text{-}cmin}{}_t(r^{pn},s^{pn}) = \sigma^{BEFORE\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{AFTER\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{MEETS\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{MET\text{-}BY\text{-}cmin}{}_t(r^{pn},s^{pn})$

$\sigma^{INTERSECTS\text{-}cmin}{}_t(r^{pn},s^{pn}) = \sigma^{OVERLAPS\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{OVERLAPPEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTS\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{DURING\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{CONTAINS\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHES\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHEDBY\text{-}cmin}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{EQUALS\text{-}cmin}{}_t(r^{pn},s^{pn})$

In the following we propose some examples of queries to show the expressiveness of our algebra. In the example, we consider the relations PSYMPTOM in Fig. 2 and PEVIDENCE in Fig. 3.

Postoperative fever lasting more than one week should raise suspicion of a deep infection. For this reason, in the case of Example 8, on May 28 a physician might want to verify when Stephen has certainly had fever. Such a question can be stated through the query Q3 with t = May 28.

Q3) When did Stephen certainly have fever?

$to\_standard\_pess_t(\sigma^{pn}_{t\ Patient=Stephen \wedge Examination=Temperature \wedge Outcome<>Normal}(\text{PEVIDENCE}))$

The answer to the query Q3 is the tuple:

A3) {<Stephen, Temperature, Moderate | May 21, May 26>},

i.e., Stephen certainly had fever from May 21 to May 25. In fact, supposing that, on day May 28, Stephen has not reported any change about its conditions, and considering the maximum latency of the tuple, which is three days, there is the possibility that Stephen's conditions are changed, but they have not been yet reported into the database (for three days). Thus, we are certain that Stephen had fever from May 21 to three days before the query time (May 28), i.e., to May 25.

On day June 1, the system asks again query Q2 with t = June 1. Supposing that the database has not changed, the result of query Q2 is the tuple <Stephen, Temperature, Moderate | May 21, May 30>. In this case, the result clearly shows that Stephen had fever for more than seven days, and further investigations are needed.

In all the following queries, we consider t = August 27.

Q4) What symptoms did Bill manifest while he suffered from high fever?

$\pi_t^{pn}{}_{Symptom}(\sigma_t^{\text{INTERSECTS-}cmin}(\sigma_t^{pn}{}_{Patient=Bill}(\text{PSYMPTOM})), \sigma_t^{pn}{}_{Patient=Bill \wedge Examination=Temperature \wedge Outcome=High}(\text{PEVIDENCE}))$

The result of the query highlights that Bill had chest pain, fatigue and cough:

A4) {<Bill, chest pain | Aug 15, Aug 26, c_{max}, -∞, +∞>, <Bill, fatigue | Aug 1, Aug 26, c_{max}, -∞, +∞>, <Bill, cough | Aug 10, Aug 26, c_{max}, -∞, +∞>}.

Q5) Has John had high fever not in presence of other symptoms?

$\pi_t^{pn}{}_{Patient}(\sigma_t^{pn}{}_{Patient=John \wedge Examination=Temperature \wedge Outcome=High}(\text{PEVIDENCE})) -_t^{pn-poss} \pi_t^{pn}{}_{Patient}(\sigma_t^{pn}{}_{Patient=John}(\text{PSYMPTOM}))$

The result of the query is the empty relation:

A5) {}.

Tom has a suspect gastritis. In order to prescribe an endoscopy, the physician wants to check whether Tom has already a recent negative abdominal x-ray.

Q6) On August 8, has Tom had a (certainly valid) abdominal x-ray examination with negative outcome?

$\sigma^{slice\text{-}pess}{}_{Aug\ 8}(\sigma_{\text{Aug 8}}{}^{pn}{}_{Patient=Tom \wedge Examination=Abdominal\ x\text{-}ray \wedge Outcome=Negative}(\text{PEVIDENCE}))$

The result of the query is:

A6) {<Tom, abdominal x-ray, Negative >}.

In [31] and [32] GLARE was extended with a set of temporal reasoning and knowledge-based methodologies to detect interactions between different CIGs concurrently executed on comorbid patients. Basically, GLARE adopts a knowledge server that maintains, in an OWL ontological model, information about the possible effects of the actions and the temporal relations between actions and effects. Furthermore, the ontological model is provided by a set of rules that, given two or more actions and their execution times, detect the possible interactions occurring between the effects that they can potentially cause. Combining our approach with the knowledge server in [31], we can identify which actions (stored, e.g., in table EXEC_ACTIONS; see Fig. 6) can have caused specific adverse situations during the treatment of a patient.

Fig. 6 Pn-relation EXEC_ACTIONS.

ROWID	Patient	Action	VT_s	VT_a	VT_e	Δ_m	Δ_M
1	Frank	warfarin therapy	Mar 22	Mar 23	c_{max}	-10	0
2	Frank	erythromycin therapy	Jun 11	Jun 11	Jun 22	-1	0
3	Tom	pantoprazole therapy	Aug 9	Aug 15	Oct 9	-10	0
4	Frank	furosemide therapy	Mar 10	Mar 12	c_{max}	-10	0

For instance, suppose that Frank, a patient treated for venous thrombosis, hypertension and an upper respiratory tract infection, manifests an unexpected low value of the International Normalized Ratio (INR) on day June 15. To investigate the possible causes, physicians can query the database to retrieve all the actions currently "active" at the time when INR resulted being low:

Q7) Which therapeutic actions could have been carried out on Frank when his INR was low?

$$\sigma^{CONTAINS\text{-}poss}{}_t(\sigma_t^{pn}{}_{Patient=Frank}(\text{EXEC\_ACTIONS}), \sigma_t^{pn}{}_{Patient=Frank \wedge Examination=INR \wedge Outcome=Low}(\text{PEVIDENCE}))$$

The result of the query is:

A7) {<Frank, warfarin therapy | Mar 22, Mar 23, c_{max}, -10, 0>, <Frank, erythromycin therapy | Jun 11, Jun 11, Jun 22, -1, 0>, <Frank, furosemide therapy | Mar 10, Mar 12, c_{max}, -10, 0>}.

The result of the query can then be used in GLARE to further investigate whether such actions can cause a low value of the INR. Indeed, GLARE would point out that the interaction between warfarin and erythromycin is a probable cause and the physicians can, for instance, have the latter medication suspended.

Before closing this Section, it is worth to make a digression. TDB techniques in general, and our approach in particular, provide users with *new*, *better*, *easier* and *theoretically-grounded* ways of managing time in relational databases (see the discussion in Appendix 1). Specifically, queries like Q5 and Q6 above involve complex temporal operations on the valid time of tuples (intersection and difference, respectively). Such operations are automatically performed by our approach, with no burden for users. Achieving the same result through "standard" non-temporal DBMSs would be very difficult (see, e.g., the explicit examples discussed in the TSQL2 book, to motivate the need for TDBs [3]). In particular, our TDB-like easy treatment of temporal data is important in the medical domain, where user-physicians may be unable (and are certainly not willing) to sustain such a technical complexity.

4. Theoretical and computational results

With our approach, we have achieved three different types of results. The first type has been already discussed throughout all the paper: our approach supports the treatment of now-related patient data overcoming the limitations of the other approaches in the literature. Already in Section 2.1, to motivate our approach, we have shown that, for instance, we correctly manage now-relative facts for which the latency is unknown, while the approaches in the literature fail to cope with them. As an example, we have shown the impact on GLARE's decision support system, that queries a database and relies on the database answers in order to make proper recommendations to physicians. Further on, in Section 3, we have shown that our extended algebra is expressive, allowing to ask different types of queries on the patients' data.

