A comprehensive approach to *'Now'* in temporal relational databases: Semantics and Representation

Luca Anselma, Luca Piovesan, Abdul Sattar, Bela Stantic, Paolo Terenziani

Abstract—Now-related temporal data play an important role in many applications. Clifford et al.'s approach is a milestone to model the semantics of 'now' in temporal relational databases. Several relational representation models for now-related data have been presented; however, the semantics of such representations has not been explicitly studied. Additionally, the definition of a relational algebra to query now-related data is an open problem. We propose the first integrated approach that provides both a neat semantics for now-related data and a compact 1NF representation (data model and relational algebra) for them. Additionally, our approach also extends current approaches to consider (i) domains where it is not always possible to know when changes in the world are recorded in the database and (ii) now-related data with a bound on their persistency in the future. To do so, we explicitly model the notion of temporal indeterminacy in the future for now-related data. The properties of our approach are also analyzed both from a theoretical (semantic correctness and reducibility of the algebra) and from the experimental point of view. Experiments show that, despite our approach is a major extension to current temporal relational approaches, no significant overhead is added to deal with 'now'.

Index Terms— H.2.4.m Temporal databases, H.2.0.b Database design, modeling and management

1 Introduction

TEMPORAL data play an important role in many domains and applications. In such contexts, data must be paired with the time when they occur (*valid* time) and/or when they are inserted/deleted in the database (*transaction* time). Starting from the 1980's, there is a long tradition of approaches coping with time in ***relational*** databases (see, e.g., [1] and [2]). TSQL2 [3] has emerged from the "consensus" of many researchers on relational temporal databases (TDBs for short). Globally, the approaches in the TDB literature cover different aspects:

(1) the definition of the semantics of time in TDBs, including *data semantics* and *query semantics* (usually expressed at the *algebraic* level). For instance, BCDM [4] is the semantic model underlying TSQL2 and several other TDB approaches.
(2) the definition of a *representational* model for temporal data. The basic non-temporal model is usually extended with *new attributes* to *explicitly model time* (e.g., four temporal attributes are added by TSQL2, to model start and end of both valid and transaction time). The meaning of the extended data model can be defined through its *mapping to a semantic model* (e.g., the function *snapshot_to_conceptual* in TSQL2, mapping TSQL2 relations into BCDM semantics).
(3) The definition of algebrae and\or query languages to operate on an extended representational model (as well as insertion\deletion operations).
(4) The study of the properties of the algebrae (or query languages). *Reducibility* is important, to grant, e.g., interoperability with non-temporal databases [3]. Also, the *correctness* of the algebrae operating on the representational model with respect to the semantics should be proven (consider, e.g., [5]).
(5) Last, but not least, the efficiency of many different implementations (often including indexing techniques) has been *experimentally* evaluated.

Despite the huge effort devoted in the area, several problems still have to be further studied. In this paper, we focus on the treatment of *'now'* in *TDBs* to cope with data such as "*John is in the Intensive Care Unit (ICU henceforth) from January 10 to now*". We call valid-time "***now-related***" those facts (tuples) starting in the past and still *valid* until the current time, as in John's example. Analogously, we call transaction-time "***now-related***" those tuples that are still *current* in the database. Though SQL-92 already had the construct CURRENT_TIMESTAMP for use in *queries*, one *cannot store* it as a value in a SQL column (i.e., as a value for the ending time of a tuple). The user is forced to store a specific time, which is clearly problematic and prone to errors (see [6]). Several different approaches have been developed to overcome such a limitation. However, each approach focuses just on one (or few) of the (1) – (5) aspects mentioned above, while there is currently no comprehensive and integrated approach coher-

- *L. Anselma is with the Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy. E-mail: anselma@di.unito.it.*
- *L. Piovesan is with the Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy. E-mail: piovesan@di.unito.it.*
- *A. Sattar is with the Institute for Integrated and Intelligent Systems, Nathan campus, Griffith University, 170 Kessels Road, Brisbane, QLD 4111, Australia. E-mail: a.sattar@griffith.edu.au.*
- *B. Stantic is with the Institute for Integrated and Intelligent Systems, Gold Coast campus, Griffith University, Parklands Drive, Southport, Brisbane, QLD 4222, Australia. E-mail: b.stantic@griffith.edu.au.*
- *P. Terenziani is with the Computer Science Institute, DISIT, Università del Piemonte Orientale, viale Teresa Michel 11, 15121 Alessandria, Italy. E-mail: paolo.terenziani@uniupo.it.*