Now, in Sections 4.1 and 4.2, we move to a more "technical" evaluation of our approach. First, in Section 4.1, we show that, besides granting "practical" advantages for clinical applications, our approach is also soundly theoretically grounded, since it supports several fundamental theoretical properties. Second, in Section 4.2, we move to a purely technical analysis of our approach, to show that our approach is ***computationally*** efficient, in that Cartesian product and

difference do not add any significant overhead to "traditional" TDB approaches (both regarding I/O and CPU time). Notably, Section 4.2 does not aim to emphasize the clinical impact of our approach (which is already shown in Sections 2 and 3), so that we adopt data and queries that we have synthesized with the only purpose of analyzing I/O and CPU time of Cartesian product and difference on different types of now-related data.

4.1 Theoretical results.

Our data model easily supports the treatment of standard TSQL2 tuples (i.e., of not now-relative tuples; see the tuples 1 and 4 of Fig. 2).

Property 1: consistent extension (with regard to TSQL2). Any not now-relative tuple can be easily represented as a special case of our data model, in which $VT_a = VT_e$ and d_m=d_M='NR'. ■

Property 2 grants that, if only not now-relative data are used, our algebraic operators behave as TSQL2 ones.

Property 2: Consistent extension (relational algebraic operators). If only not now-relative data are used, our relational operators $\cup_t^{pn}$, $-_t^{pn-cert}$, $-_t^{pn-poss}$, σ_P^{pn}, $\pi_{t,X}^{pn}$ and $\times_t^{pn}$ are equivalent to the standard TSQL2 valid-time relational operators $\cup$, $-$, σ_P, π_X and $\times$. ■

Reducibility is a fundamental property of temporal algebras [3], [33]. To define reducibility, we first introduce the pn-slice operator, which removes the indeterminate part from the valid time of the fact and retains only the certain part giving as a result a standard TSQL2 valid-time relation.

Definition: pn-slice operator. Let r^{pn} be a pn-relation defined over the schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and t a time. The result of the pn-slice operator $\rho_t^{pn}(r^{pn})$ is a standard TSQL2 valid-time relation over the schema (A_1, …, A_n | VT_s, VT_e) defined as follows:

$\rho_t^{pn}(r^{pn})$={<x \ ∃x'∈r^{pn} ∧ x[A_1, …, A_n]=x'[A_1, …, A_n] ∧ x[VT_s]=x'[VT_s] ∧ x[VT_e]=interpr(x'[VT_a], x'[VT_e], x'[Δ_m], t) }

Property: Reducibility of pn-relational algebra to TSQL2 relational algebra. pn-algebraic operators are reducible to TSQL2 valid-time algebraic operators, i.e., for each algebraic operator Op^{pn} that extends Codd's operator to cope with our model – and indicating with Op^T the corresponding TSQL2 valid-time relational operator – for each pn-relations r^{pn} and s^{pn} the following holds (the analogous holds for unary operators):

$$\rho_t^{pn}(r^{pn}\ Op^{pn}\ s^{pn}) = \rho_t^{pn}(r^{pn})\ Op^T\ \rho_t^{pn}(s^{pn}).$$

4.2 Experimental evaluation of the computational complexity.

While the clinical impact of our approach has been discussed in Section 2, here we aim at experimentally studying the *computational complexity* of our algebraic operators (and, in particular, of difference and Cartesian product, which are the only operators that manipulate time). We compare the performance of our approach with an "ideal" (and not realistic) approach in which the exact ending time of now-relative data is known a priori. In such a context, only "standard" temporal tuples have to be managed, so that "classical" TSQL2 representation and algebraic operators can be used. In this way, we are able to study the extra-effort we introduce to cope with "now'' with respect to an ideal approach in which no treatment for "now" is required.

All experimental results are computed on a four 450 MHz CPU—SUN UltraSparc II processor machine, running Oracle 10.2.0 RDBMS, with a database block size of 8K and SGA size of 500 MB. To ensure that the logical read of data already in SGA does not influence the results we flushed the database buffer cache in SGA before every test. At the times of testing the database server did not have any other significant load. We used Oracle built-in methods for statistics collection, analytic SQL functions and the PL/SQL procedural runtime environment.

The experimental evaluation is quite articulated, to consider the different aspects covered by our approach:

(1) We separately consider the different temporal operators that manipulate the temporal attributes (Cartesian product, certain difference and possible difference).
(2) We separately consider different types of relations:
    (2.i) "ideal" approach relations (which are standard TSQL2 relations),
    and pn-relations:
    (2.ii) without now-relative tuples (indicated as "not now" in Tables 4 and 6),
    (2.iii) in which all tuples are valid-time now-relative with exact latency ("now exact ($\Delta m=\Delta M=k$)"),
    (2.iv) with a minimum and a maximum latency ("now $\Delta m<\Delta M$ ($\Delta m\neq-\infty$, $\Delta M\neq+\infty$)"),
    (2.v) with unknown latency ("now unknown ($\Delta m=-\infty$, $\Delta M=+\infty$)").
    (2.vi) mixed pn-relations ("Now mix"), consisting of a mixture of the different types of tuples (50% of the tuples are not now-relative, the rest are now-relative with unknown latency (20%), minimum and maximum latency (20%), exact latency (10%)).
(3) To analyze scalability, we consider three different dimensions for each one of the different types of input relations. In particular, we analyze difference on relations consisting of 10,000, 100,000 and 1,000,000 tuples, and Cartesian product on relations of 300, 1,000, and 3,000 tuples[8].
(4) For each execution, we have measured answer size, CPU time and I/O.

The main goal of our experiments is to compare our approach and the ideal approach considering the different data distributions defined above, in such a way to analyze the impact of the different parameters. For such a reason, we have chosen to carry on our experiments on synthesized data, that we have generated in such a way to follow the general specifications described above. Since experiments only focus on CPU time and I/O, the specific choice of relation schemas and tuple values is inessential. Indeed, for the sake of simplicity, we have used relations of the form of PSYMPTOM and PEVIDENCE in Fig.2 and Fig.3. For each patient, we entered between 10 and 100 tuples.

Tables 3 and 4 contain the results of our tests regarding the difference operation. In Table 3, we report the results of computing the difference operation in the ideal approach. As would be expected, the answer size, the CPU time and the I/O roughly scale with the same rate as the dataset size. In Table 4 we report the results of computing the certain and the possible difference in our approach (i.e., the queries *PSYMPTOM* $-^{pn-cert}$ *PEVIDENCE* and *PSYMPTOM* $-^{pn-poss}$ *PEVIDENCE* respectively) to datasets of the same sizes as the ones used for the ideal approach. All the queries have been executed ten times, and the average answer size, CPU time and I/O are reported. In the tables below, the

[8] Since the result of Cartesian product (independently on whether time is considered or not) has a size quadratic with respect to the number of input tuples, the size of the relations we used for Cartesian product is significantly smaller than the one of the relations used for difference (which, on the other hand, gives results that have size linear with respect to the number of input tuples).

"Dataset size" (i.e., the size of the relations) and the "Answer size" are expressed considering the number of tuples. The "CPU time" has been captured through Oracle built-in system statistics by querying the V$SYSSTAT view, and it is expressed in milliseconds. The "I/O" has been captured through the Oracle built-in system statistics by querying the view V$FILESTAT and it is represented as the number of physical reads done.