Published in IEEE TKDE See https://doi.org/10.1109/TKDE.2016.2588490

ently facing all of them. *Data semantics* for now-related data has been studied by the milestone work by Clifford et al. [6]: many of the later works in TDBs have (explicitly or implicitly) assumed such a semantics as the basis of their approach. Notably, however, *no algebra* has been provided by Clifford et al.'s semantics. Concerning *representational* models, several approaches have introduced a variable such as 'now' (other symbols have been used, e.g., "–", "∞", "@" and "until-changed"), as the ending time of now-related tuples, leading to the "***variable***" databases [7]. Variable databases require a significant departure from the "consensus" relational model and cannot be easily implemented on existing relational databases. Concerning non-variable representations, the NULL, MIN, MAX [8] and POINT [9] model have been introduced. Such representational approaches adopt *indexing* techniques to enhance efficiency, and are *experimentally evaluated* and compared [8][9]. An *algebra* for such approaches have been recently proposed by Anselma et al. [10]. However, their data and query *semantics* has not been explicitly explored yet. Two common limitations of *all* the above semantic and representational approaches are that:

(i) their treatment of now-related tuples is based on assumptions on the "*latency*" of updates. Roughly speaking, they assume that it is *exactly known when the changes in the world are recorded into the database*. This is a strong assumption that *does not hold in general*[1]. For instance, in [12], relations are classified on the basis of the interrelationships between changes in the real world and when such changes are recorded in the database. In the definition in [12], in ***general*** temporal relations «*there are no restrictions on the interrelations of, or correlations between, the transaction and valid timestamps of an item*».

(ii) They cannot cope with the possibility of specifying an *upper bound* for the persistence of valid-time now-related tuples in the future (henceforth, we call such tuples "*now-bounded*"). Explicitly coping with such issue involves a deep extension to the model since it requires a treatment also of the *possible* future times when a now-related tuple *may* hold.

Regarding point (ii), consider, e.g., Example 1.

Example 1. Tom was hospitalized in the Emergency Department (ER) yesterday (on day 4), he is currently hospitalized today (on day 5), and the maximum stay in the ER is three days.

Tom is *certainly* hospitalized in the ER on day 4 and 5, and *possibly* hospitalized in the ER tomorrow (on day 6).

In this paper, we propose the *first integrated* approach coping with now-related data, which:

- systematically takes into account all the aspects (1) – (5) mentioned above, and
- overcomes the limitations (i) and (ii) of all current approaches.

[1] Indeed, the orthogonality of valid time and the time when data are inserted/deleted (transaction time) is one of the basic principles of bitemporal databases (see, e.g., TSQL2 [3] and BCDM [11]), and states that valid time and transaction time are independent of each other. This implies that, in the general case, no assumption can be done on when changes in the real (modelled) world are recorded in the database.

In Section 2, we propose a new *semantic model* for now-related data, which *extends* (see Property 1) Clifford et al.'s semantics of 'now' to overcome the limitations (i) and (ii). In Section 3, we then move towards a compact 1NF *representational data model* (which does not adopt variables) to implement the semantic model. We show its *semantics* through a *mapping to the semantic model*. In Section 4, we define *algebraic* (and *update*) *operators* on the *representational* model, and study their *correctness* (with respect to the semantic model proposed in Section 2) and their *reducibility*. In Section 5, we propose an experimental evaluation of our representational approach. The experiments clearly show that our approach does not add any significant overhead to the "ideal" (but not realistic) approach in which the exact ending time of now-relative data is known a priori. In Section 6, we discuss related works. Section 7 contains conclusions.

Proofs and details are reported as supplementary materials.

2 Semantics of now-related data

In this section we focus on the *semantics* of *'now'* in TDBs, while in the rest of the paper we propose and analyze a *representational model* based on it. We interpret *TDB semantics* as in [11], [13]. Also, our notion of TDB semantics is very close to the notion of *extensional-level* databases in [6]. As in such approaches, the semantic model is used to represent the *meaning* of (temporal) data in a neat and formal way, wholly *abstracting from any representation\implementation issue*. Such an abstract semantics can then be used as a *formal specification* for the development representational models. For the sake of completeness, in subsections 2.1 and 2.2 we provide some background about TDB semantics. Then we move to our original contribution (Subsections 2.3 – 2.6).

2.1 Background: temporal database semantics

To introduce TDB semantics, we sketch BCDM (Bitemporal Conceptual Data Model) [11], a unifying and "consensus" semantic model which has been developed to isolate the "core" semantics underlying TDB approaches, including TSQL2 [3]. In BCDM, tuples are associated with *valid time* and *transaction time*. For time, a limited precision is assumed and the *chronon* is the basic time unit. The domain of chronons is totally ordered and isomorphic to a subset of the domain of natural numbers. The domain of valid times D_{VT} is given as a set $D_{VT}=\{c_1,\dots,c_k\}$ of chronons and the domain of transaction times D_{TT} is given as $D_{TT}=\{c'_1,\dots,c'_j\}$ (a distinguished symbol, "UC"–Until Changed– is added to deal with 'now' in transaction time). The schema of a BCDM relation $R=(A_1,\dots,A_n \mid T)$ consists of an arbitrary number of non-timestamp (*explicit* henceforth) attributes $A_1, \dots, A_n$, encoding some fact, and of a timestamp attribute T with domain $D_{TT}\times D_{VT}$. Thus, a tuple $x=(v_1,\dots,v_n \mid t_b)$ in a BCDM relation $r(R)$ on the schema R consists of a number of attribute values associated with a set of bitemporal chronons $c_{bl}=(c_h,c'_i)$, with $c_h\in D_{TT}$ and $c'_i\in D_{VT}$, to denote that the fact $v_1,\dots,v_n$ is current (present in the database) at the chronon c_h and valid at the chronon c'_i. Valid-time only, transaction-time only and nontemporal tuples are special cases, in which either the transaction time, or the valid time, or both of them are not supported. As an example, we show the BCDM semantics of Example 2.

Example 2. Bill has been hospitalized in the Cardiac Surgery department from day 16 to 31; the fact is inserted on (transaction time) day 18 and deleted on day 41.

{<Bill, Cardiac Surgery | {(18,16), (18,17), …, (18,31), (19,16), …,(19,31), …, (40,16), …, (40,31)}>}

Query semantics is modeled by defining a temporal algebra. As in most TDB models, BCDM algebraic operators behave as standard non-temporal operators on the non-temporal attributes and apply set operators on the temporal component of tuples (see, e.g., [5]). Cartesian product involves the intersection of the temporal components, projection and union involve their union, and difference their difference. This definition can be motivated by the *sequenced* semantics [14]: results should be valid independently at each point of time.

Anselma, Snodgrass and Terenziani [5] have recently extended BCDM to cope with *temporal indeterminacy* (i.e., "*don't know exactly when*" indeterminacy [15]). In their semantic model, disjunctive sets of chronons (called DTEs) are introduced, each one representing one of the alternative possible temporal scenarios. Consider, e.g., example 3, in which both the starting and the ending times are indeterminate:

Example 3. John has been in ICU from 9 or 10 until 11 or 12.

Example 3 is modeled in [5] by a DTE representing the disjunction of four different sets of chonons, meaning that John has been hospitalized in ICU at {9,10,11}, or at {9,10,11,12}, or at {10,11}, or at {10,11,12}:

{<John, ICU | {{9,10,11}, {9,10,11,12}, {10,11}, {10,11,12}}>}

In [5] algebraic operators are defined as in BCDM. However, unlike BCDM, they operate on each alternative pair of sets of chonons, to take into account pairwise all the possible combinations of scenarios. For instance, intersection is defined as follows:

$DA \cap^T DB = \{A \cap B \mid A \in DA \wedge B \in DB\}$.

2.2 Background: Clifford et al.'s semantics of 'now'

Clifford, Dyreson, Isakowitz, Jensen and Snodgrass [6] have provided an extended approach coping with the semantics of 'now' both in valid and transaction time. Their approach constitutes a milestone in the treatment of 'now' in TDBs (see, e.g., the Encyclopedia entry [7]). Besides the NOW variable, which is used both in valid and transaction time, Clifford et al. also introduce ***now-relative*** variables NOW+Δ specifying a (positive or negative) span Δ with respect to NOW to model the fact that the specific tuple is updated in advance (in the case of Δ>0) or with a delay (in the case of Δ<0) of Δ time units with respect to the time in which the fact that the tuple models changes in the modeled world. Henceforth, we call ***latency*** such a span of time Δ.