TABLE 3. IDEAL APPROACH, DIFFERENCE

	Difference (ideal approach)		
Dataset size (tuples)	*Answer size (tuples)*	*CPU time (ms)*	*I/O (reads)*
10,000	10,129	92	73
100,000	101,379	914	581
1,000,000	1,014,097	8,831	5,594

TABLE 4. OUR APPROACH, CERTAIN DIFFERENCE & POSSIBLE DIFFERENCE

		Certain Difference (pn approach)			**Possible Difference (pn approach)**		
Dataset size (tuples)	*Dataset type*	*Answer size (tuples)*	*CPU time (ms)*	*I/O (reads)*	*Answer size (tuples)*	*CPU time (ms)*	*I/O (reads)*
10,000	Not now	10,129	96	73	10,129	106	73
100,000		101,379	927	581	101,379	1,135	581
1,000,000		1,014,097	9,125	5,594	1,014,097	9,274	5,584
10,000	Now exact ($\Delta_m=\Delta_M=k$)	9,801	99	73	9,913	92	73
100,000		97,934	1,192	654	99,015	876	654
1,000,000		979,922	10,553	6,210	990,250	8,738	6,210
10,000	Now $\Delta_m<\Delta_M$ ($\Delta_m\neq-\infty$, $\Delta_M\neq+\infty$)	9,795	97	66	9,888	92	66
100,000		97,962	1,196	661	99,004	884	580
1,000,000		979,987	10,621	6,219	989,968	9,124	6,219
10,000	Now unknown ($\Delta_m=-\infty$, $\Delta_M=+\infty$)	10,130	99	72	10,130	116	73
100,000		101,435	982	599	101,435	1,136	584
1,000,000		1,018,211	9,312	5,891	1,014,435	11,014	5,877
10,000	Now mix	10,056	98	71	10,021	99	71
100,000		101,129	987	612	101,186	981	608
1,000,000		1,000,097	10,012	5, 946	1,000,024	9,912	5, 941

Answer sizes, CPU times and I/O are roughly the same between certain and possible difference. However, due to the definition of the two operators, in the "Now exact" and "Now $\Delta_m<\Delta_M$" datasets, possible difference is usually greater than certain difference, so that the answer size of possible difference is (slightly) greater. As in the ideal approach, also in our approach the answer size, the CPU time and the I/O roughly scale with the same rate as the dataset size. Comparing the ideal approach with our approach on the "Not now" dataset, it is possible to observe that the answer size is exactly the same, since, of course, exactly the same results are provided (since the data in the two datasets are semantically equivalent). Notably, with the "Not now" dataset, the CPU times are roughly equal between the ideal approach and our approach; this fact shows that no overhead is added by our approach in such a basic situation. Indeed, also the comparisons considering the other datasets confirm that no (or a negligible) overhead is added by our approach with respect to the ideal one.

In Tables 5 and 6 we report the results of the Cartesian product in the ideal approach and in our approach (i.e., the query *PSYMPTOM* $\times^{pn}$ *PEVIDENCE)*. All the queries have been executed ten times, and the average answer size,

CPU time and I/O are reported in the tables. The CPU time grows consistently with the growth of the answer size. On the other hand, the I/O tend to grow more slowly (with respect to the growth of the answer size), mostly because of the policies used by the DBMS to pack data into blocks. Comparing our approach to the ideal one, it is worth noticing that, once again, on the "Not now" dataset, the answer size is exactly the same, and only a negligible CPU-time overhead is added by our approach When also now relative data are taken into (datasets "Now exact", "Now $\Delta_m<\Delta_M$", "Now unknown", "Now mix") it is possible to notice that answer set sizes are significantly greater in our approach than in the ideal one. This is due to the fact that, intuitively speaking, now-relative data are more likely to intersect than not now-relative ones. Such an increase of the answer size justifies the fact that the CPU time in our approach is significantly greater than in the ideal one.

TABLE 5. IDEAL APPROACH, CARTESIAN PRODUCT

	Cartesian product (ideal approach)		
Dataset size (tuples)	*Answer size (tuples)*	*CPU time (ms)*	*I/O (reads)*
300	44,444	236	22
1,000	490,750	2,678	22
3,000	4,415,850	24,988	38

TABLE 6. OUR APPROACH, CARTESIAN PRODUCT

		Cartesian product (pn approach)		
Dataset size (tuples)	*Dataset type*	*Answer size (tuples)*	*CPU time (ms)*	*I/O (reads)*
300	Not now	44,444	249	19
1,000		490,750	2,953	21
3,000		4,415,850	25,404	34
300	Now exact ($\Delta_m=\Delta_M=k$)	70,696	369	13
1,000		783,146	4,716	13
3,000		7,062,054	43,664	21
300	Now $\Delta_m<\Delta_M$ ($\Delta_m\neq-\infty$, $\Delta_M\neq+\infty$)	69,784	449	13
1,000		773,082	4,805	21
3,000		7,001,262	39,143	21
300	Now unknown ($\Delta_m=-\infty$, $\Delta_M=+\infty$)	69,634	356	21
1,000		757,836	4,544	21
3,000		6,976,486	37,605	28
300	Now mix	52,124	335	13
1,000		563,148	4,018	21
3,000		6,102,268	35,742	21

5. Extensions: dealing with transaction time (sketch)

Transaction time is the "time of the database", i.e., the time when a tuple is inserted in the database (starting point of the transaction time) and deleted from the database (ending point of the transaction time) [3]. Many database approaches insist about the importance of supporting, besides valid time, also transaction time (consider, e.g., [3]). We believe that the treatment of transaction time is very important in many domains and may be useful also in the medical domain, since it allows to model the availability (or non-availability) of patients' data at the time when physicians have to take

clinical decisions. In the following, we briefly show that our approach can be easily extended to consider also transaction time.

First of all, it is important to clarify the differences between valid time and transaction time: while the former models the time when (the fact modeled by) a tuple holds in the real world, the latter only concerns the time when the tuple is inserted and (possibly) deleted in the database (see [3] for additional explanations). There is a general consensus in the TDB community that transaction time is ***orthogonal*** with respect to valid time, meaning that a fact can be inserted in the database before it starts to hold (in such a case, there is a *proactive* update of the database; see [3] for examples), at the same time when it starts to hold, or after it starts to hold (*retroactive* update of the database).

In general (consider, e.g., TSQL2 [3] and BCDM [21]), also the domain of transaction time is a set of chronons. However, their granularity may be different from the one of valid time (e.g., nanoseconds vs. seconds), and, since it represents the "database time", transaction time cannot be a future time.

The fact that transaction time is orthogonal with respect to valid time allows us (as in most TDB approaches, including, e.g., TSQL2) to deal with transaction time independently of valid time. And the fact that transaction time does not stretches into the future makes the treatment of transaction-time now-related tuples significantly easier than the treatment of valid-time now-related tuples, since one has not to care about possible persistence in the future. Also, the notion of latency is greatly simplified in this context: transaction time is the time when a tuple is inserted/deleted in the database, so that, by definition, there is no latency. Indeed, several current approaches already cope with ***transaction-time now-related*** tuples in a correct way, so that we do not need to devise a new way to cope with them, but just compose our approach for valid time with one of the approaches to transaction time in the literature. In particular, as regards transaction time, we choose to adopt the consensus approach proposed by BCDM semantics [21], and implemented in 1NF by TSQL2.

In short, in TSQL2 transaction time is represented by a pair of attributes TT_s and TT_e, representing the starting point (i.e., the insertion time) and the ending point (i.e., the deletion time) of a tuple. In case a tuple is transaction-time now-related (i.e., in case the tuple has been inserted in the database, and is still present in the database at the current time – i.e., it has not been deleted yet), the attribute TT_e assume a special value UC (Until Changed)[9].

Bitemporal tuples and relations in our approach can be simply modeled by "composing" the representation of valid time shown in Section 2 with the above representation of transaction time:

Definition: bitemporal pn-tuple and pn-relation. Given a schema $(A_1, \dots, A_n)$ where each A_i represents a non-temporal attribute on the domain D_i, a bitemporal pn-relation r^{pn} is an instance of the schema $(A_1,\dots,A_n|VT_s,VT_a,VT_e,\Delta_m,\Delta_M,TT_s,TT_e)$ defined over the domain $D_1\times \dots \times D_n \times T^C \times T^C \times T^C \times (\mathbb{Z} \cup \{-\infty,' NR'\}) \times (\mathbb{Z} \cup \{+\infty,' NR'\}) \times T'^C \times (T'^C \cup \{\text{UC}\})$, where T^C is the domain of *valid-time chronons*, T'^C is the domain of *transaction-time chronons*, VT_s, VT_a, VT_e, Δ_m and Δ_M are defined as in Section 2, TT_s and TT_e represent the start and the end of the transaction time respectively. Each tuple $x = (a_1, \dots, a_n \,|vt_s, vt_a, vt_e, d_m, d_M, tts, tte) \in r^{pn}$ (where $tt_s \geq vt_a$) is termed a bitemporal pn-tuple.

As an example, Fig. 7 shows the bitemporal relation PSYMPTOM^B, which extends relation PSYMPTOM in Fig.2 to consider also transaction time. In the figure, we assume that only the last two tuples have been logically deleted (on Aug 30), while all the other tuples are transaction-time now-relative. For instance, the first tuple has been inserted at time TT_s=Aug 26 and is still present in the database (i.e., TT_e=UC).

[9] Notably, in TDB approaches, there is no physical deletion of tuples. In order to delete a tuple, the value of the end of its transaction time (i.e., the value of the attribute TT_e) is set to the current time.

Fig. 7. Bitemporal pn-relation PSYMPTOM^B.

ROWID	Patient	Symptom	VT_s	VT_a	VT_e	Δ_m	Δ_M	TT_s	TT_e
1	John	severe headache	Aug 19	Aug 25	Aug 25	NR	NR	Aug 26	UC
2	John	neck stiffness	Aug 20	Aug 26	c_{max}	$-\infty$	$+\infty$	Aug 26	UC
3	Tom	abdominal pain	Jul 3	Aug 7	c_{max}	$-\infty$	$+\infty$	Aug 7	UC
4	Tom	nausea	Aug 1	Aug 4	Aug 4	NR	NR	Aug 7	UC
5	Bill	chest pain	Aug 15	Aug 26	c_{max}	$-\infty$	$+\infty$	Aug 26	UC
6	Bill	fatigue	Aug 1	Aug 26	c_{max}	$-\infty$	$+\infty$	Aug 27	Aug 30
7	Bill	cough	Aug 10	Aug 26	c_{max}	$-\infty$	$+\infty$	Aug 27	Aug 30

The orthogonality between transaction time and valid time can be exploited also in order to define the algebraic operators. As an example, we show the definition of the Cartesian product between bitemporal pn-tuples. The valid time is computed as in Definition 2. Additionally, the intersection between the transaction times of the tuples is also computed (second part of the definition).

Definition 10. Bitemporal Cartesian product. Given two bitemporal pn-relations r^{pn} and s^{pn} defined on the schemas R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M, TT_s, TT_e) and S: (B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M, TT_s, TT_e) respectively and a reference time t, the Cartesian product $r^{pn} \times_t^{pn} s^{pn}$ has schema (A_1, …, A_n, B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M, TT_s, TT_e) and it is defined as follows:

$r^{pn} \times_t^{pn} s^{pn} = \{x \setminus \exists x' \in r^{pn} \wedge \exists x'' \in s^{pn}\ x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge x[B_1, \ldots, B_m]= x''[B_1, \ldots, B_m] \wedge$

$x[VT_s]=\max(x'[VT_s], x''[VT_s]) \wedge x[VT_e]=\min(x'[VT_e], x''[VT_e]) \wedge$

$x[VT_a]=\max(\min(\text{interpr}(x'[VT_a], x'[VT_e], x'[\Delta_m], t), \text{interpr}(x''[VT_a], x''[VT_e], x''[\Delta_m], t)), x[VT_s]) \wedge$

$x[\Delta_m] = (\text{if } x[VT_a]=x[VT_e] \text{ then NR else } -\infty) \wedge x[\Delta_M] = (\text{if } x[VT_a]=x[VT_e] \text{ then NR else } +\infty)$

$\wedge x[TT_s]=\max(x'[TT_s], x''[TT_s]) \wedge$

$x[TT_e]= (\text{if } (x'[TT_s] \neq UC \wedge x''[TT_s] \neq UC) \text{ then } \min(x'[TT_e], x''[TT_e])$

$\text{if } (x'[TT_s]=UC \wedge x''[TT_s] \neq UC) \text{ then } x''[TT_e]$

$\text{if } (x'[TT_s] \neq UC \wedge x''[TT_s]=UC) \text{ then } x'[TT_e]$

$\text{if } (x'[TT_s]=UC \wedge x''[TT_s]=UC) \text{ then } UC$

$\wedge x[TT_s] \leq x[TT_e] \wedge\ x[VT_s] \leq x[VT_e] \}$

A final consideration regards the relationship between the assertion time VT_a and the transaction time. In general, the start TT_s of the transaction time may be equal to the assertion time (i.e., the fact is inserted in the TDB at the time when it is asserted) or greater. However, in some cases, and in particular in medical domains, the assertion time VT_a might not be available. In such cases, the only time available for determining the persistence of now-relative facts (in the case of unknown latency) is the start of the transaction time (which is always available in bitemporal relations). Therefore, in such cases, the start of the transaction time might be used in the place of the (unknown) assertion time.

6. Discussion and Conclusions

Now-relative temporal data play an important role in the medical context, where specific attention is devoted to patient symptoms, treatments, measurements holding at the current time. As a consequence, the development of a relational representation and query language to cope with them in a domain-independent and application-independent way is of primary importance, e.g., to provide once-and-for-all a temporal support to medical decision support systems. However,

now-relative facts are not supported by standard DBMSs and a limited number of relational approaches in the literature face them. Such approaches assume that latency is ***zero*** (i.e., changes in the world are instantaneously recorded into the database) [13], [17] or that it is ***exact*** and ***known*** [16]. Such assumptions constitute clearly an unacceptable limitation in the medical domain. In this paper, we propose an innovative approach in which ***zero***, ***exact***, ***interval*** and ***unknown*** latency may be homogeneously coped with, and we show examples of applications to the medical context. To achieve such a goal, we move from the "traditional" relational representations towards a richest one, in which, as in most AI temporal approaches, we also consider *temporal indeterminacy*, and *certain* and *possible* times of facts. We also propose new temporal algebraic operators to query them, supporting the distinction between possible and necessary time, and Allen's relationships between data [30], which have proved to be very important in the temporal interpretation of data (consider, e.g., *temporal abstraction* [34]). We exemplify the impact of our approach, and study the theoretical and computational properties of the new representation and algebra. From the practical point of view, we have implemented our approach and extensive experiments have demonstrated that our approach only adds a negligible overhead with respect to current temporal database approaches (which do not support "now").

It is worth concluding this paper with a final reflection about the notion of ***now-related facts***, and about their treatment in the area of ***Artificial Intelligence*** (AI). As discussed through the paper, we overcome current relational database approaches by supporting not only exact latency, but also imprecise and unknown latency for now-related facts. This is a crucial advance to extend current TDB approaches to support, e.g., medical decision support systems. Roughly speaking, we provide a domain-independent and application-independent relational data server to represent and query temporal (medical) data. We do so by providing limited "AI-style" capabilities to represent and manage the temporal indeterminacy which is intrinsic in "now-relative" facts. We deliberately limit such capabilities, in such a way to retain the computational "efficiency" of standard temporal relational approaches: indeed, in Section 4.2 we experimentally show that, despite our increased expressiveness, we do not add any significant computational overhead.

On the other hand, in the area of Artificial Intelligence, usually approaches favour expressiveness and powerful inferences over computational complexity, achieving more sophisticated approaches to now-related facts and their persistence. In AI, one goal is to perform appropriate forms of *reasoning* to *infer*, given the current knowledge, how long now-related facts will persist (with no support of users or administrators). In general, the problem of determining *how long now-related facts hold* is an instance of the well-known ***frame problem*** and of ***reasoning about change***, topics which have been widely investigated in AI [35], [36]. The problem is to investigate the persistence of information and the conflict with monotonic reasoning mechanisms such as classical logic. Thus, in AI non-monotonic reasoning mechanisms have been devised such as *circumscription* [37] and they have been applied to the frame problem. More specifically, focusing on temporal reasoning, Dean and McDermott [38] devised an approach for dealing with logical temporal knowledge bases ("databases of assertions" in the terminology of Dean and McDermott) and for managing the persistence of their effects over time. The approach is meant as an extension of predicate-calculus knowledge bases such as PROLOG and the focus is on maintaining consistency over uncertain temporal information about events. Dean and Kanazawa [39] proposed a probabilistic model for dealing with the persistence of propositions over time. In the medical domain, Klimov and Shahar [40] proposed an approach to medical monitoring and related alerting system where raw temporal data about the patient is abstracted using domain knowledge and when a triggering pattern is detected an alarm is fired. All the above-mentioned approaches use the idea of "clipping events", i.e. in the case of a proposition whose temporal ending is unknown, its persistence is stopped by a subsequent "clipping event". For example, Klimov and Shahar use clipping events to stop unnecessary alerts for avoiding users' alert fatigue.