In order to provide the semantics of NOW, NOW+Δ and NOW-related tuples, Clifford et al. explicitly introduce a new type of time, the ***reference time*** (RT), «*to represent the relationship between a temporal database and the "real world" time at which it is viewed*» [6, p. 180]. Notice that RT is different from the *transaction* time and it is not bounded by the current time «*This provides the ability to ask "hypothetical now" queries, that is, from the perspective of a future valid time (i.e., ten years from now)*» [6, p. 182]. The *data semantics* is then provided through a mapping from *variable-level* databases to *extensional-level* databases, called *extensionalization*, and extensionalizations are relative to a specific RT. For the moment we simply assume RT=c_{now}. Roughly and intuitively speaking, in [6], NOW is a variable that assumes new values whenever time progresses. Consider, e.g., Example 4:

Example 4. John is hospitalized in ICU from day 10 to NOW; the fact is inserted at day 10 and is still current.

At reference time 11, the semantics of Example 4 is *<John, ICU | {(10,10), (10,11), (11,10), (11,11)}>*, and at time 12 it becomes *<John, ICU | {(10,10), (10,11), (10,12), (11,10), (11,11), (11,12),(12,10),(12,11),(12,12)}>*.

The semantics of now-relative tuples is similar, except that the display Δ is considered. For instance, in case NOW-1 is used instead of NOW, the semantics of Example 4 at reference time 12 is: *<John, ICU | {(10,10), (10,11), (11,10), (11,11),(12,10),(12,11)}>*.

Note. *The above examples highlight a very important issue: in Clifford et al.'s approach the semantics of NOW is a special case (with Δ=0) of the semantics of NOW+Δ.*

This means that Clifford et al.'s semantics for 'now' *assumes* that the ***latency*** of TDB updates is ***exact*** and ***known***: when some changes happen in the modelled world, they are recorded soon in the TDB (NOW variable; called *punctuality assumption* in [6]), or *exactly* Δ before/after the change (NOW+Δ variable).

Clifford et al. do not devise any algebra coping with such a data model. Notably, Clifford et al. extended their model to cope also with temporally indeterminate tuples (as an *independent* phenomenon, *not used to model the semantics of NOW*). Later on, in [16], Torp et al. have extended Clifford's approach in order to cope with updates.

2.3 Ratio for the proposal of an extended semantics for 'now'

The starting points of our semantics for now-related tuples are [11][6][5]. We extend Clifford et al.'s data model [6] to cope also with (1) "now-bounded" tuples and\or (2) unknown latency.

Considering issue (1), we have already discussed the fact that the treatment of bounds for 'now' involves the explicit treatment of possible future times (see the discussion about Example 1). Possible future valid times can be coped with by models dealing with *temporal indeterminacy*, like the one in [5]. For instance, considering the example, we can state that the Tom's transfer or discharge from the ER is temporally indeterminate since it can occur on day 5 or on day 6. Temporal indeterminacy is also a cue notion to cope with issue (2) above. Indeed, if *latency is unknown*, the valid time of now-related tuples depends only on the time when the now-related fact is asserted (henceforth called *assertion time*), and it is *independent* of the value of NOW. To explain this apparently contradictory point, let us consider an example focusing on valid time only. Consider the fact in Example 5, supposing that it has been asserted at time 14.

Example 5. John is hospitalized in ICU from 10 to NOW (assertion time = 14).

At time 14 (i.e., with c_{now}=14), we are certain that John has been in ICU in days 10, 11, 12, 13 and 14 (as in Clifford et al., we include the current value of NOW). Possibly, John may stay in ICU on day 15, on 16 and so on, but this is not certain. Then, let us look at the same information the day after (i.e., at c_{now}=15), supposing that no modification has been done to the TDB. Clearly, if latency were known, the fact that the TDB has not been changed would provide us an additional piece of information. For instance, with latency equal to zero, we could be certain that John is in ICU also on day 15 (as Clifford et al. clearly state and manage). However, if latency is unknown, the fact that the TDB has not been changed does not provide any new knowledge. It could be the case that John has been discharged on day 15 and this fact has not been recorded yet (e.g., due to a long-term strike of data-entry operators). Even, e.g., at c_{now}=25, the fact that the information about John is still present in the TDB does not convey any additional certain information (e.g., maybe the long-term strike of operators is still going on): we still are only certain that John was in ICU on days 10, 11, 12, 13 and 14 (and it is possible that John was still in ICU in the following days, and even in future days).