A different approach is followed in the work of McDermott [41]. He defined a first-order logic where a fact can be associated with a characteristic lifetime, which is an interval over which it is reasonable to infer that a fact remains true; a fact remains true until its lifetime is gone or until it is explicitly stated that it ceased to be true; after the lifetime a fact does not necessarily stops being true, but one loses information after that point. Shahar [42] faces the problem of persistence by distinguishing between local persistence (related to a proposition) and global persistence (that "bridges the gap" between the local persistencies of two propositions). Local persistence uses a probabilistic function that gives an exponential decay in the truth value of a fact. The idea of a probabilistic model is similar to the one of Dean and Kanazawa [39], however a major difference lies in the fact that Shahar's probabilistic function is "symmetric", thus modeling a decay in the probability of the truth value that extends both in the future and in the past. It is possible to use such a probabilistic framework to infer the validity of a fact by imposing a threshold above which the fact is deemed to hold.
Tawfik and Neufeld [43] deal with the issue of irrelevance, which is influenced by time and regards identifying what information are relevant and what information are not relevant to draw some conclusions. Tawfik and Neufeld use Markov chains to model the decay of the relevance of a fact over time.

References

[1] Y. Wu, S. Jajodia, and X. S. Wang, "Temporal database bibliography update," in *Temporal Databases: Research and Practice*, vol. 1399, O. Etzion, S. Jajodia, and S. Sripada, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 338–366.
[2] L. Liu and M. T. Özsu, Eds., *Encyclopedia of database systems*. Springer, 2009.
[3] R. T. Snodgrass, *The TSQL2 temporal query language*. Kluwer, 1995.
[4] A. K. Das and M. A. Musen, "A temporal query system for protocol-directed decision support," *Methods Inf. Med.*, vol. 33, no. 4, pp. 358–370, Oct. 1994.
[5] M. J. O'Connor, S. W. Tu, and M. A. Musen, "The Chronus II temporal database mediator," *Proc. AMIA Annu. Symp. AMIA Symp.*, pp. 567–571, 2002.
[6] P. Terenziani, R. T. Snodgrass, A. Bottrighi, M. Torchio, and G. Molino, "Extending temporal databases to deal with telic/atelic medical data," *Artif. Intell. Med.*, vol. 39, no. 2, pp. 113–126, Feb. 2007.
[7] B. Stantic, P. Terenziani, G. Governatori, A. Bottrighi, and A. Sattar, "An implicit approach to deal with periodically repeated medical data," *Artif. Intell. Med.*, vol. 55, no. 3, pp. 149–162, 2012.
[8] L. Anselma, A. Bottrighi, S. Montani, and P. Terenziani, "Managing proposals and evaluations of updates to medical knowledge: theory and applications," *J. Biomed. Inform.*, vol. 46, no. 2, pp. 363–376, Apr. 2013.
[9] L. Anselma, A. Bottrighi, S. Montani, and P. Terenziani, "Extending BCDM to Cope with Proposals and Evaluations of Updates," *IEEE Trans Knowl Data Eng*, vol. 25, no. 3, pp. 556–570, 2013.
[10] L. Anselma, L. Piovesan, and P. Terenziani, "A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy," *J. Intell. Inf. Syst.*, pp. 1–30, Jun. 2015.
[11] L. Anselma, P. Terenziani, and R. T. Snodgrass, "Valid-Time Indeterminacy in Temporal Relational Databases: A Family of Data Models," 2010, pp. 139–145.
[12] L. Anselma, P. Terenziani, and R. T. Snodgrass, "Valid-Time Indeterminacy in Temporal Relational Databases: Semantics and Representations," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2880–2894, Dec. 2013.
[13] K. Torp, C. S. Jensen, and M. H. Böhlen, "Layered Temporal DBMS: Concepts and Techniques," in *Database Systems for Advanced Applications '97, Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA), Melbourne, Australia, April 1-4, 1997*, 1997, vol. 6, pp. 371–380.
[14] P. T. and G. M. and M. Torchio, "A modular approach for representing and executing clinical guidelines," *Artif. Intell. Med.*, vol. 23, no. 3, pp. 249–276, 2001.
[15] A. Bottrighi and P. Terenziani, "META-GLARE: A meta-system for defining your own computer interpretable guideline system - Architecture and acquisition," *Artif. Intell. Med.*, vol. 72, pp. 22–41, 2016.
[16] J. Clifford, T. Isakowitz, C. Dyreson, C. S. Jensen, and R. T. Snodgrass, "On the Semantics of 'Now' in Databases," *ACM Trans. Database Syst.*, vol. 22, pp. 171–214, 1997.
[17] B. Stantic, A. Sattar, and P. Terenziani, "The POINT approach to represent now in bitemporal databases," *J. Intell. Inf. Syst.*, vol. 32, no. 3, pp. 297–323, Jul. 2008.
[18] L. Anselma, B. Stantic, P. Terenziani, and A. Sattar, "Querying now-relative data," *J. Intell. Inf. Syst.*, vol. 41, no. 2, pp. 285–311, Oct. 2013.
[19] "CREATE TABLE," in *Oracle database SQL Language Reference 12c Release 1*, Oracle, 2015, p. 452.

[20] C. S. Jensen and R. Snodgrass, “Temporal specialization and generalization,” *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 6, pp. 954–974, Dec. 1994.
[21] C. S. Jensen and R. T. Snodgrass, “Semantics of Time-Varying Information,” *Inf. Syst.*, vol. 21, pp. 311–352, 1996.
[22] G. H. Guyatt, E. A. Akl, M. Crowther, D. D. Gutterman, H. J. Schuünemann, and for the American College of Chest Physicians Antithrombotic Therapy and Prevention of Thrombosis Panel, “Executive Summary: Antithrombotic Therapy and Prevention of Thrombosis, 9th ed: American College of Chest Physicians Evidence-Based Clinical Practice Guidelines,” *Chest*, vol. 141, no. 2 Suppl, p. 7S–47S, Feb. 2012.
[23] C. Dyreson, “Temporal Indeterminacy,” in *Encyclopedia of Database Systems*, L. Liu and M. T. Ozsu, Eds. Boston, MA: Springer US, 2009, pp. 2973–2976.
[24] J. Melton and A. R. Simon, *SQL: 1999: Understanding Relational Language Components*. Morgan Kaufmann, 2001.
[25] L. Anselma, L. Piovesan, A. Sattar, B. Stantic, and P. Terenziani, “A General Approach to Represent and Query Now-Relative Medical Data in Relational Databases,” in *Artificial Intelligence in Medicine - 15th Conference on Artificial Intelligence in Medicine, AIME 2015, Pavia, Italy, June 17-20, 2015. Proceedings*, 2015, pp. 327–331.
[26] L. Anselma, L. Piovesan, A. Sattar, B. Stantic, and P. Terenziani, “A Comprehensive Approach to ‘Now’ in Temporal Relational Databases: Semantics and Representation,” *IEEE Trans Knowl Data Eng*, vol. 28, no. 10, pp. 2538–2551, 2016.
[27] “SQL3 Support for Time.” [Online]. Available: http://www.cs.arizona.edu/~rts/sql3.html. [Accessed: 06-Sep-2016].
[28] T. Johnston and R. Weis, *Managing time in relational databases: how to design, update and query temporal data*. Amsterdam ; Boston: Morgan Kaufmann/Elsevier, 2010.
[29] E. F. Codd, “A relational model of data for large shared data banks,” *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.
[30] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.
[31] L. Anselma, L. Piovesan, and P. Terenziani, “Temporal detection and analysis of guideline interactions,” *Artif. Intell. Med.*, vol. 76, pp. 40–62, Feb. 2017.
[32] L. Piovesan, L. Anselma, and P. Terenziani, “Temporal detection of guideline interactions,” in *HEALTHINF 2015 - 8th International Conference on Health Informatics, Proceedings; Part of 8th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2015*, 2015, pp. 40–50.
[33] L. E. McKenzie and R. T. Snodgrass, “Evaluation of Relational Algebras Incorporating the Time Dimension in Databases,” *ACM Comput Surv*, vol. 23, no. 4, pp. 501–543, 1991.
[34] Y. Shahar and M. A. Musen, “Knowledge-based temporal abstraction in clinical domains,” *Artif. Intell. Med.*, vol. 8, no. 3, pp. 267–298, Jul. 1996.
[35] J. McCarthy and P. J. Hayes, “Some philosophical problems from the standpoint of artificial intelligence,” in *Machine Intelligence 4*, D. Michie and B. Meltzer, Eds. Edinburgh University Press, 1969, pp. 463–502.
[36] V. Lifschitz, “The Dramatic True Story of the Frame Default,” *J. Philos. Log.*, vol. 44, no. 2, pp. 163–176, 2015.
[37] J. McCarthy, “Applications of circumscription to formalizing common-sense knowledge,” *Artif. Intell.*, vol. 28, no. 1, pp. 89–116, 1986.
[38] T. L. Dean and D. V. McDermott, “Temporal data base management,” *Artif. Intell.*, vol. 32, no. 1, pp. 1–55, 1987.
[39] T. Dean and K. Kanazawa, “A Model for Reasoning About Persistence and Causation,” *Comput Intell*, vol. 5, no. 3, pp. 142–150, Dec. 1989.
[40] D. Klimov and Y. Shahar, “iALARM: An Intelligent Alert Language for Activation, Response, and Monitoring of Medical Alerts,” in *Process Support and Knowledge Representation in Health Care*, D. Riaño, R. Lenz, S. Miksch, M. Peleg, M. Reichert, and A. ten Teije, Eds. Springer International Publishing, 2013, pp. 128–142.
[41] D. McDermott, “A temporal logic for reasoning about processes and plans,” *Cogn. Sci.*, vol. 6, no. 2, pp. 101–155, 1982.
[42] Y. Shahar, “Knowledge-based temporal interpolation,” *J Exp Theor Artif Intell*, vol. 11, no. 1, pp. 123–144, 1999.
[43] A. Y. Tawfik and E. M. Neufeld, “Irrelevance in uncertain temporal reasoning,” in *, Third International Workshop on Temporal Representation and Reasoning, 1996. (TIME ’96), Proceedings*, 1996, pp. 196–202.
[44] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, 1999.
[45] A. K. Das and M. A. Musen, “A foundational model of time for heterogeneous clinical databases,” *AMIA Annu. Fall Symp.*, pp. 106–110, 1997.
[46] R. James and C. Goble, “Survey and critique of time and medical records,” *Medinfo MEDINFO*, vol. 8 Pt 1, pp. 271–275, 1995.

APPENDIX 1

About the usefulness of temporal database techniques

"*A common question that arises when the topic of temporal query languages* [and TDB approaches in general] *is, why are they needed? SQL-92 includes date and time data types. Many applications seem to have gotten along fine using SQL.*"

(taken from TSQL2's book [3, p. 3], one of the manifestos of the TDB community, written out of the consensus of 21 "top" researchers in the area). Presenting TDB techniques to readers who have never directly worked on the treatment of time in relational databases is quite challenging, because:

(i) they often have the feeling that just adding one (for instantaneous events) or two (for durative facts) temporally-valued attributes solves all the problems, and

(ii) they indeed know that temporal data have been managed in many applications without resorting to TDB techniques (since only in the very last few years such techniques have become available in some commercial DBMS).

However, there are many subtle and *hidden* problems that suddenly appear when coping with temporal data (see, e.g., the concrete examples shown in [3, Sec. 1.4]), and ad-hoc solutions to these problems are very difficult to be devised, and are likely to contain errors:

"*Two decades of research into temporal databases have unequivocally shown that a time-varying table, containing certain kinds of DATE columns, is a completely different animal than its cousin, the table without such columns. Effectively designing, querying, and modifying time-varying tables requires a different set of approaches and techniques than the traditional ones taught in database courses and training seminars. Developers are naturally unaware of these research results (and researchers are often clueless as to the realities of real-world application development).*

As such, developers often reinvent concepts and techniques with little knowledge of the elegant conceptual framework that has evolved and recently consolidated…"

(in [44, p. XVIII]).

The "core" problem is the treatment of ***valid time***, i.e., the time when data (i.e., tuples) hold in the real world. Informally, dealing with valid time means being able to *associate*, with each *fact* in the database, the *time when it holds* in the real world, and to *manage* such a time *properly* in each one of the *DB operations*. When using non-TDB approaches, this goal is quite challenging, especially in case one wants to cope with *durative* facts, i.e., with facts that hold not just on a single time unit (e.g., patients' admission in a ward), but on a *time interval* (i.e., a set of time units; e.g., patients' symptoms). Indeed, already SQL-92 provides temporal types, and the possibility of adding two temporally-valued attributes to represent the starting and ending times of the validity of a durative fact. However, in non-TDB approaches, such temporal attributes are *interpreted* by the DBMS as any other attribute while, indeed, they have a *special relationship* with the other (non-temporal) attributes: they, indeed, represent the *time when the values specified by the non-temporal attributes hold*. Such an interpretation has subtle but fundamental *consequences*: for instance, to correctly *join* two temporal tables, one has to perform the *intersection of the valid times* of the tuples being combined (see, e.g., the definition of the *join* operation in BCDM [21], which models the *semantics* of most TDB approaches, including TSQL2). However, such an interpretation is not supported by traditional DBMSs, so that developers of temporal applications had to either hardcode the correct procedures, or to find ad-hoc solutions (which, unfortunately, quite rarely adhere to the correct theory).

The difficulties of dealing with time in "traditional" non-temporal databases have been pointed out by many researchers (see, e.g., [3, Ch. 1]), and also in area of Medical Informatics. For instance, Das and Musen [45] have identified several types of mismatches between the temporal support of standard databases and the richness of clinical data; James and Goble [46] have identified the requirements that medical records impose on a temporal model. The main goal of the research in the TDB area is to overcome such problems, providing once-and-for-all a general-purpose and application-independent solution to the treatment of temporal phenomena in relational databases. A huge effort has been made by TDB researchers (e.g., the cumulative bibliography [1] published in 1998 refers more than 2000 papers). As mentioned above, many TDB results have been consolidated in the TSQL2 consensus approach [3]. In the medical area, Chronus [4] and Chronus II [5] have implemented a subset of TSQL2 focusing on valid time.

Despite the wide range of temporal phenomena covered by the TDB literature, some common *principles* clearly emerge. Two fundamental principles are:

Principle 1. *Durative facts* must be treated as *first-class* entities, that must be modelled as a whole by tuples explicitly containing a representation of the time when they hold (i.e., of their *valid time*, usually represented by the pair of their starting and ending points [3], or by the set of time units when they hold [21]);

Principle 2. Explicit support must be provided for the management of the valid time of durative facts within all the operations (e.g., the definition/implementation of the *join* operation should provide the intersection of valid times).