As the above example shows, if no assumption can be made on when changes in the modelled world are recorded in the TDB (i.e., if ***latency*** of updates is ***unknown***), the meaning of valid-time now-related facts depends on the assertion time only and it is independent of the value of the variable NOW. As a matter of fact, the (intended) meaning of "the fact *f* holds from *start* to *NOW*", asserted at time t_a is that *f* holds at each chronon from *start* to t_a, and it will end sometime in the future (i.e., some time after t_a). In other words, the semantics of NOW with unknown latency involves ***temporal indeterminacy in the future*** with respect to the ***assertion time*** t_a.

Definition 1 (informal). Assertion time. *Assertion time* is the time when the user expresses (i.e., utters, or writes, or communicate in some way) a given fact (tuple).[2]

Notice that the time when a tuple is inserted in the database (i.e., transaction time) may be different from the *assertion* time (in fact, the tuple may be inserted in the database later than –but never before– the time when the fact is expressed). Also, *assertion time* is different from Clifford et al.'s *reference time*. For instance, the fact described by Example 5 above, which is valid (*valid* time) from *10* to NOW, can be uttered by the user (*assertion* time) at time 14, physically inserted into the TDB (*transaction* time) at time 15, and the database can then be inspected, e.g., at time 25 (*reference* time). Indeed, in our approach, if latency is unknown, *assertion time* is the maximum time until which now-related facts certainly hold: *temporal indeterminacy starts after the assertion time*.

In the following, we provide a formal semantics covering such an intuition, based on the semantics in [5].

2.4 Semantics of '*now*' with unknown latency

We start from the treatment of 'now' in valid time and then, in Section 2.5, we extend it to consider also transaction time.

Definition 2. Semantics of (valid-time) *now-related* tuples with unknown latency. Given a non-temporal tuple f=(a_1,…,a_n) in an instance $r(R)$ of $R(x_1, ..., x_h)$, with valid time starting at c_s and that is now-related and is asserted at time NOW=c_a ($c_s \leq c_a$), the semantics of the relation {*f*} ({*f*} is the relation containing only the tuple *f*) at reference time c_t ($c_t \geq c_a$) is

$\{<a_1,\ldots,a_n \mid \{\{c_s,\ldots,c_a\}, \{c_s,\ldots,c_a+1\}, \ldots, \{c_s,\ldots, c_{max}\}\}\}$,

where, like in BCDM, the temporal domain T^C is an ordered set of chronons $\{c_1,\ldots,c_{max}\}$; c_{max} is the greatest element in T^C. ♦

Notice that, since all the alternative sets contain the chronons c_s,…, c_a, *f* certainly holds in such chronons. In the semantics, all the possible alternative endings of *f* in the future are explicitly modelled. Notably, the semantics of *f* depends on the assertion time c_a (in the sense that the certain chronons span from c_s to c_a) but it is independent of the reference time c_t (c_t must follow the assertion time).

For instance, if the latency is unknown, the semantics of Example 5 at any reference time c_t ($c_t \geq 14$) is

$\{<John,\ ICU \mid \{10,11,12,13,14\}, \{10,11,12,13,14,15\}, \ldots, \{10,11,12,13,14,15, \ldots,c_{max}\}\}$.

Since our semantic model explicitly deals with the temporal indeterminacy about the termination of now-related tuples, it can easily accommodate the semantics of valid-time "*now-bounded*" tuples. As a matter of fact, the bound is simply an upper bound for the possible alternatives in the future as shown in Definition 3.

Definition 3. Semantics of *now-bounded* tuples. Given a non-temporal tuple $f=(a_1,\ldots,a_n)$ in an instance $r(R)$ of $R(x_1, ..., x_k)$, whose validity started at c_s and that is now-related, is asserted at time $NOW=c_a$, and has an upper bound $c_b \in T^C$ ($c_s \leq c_a \leq c_b$), the semantics of $\{f\}$ at time c_t ($c_t \geq c_a$) is

$\{<a_1,\ldots,a_n \mid \{\{c_s,\ldots,c_a\}, \{c_s,\ldots,c_a+1\},\ldots, \{c_s,\ldots, c_b\}\}$ ♦

For instance, considering again Example 5, but supposing that ICU hospitalization cannot last more than 30 days (e.g., for an internal policy of the hospital), we have the following semantics (i.e., c_b=39):

$\{<John,\ ICU \mid \{10,11,12,13,14\}, \{10,11,12,13,14,15\}, \ldots, \{10,11,12,13,14,15, \ldots,39\}\}$.

2.5 Semantics of '*now*' with known latency and/or with Transaction Time

In case the latency of updates is known, we basically maintain the semantics by Clifford et al.

Definition 4. Semantics of (valid-time) *now-related* tuples with known latency Δ. Given a non-temporal tuple f=(a_1,…,a_n) in an instance r(R) of R(x_1, ..., x_h), with valid time starting at c_s and that is now-related and is asserted at time NOW=c_a ($c_s \leq c_a$), the semantics of the relation {*f*} ({*f*} is the relation containing only the tuple *f*) at time c_t

[2] Also Johnston and Weis [17] have pointed out that, besides valid (called effective) time and transaction time, also assertion time should be considered. Their notion of assertion time is quite similar to ours.

(c_t+Δ≥c_a) is
{<a_1, …, a_n | {{c_s,…,c_t+Δ}, {c_s,…,c_t+Δ+1}, …, {c_s,…,c_{max}}}} ♦

Notice that, as in Clifford et al.'s approach, in the case of known latency, the semantics depends on the time c_t when the database is inspected (and it is independent of the assertion time). As in Clifford et al.'s approach, we have that <a_1,…,a_n> certainly holds during the interval starting with c_s and ending with c_t+Δ. On the other hand, we also explicitly model temporal indeterminacy in the future, i.e., the fact that <a_1,…,a_n> can possibly persist until c_{max}. Such an extension is crucial to model now-bounded facts. For instance, Definition 4 above can be easily extended to cope with the case in which *f* has an upper bound *b*, by removing from the formula all the sets containing chronons greater or equal to *b*.

By definition, transaction time is always determinate and cannot be in the future. We thus retain its semantics from Clifford et al.'s approach.

Definition 5. Semantics of transaction-time tuples. Given a non-temporal tuple *f*=(a_1,…,a_n) in an instance r(R) of R(x_1, ..., x_k), inserted at (transaction time) c_i ($c_i \geq c_a$) and still current, the semantics of {*f*} at time c_t ($c_t \geq c_i$) is
{<a_1,…,a_n | { {c_i, …, c_t}}>).♦

Finally, the semantics of bitemporal now-related tuples can be obtained as the composition of the semantics of transaction-time and of valid-time now-related tuples. As an example, we show the semantics of bitemporal now-related and now-bounded tuples with known latency Δ.

Definition 6. Semantics of bitemporal now-related and now-bounded tuples with known latency Δ. Given a non-temporal tuple *f*=(a_1,…,a_n) in an instance *r(R)* of *R*(x_1, ..., x_k), whose validity started at c_s and that is now-related, is asserted at time NOW=c_a, and has an upper-bound $c_b \in T^C$ ($c_s \leq c_a \leq c_b$), is inserted at (transaction time) c_i ($c_i \geq c_a$) and is still current, with a known latency Δ, the semantics of {*f*} at time c_t ($c_t \geq c_i$) is
*{<a_1, …, a_n | {{(c_i,c_s),…,(c_i,c_t+Δ), …, (c_t,c_s),…,(c_t,c_t+Δ)},
{(c_i,c_s),…,(c_i,c_t+Δ+1), …, (c_t,c_s),…,(c_t,c_t+Δ+1)}, …,
{(c_i,c_s),…,(c_i, c_b), …, (c_t,c_s),…,(c_t, c_b)}}}.* ♦

Property 1 shows that, if we neglect the cases in which latency is 'UNK', and we consider only the "certain" part of valid time, our semantics for now-related tuples reduces to Clifford et al.'s one. This is important to grant that our semantics is a consistent extension of Clifford et al.'s one, so that it is the theoretical basis to grant the interoperability of our approach with all the approaches based on Clifford et al.'s semantics (e.g., NULL, MIN, MAX, and POINT approaches).