Unfortunately, since only recently commercial DBMSs have implemented part of the TDB results, in many domains, including the medical domain, developers/users still adopt "standard" non-temporal databases, neglecting in the large majority of cases Principle (2), and often also Principle (1) above. For instance, although the validity of patients' symptoms, of values of laboratory tests, and of clinical findings are usually *durative*, in many medical applications they are represented as *punctual events* (adopting standard "non-temporal" databases), considering just the time when such data are collected. It is then demanded to users/physicians the hard problem of coping with the issue that, indeed, such facts are durative, adopting ad-hoc rules to estimate the possible persistence of such facts (e.g., to infer the end of their validity time). However, the fact that ad-hoc solutions have been and are currently used does not prove that the work in the TDB area is not useful or worth, just in the same way as the fact that physicians already treated patients at times in which computer did not exists does not imply that computers and medical informatics are useless for physicians. Indeed, ad-hoc solutions are costly, and it is difficult to guarantee their correctness, extendibility and generality, while TDB techniques provide "for-free" users with ***new***, *better*, *easier* and *theoretically-grounded* ways of managing time in (relational) databases. To quote once again TSQL2 "consensus" book:

"*The short answer to the question of why are temporal query languages needed,*
is well known to those who have developed temporal database applications:
conventional query languages are inadequate.
TSQL2 is a proof by demonstration that ***there is a better way*** *to implement applications manipulating*
time-varying information"

([3, p. 4]).

APPENDIX 2

Further comparisons with the approaches in the literature

In the following, we compare our approach with respect to the ones of Oracle 12c, Clifford et al., NULL, MIN, MAX, and POINT from a technical point of view, and by considering different assumptions about latency. The main differences between our approach and such approaches are graphically highlighted in Fig. 8 and Fig.9. In the figures, we consider the now-relative and now-bounded fact in Example 9 and we compare its valid time at two different reference times (RT=20 in Figure 7, and RT=30 in Figure 8) in the different approaches.

Example 9. John is in ICU from day 11 to NOW (asserted at time NOW=15). No one can stay in ICU more than 15 days.

Regarding latency, we consider four different options:

Case (i): the latency of the fact in Example 9 is exactly 0.

Case (ii): the latency of the fact in Example 9 is exactly -1 day.

Case (iii): the latency of the fact in Example 9 ranges between -2 and -3 days.

Case (iv): the latency of the fact in Example 9 is unknown.

Our approach correctly copes with all such cases, as shown by rows one to four of Fig. 8 and Fig. 9. In particular, notice that (a) if the latency is unknown, both at RT=20 and at RT=30 one can only be certain that John is in ICU from 11 to 15 (see the fourth row of Fig. 8 and Fig. 9), and (b) we correctly cope with bounds, so that the validity of the fact cannot persist beyond day 25 (see the rows one to four in Fig. 9). The NULL, MIN, MAX and POINT approaches can only cover case (i) (see the fifth row of Fig. 8), and they do not cope with the bound (so that the validity of the fact persists until day 30; see the fifth row of Fig. 9). Clifford et al.'s approach, besides coping with case (i), can also cope with case (ii), but cannot cope with bounds (see the sixth row of Fig. 9). On the other hand, if one exploits the "NULL" value of Oracle 12c to represent "NOW", one simply obtains that the fact always persists in the future (regardless of any bound; see the seventh row of Fig. 8 and Fig. 9). As a concrete example, let us suppose that the latency of the fact in Example 8 is unknown. In such a case, whenever one asks a query (e.g., at RT=20 or at RT=30), one can be certain that John is in ICU only on days 11, 12, 13, 14 and 15. Our approach coping with unknown latency (last row in Fig. 8 and Fig. 9) correctly copes with such a situation. On the other hand, in the MIN, MAX, NULL, POINT, and in Clifford's approach, one erroneously concludes that John is in ICU much longer than day 15. Notably, all such approaches erroneously conclude that John can stay in ICU longer than the final possible future bound for the stay (day 25). An extreme case is the approach based on Oracle 12c, which can conclude from Example 9 that John will certainly be in ICU forever.

Fig. 8. Valid time of the fact in Example 9 at reference time 20, in the different approaches, and considering different possibilities for latency.

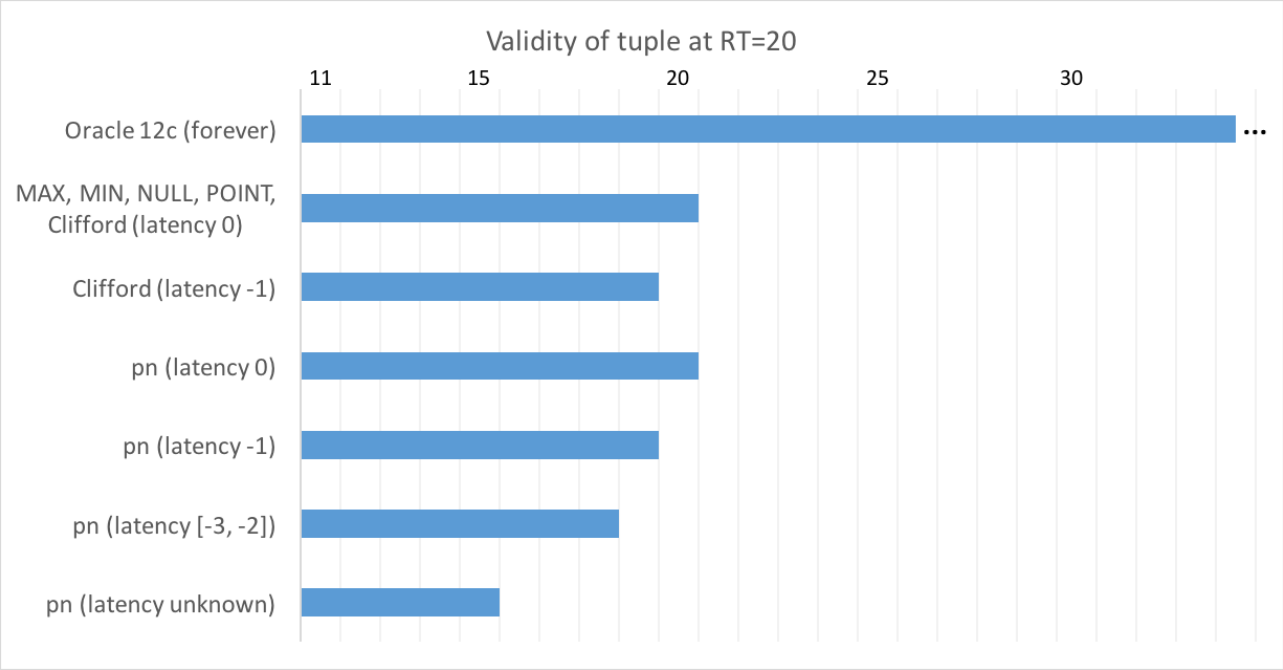
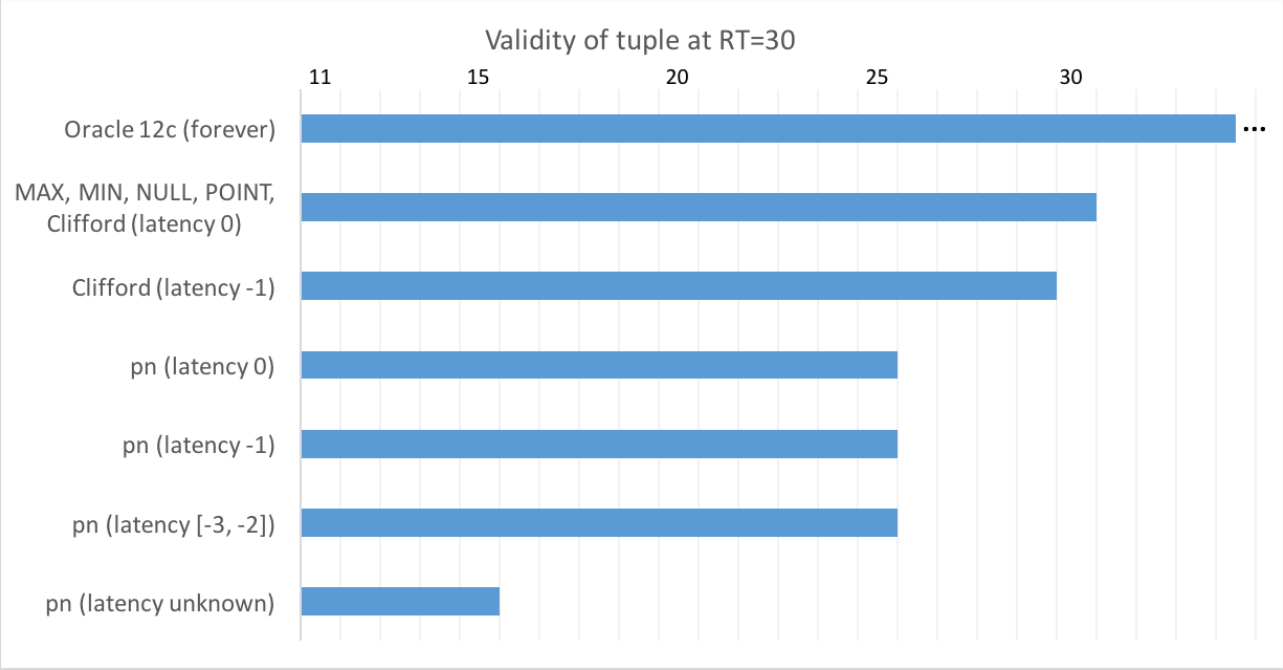