Property 1. For each non-temporal tuple *f*=(a_1,…,a_n) whose valid time started at c_s and that is now-related, asserted at time NOW=c_a, inserted at (transaction time) c_i ($c_i \geq c_a$) and still current, with known latency Δ and with an upper bound c_b ($c_s \leq c_a \leq c_b$), the semantics of {*f*} at a reference time *t* ($t \geq c_a$) in our approach (indicated by $Semantics_t(f,c_s,c_i,\Delta,c_b)$) is reducible via the *Cert* operator to its semantics in Clifford et al. (indicated by $Semantics^{Clifford}_t(f,c_s,c_i,\Delta,c_b)$), i.e.,
$Cert(Semantics_t(f,c_s,c_i,\Delta,c_b)) = Semantics^{Clifford}_t(f,c_s,c_i,\Delta,c_b)$
where $Cert(<f \mid \{\sigma_1,\ldots,\sigma_k\}>)=<f \mid \sigma>$, with $\sigma=\{\sigma_1 \cap \ldots \cap \sigma_k\}$♦

The proofs of this property and of the following ones are reported in the supplementary materials.

2.6 Semantics of queries (algebra)

As shown above, valid-time and/or transaction-time now-related tuples can be modelled in Anselma et al.'s approach [5] as a specific case of temporal indeterminacy. As a consequence, the temporal algebraic operators (trivially extended to consider also bitemporals) in such an approach can be adopted to query such kind of data.

3 A REPRESENTATIONAL MODEL BASED ON THE SEMANTICS

The above semantics of now-related data is expressive but it has several limitations from the implementation point of view. Our semantic data model is not 1NF and it is not compact at all. Additionally, relational algebraic operators like Cartesian product and difference must explicitly manage all possible combinations of alternative times, and this fact increases the time complexity of such operations. In this section, we propose a compact 1NF representation data model and show its *mapping* over the above semantics (function $\boldsymbol{Sem_t}$ in Definition 8). Section 4 proposes a new temporal algebra operating on the representational model, and proves its *correctness* with respect to the semantics (Property 2).

3.1 A 1NF representational data model

In this subsection, we propose a compact 1NF representation for now-related tuples. Such a representation takes into account both (i) valid time and (ii) transaction time. For now-related valid time, it copes with the case in which (iii) the latency Δ of updates is a known constant value or (iv) it is unknown, and it also deals with (v) now-bounded tuples. The definition of relations considering only transaction time, or only valid time is easier, and can be easily derived from Definition 7.

Definition 7. Bitemporal pn-tuple and pn-relation (where "pn" stands for "possibly now-related"). Given a schema (A_1, …, A_n) where each A_i represents a non-temporal attribute on the domain D_i, a bitemporal relation r^{pn} is an instance of the schema (A_1, …, A_n | *TTs, TTe, VTs, VTa, VTe,* Δ) defined over the domain $D_1 \times \ldots \times D_n \times T^C \times T^C \times T^C \times T^C \times T^C \times (Z \cup \{UNK,NR\})$. The constant 'UNK' stands for "unknown" and it is used for tuples with an unknown latency, while 'NR' stands for "not-relevant" and it is used for not valid-time now-related tuples. A tuple $x = (a_1, \ldots, a_n \mid t_s, t_e, v_s, v_a, v_e, d) \in r^{pn}$ is termed a pn-tuple (***p****ossibly* ***n****ow-related* tuple), while r^{pn} is called pn-relation. In a pn-tuple it must hold that (i) $t_s \leq t_e$, (ii) $v_s < v_e$, (iii) $v_s \leq v_a \leq v_e$. ♦

Intuitively speaking, and considering a valid-time now-related tuple, v_s represents the start of valid time, v_a the assertion time (plus 1, for technical reasons: in fact, we assume intervals closed to the left and open to the right) and v_e the future bound for 'now' (plus 1; the value c_{max} is used in case no bound has to be modelled). Δ, which may be either an integer number or the special values 'UNK' or 'NR', represents the latency of updates. Intuitively, considering tuples with unknown latency, the interval [v_s,

v_a) includes the set of valid-time chronons in which the tuple is certainly valid, while the interval [v_a, v_e) represents the set of chronons in which it possibly holds. On the other hand, in case Δ is an integer value d (i.e., Δ is different from 'UNK' and 'NR'), the certain chronons can extend past v_a to include also all chronons from v_s to $t+d$, where t is the reference time. However, they can never exceed v_e.