Fig. 9. Valid time of the fact in Example 9 at reference time 30, in the different approaches, and considering different possibilities for latency.



APPENDIX 3

Temporal selection operators.

In the following, for the sake of completeness, we detail the algebraic operators that, for the sake of brevity, we omitted in Section 3.3.

Definition 8'. Temporal-duration selection $\sigma^{\text{dur-min}}_{t}$. Given a pn-relation r^{pn} defined on the schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, $\sigma^{dur\text{-}cmin}{}_t(r^{pn}, Op, dur)$, where $Op \in \{<,>,=,\leq,\geq,\neq\}$ and *dur* is a duration, has schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and is defined as follows:

$\sigma^{dur\text{-}cmax}{}_t(r^{pn}, Op, dur) = \{x \setminus x \in r^{pn} \wedge \text{duration}([x[VT_s], \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t))) \; Op \; dur\}$

$\sigma^{dur\text{-}poss}{}_t(r^{pn}, Op, dur) = \{x \setminus x \in r^{pn} \wedge \text{duration}([x[VT_s], x[VT_e])) \; Op \; dur\}$

Definition 9'. Temporal-relation selection $\sigma^{\text{OP-cmin}}_{t}$ (here OP is a placeholder for one of the temporal relations below) Given two pn-relations r^{pn} and s^{pn} defined on the schema R: (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and S: (B_1, …, B_m | VT_s, VT_a, VT_e, Δ_m, Δ_M) and a reference time t, $\sigma^{OP\text{-}cmax}{}_t(r^{pn})$ and $\sigma^{OP\text{-}poss}{}_t(r^{pn})$ have schema (A_1, …, A_n | VT_s, VT_a, VT_e, Δ_m, Δ_M) and are defined as follows:

$\sigma^{BEFORE\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) < y[VT_s]\}$

$\sigma^{BEFORE\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_e] < y[VT_s]\}$

$\sigma^{MEETS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) = y[VT_s]\}$

$\sigma^{MEETS\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_e] = y[VT_s]\}$

$\sigma^{OVERLAPS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] < y[VT_s] \wedge y[VT_s] < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t)\text{-}1 \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\}$

$\sigma^{OVERLAPS\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] < y[VT_s] \wedge y[VT_s] < x[VT_e]\text{-}1 \wedge x[VT_e] < y[VT_e]\}$

$\sigma^{STARTS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] = y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\}$

$\sigma^{STARTS\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] = y[VT_s] \wedge x[VT_e] < y[VT_e]\}$

$\sigma^{DURING\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] > y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\}$

$\sigma^{DURING\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] > y[VT_s] \wedge x[VT_e] < y[VT_e]\}$

$\sigma^{FINISHES\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] > y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) = \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\}$

$\sigma^{FINISHES\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] > y[VT_s] \wedge x[VT_e] = y[VT_e]\}$

$\sigma^{EQUALS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] = y[VT_s] \wedge \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t) = \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\}$

$\sigma^{EQUALS\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; x[VT_s] = y[VT_s] \wedge x[VT_e] = y[VT_e]\}$

$\sigma^{AFTER\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t) < x[VT_s]\}$

$\sigma^{AFTER\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; y[VT_e] < x[VT_s]\}$

$\sigma^{METBY\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t) = x[VT_s]\}$

$\sigma^{METBY\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; y[VT_e] = x[VT_s]\}$

$\sigma^{OVERLAPPEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; y[VT_s] < x[VT_s] \wedge x[VT_s] < \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t)\text{-}1 \wedge \text{interpr}(y[VT_a], y[VT_e], y[\Delta_M],t) < \text{interpr}(x[VT_a], x[VT_e], x[\Delta_M],t)\}$

$\sigma^{OVERLAPPEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \; \exists y \in s^{pn} \; y[VT_s] < x[VT_s] \wedge x[VT_s] < y[VT_e]\text{-}1 \wedge y[VT_e] < x[VT_e]\}$

$\sigma^{STARTEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] = x[VT_s] \wedge interpr(y[VT_a], y[VT_e], y[\Delta_M],t) < interpr(x[VT_a], x[VT_e], x[\Delta_M],t)\}$

$\sigma^{STARTEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] = x[VT_s] \wedge y[VT_e] < x[VT_e]\}$

$\sigma^{CONTAINS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] > x[VT_s] \wedge interpr(y[VT_a], y[VT_e], y[\Delta_M],t) < interpr(x[VT_a], x[VT_e], x[\Delta_M],t)\}$

$\sigma^{CONTAINS\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] > x[VT_s] \wedge y[VT_e] < x[VT_e]\}$

$\sigma^{FINISHEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] > x[VT_s] \wedge interpr(y[VT_a], y[VT_e], y[\Delta_M],t) = interpr(x[VT_a], x[VT_e], x[\Delta_M],t)\}$

$\sigma^{FINISHEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) = \{x \setminus x \in r^{pn} \exists y \in s^{pn}\ y[VT_s] > x[VT_s] \wedge y[VT_e] = x[VT_e]\}$

$\sigma^{DISJOINT\text{-}cmax}{}_t(r^{pn},s^{pn}) = \sigma^{BEFORE\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{AFTER\text{-}cmax}{}_t(r^{pn},s^{pn})$

$\sigma^{DISJOINT\text{-}poss}{}_t(r^{pn},s^{pn}) = \sigma^{BEFORE\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{AFTER\text{-}poss}{}_t(r^{pn},s^{pn})$

$\sigma^{INTERSECTS\text{-}cmax}{}_t(r^{pn},s^{pn}) = \sigma^{MEETS\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{METBY\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{OVERLAPS\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{OVERLAPPEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTS\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{DURING\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{CONTAINS\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHES\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHEDBY\text{-}cmax}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{EQUALS\text{-}cmax}{}_t(r^{pn},s^{pn})$

$\sigma^{INTERSECTS\text{-}poss}{}_t(r^{pn},s^{pn}) = \sigma^{MEETS\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{METBY\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{OVERLAPS\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{OVERLAPPEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTS\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{STARTEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{DURING\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{CONTAINS\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHES\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{FINISHEDBY\text{-}poss}{}_t(r^{pn},s^{pn}) \cup^{pn}_t \sigma^{EQUALS\text{-}poss}{}_t(r^{pn},s^{pn})$