As regards transaction time, t_s represents the starting time (the time when a tuple is inserted) and t_s the ending time (the time when a tuple is deleted). t_s and t_e cannot be future times. As suggested in the POINT approach [18], we represent transaction-time now-related tuples (i.e., tuples which are current in the database) by imposing $t_e=t_s$ (notice that there is no ambiguity in the representation, since we adopt the convention that all time intervals are closed to the left and open to the right).

Notice that a tuple that is not valid-time now-related can be easily represented as a special case of the above representation in which $v_a=v_e$ (and with value 'NR' for the attribute Δ). Notice also that a tuple that is transaction-time not now-related can be easily represented as a special case of the above representation, in which $t_s<t_e$. Thus, pn-relations can include heterogeneous types of tuples, in the sense that any of them, independently of the others, may be now-related as regards valid time and/or transaction time, or not now-related at all. Moreover, valid-time now-related tuples in the same relations may have different bounds and/or different latencies. For instance, Table 1 contains three different types of tuples. The first row represents a standard (not now-related) tuple, representing the fact that Bill has been in the Cardiac Surgery Ward from 16 to 32 (certain valid time), and that the tuple has been inserted in the database at time 18 and deleted at time 42 (transaction time). The second row models a valid-time and transaction-time now-related tuple, not bounded, with a known latency. It represents the fact that at time 14 it was asserted (VTa contains the value of the assertion time plus 1) that John is in ICU from 10 to 'now', and that such a tuple has been inserted in the database at time 21, and it is still present in the database. Notably, latency is -1, which means that, e.g., at reference time 30, we are certain that John was in ICU from 10 to 29 (see Definition 4). The third row represents a valid-time now-related and now-bounded fact with unknown latency and still current in the database. The fact that Tom is in ER from 4 to 'now' has been asserted at time 4 and the upper bound for 'now' is 7 (e.g., to model the fact that the maximum stay in ER lasts three days). Since latency is unknown, at reference time 5 (or any RT>5) we are only certain that Tom was in ER at time 4, while he might have been in ER at times 5 and 6.

TABLE 1. Tabular representation of a pn-relation PWARD

Patient	Ward	TTs	TTe	VTs	VTa	VTe	Δ
Bill	Cardiac Surgery	18	42	16	32	32	NR
John	ICU	21	21	10	15	Cmax	-1
Tom	ER	5	5	4	5	7	UNK

Notice that the representation in Definition 7 is a compact 1NF representation of the semantic concepts discussed in Section 2 above. Thus, we define the Sem_t function (where t stands for the chosen reference time, $t \in T^C$), which maps a bitemporal pn-tuple into the equivalent bitemporal tuple in the semantic model.

Definition 8. Semantics of bitemporal pn-tuple and pn-relation. Given a bitemporal pn-relation r^{pn} which is an instance of the schema ($A_1, \ldots, A_n$ | TTs, TTe, VTs, VTa, VTe, Δ) and given any pn-tuple $x = (a_1, \ldots, a_n \mid t_s, t_e, v_s, v_a, v_e, d) \in r^{pn}$, the semantics $Sem_t(\{x\})$ of the relation $\{x\}$ at reference time t is defined as follows:

$Sem_t(x) = (a_1, \ldots, a_n \mid SemTT_t(t_s, t_e) \times SemVT_t(v_s, v_a, v_e, d))$,

$SemTT_t(t_s, t_e) =$

(i) $\{\{c \in T^C \setminus t_s \leq c \leq t\}\}$ (if $t_s=t_e$)

(ii) $\{\{c \in T^C \setminus t_s \leq c \leq min(t, t_e-1)\}\}$ (if $t_s<t_e$)

$SemVT_t(v_s, v_a, v_e, d) =$

(iii) $\{\{c \in T^C \setminus v_s \leq c \leq min(max(t+d,v_a-1), v_e-1)\}, \{c \in T^C \setminus v_s \leq c \leq min(max(t+d,v_a-1), v_e-1)+1\}, \ldots, \{c \in T^C \setminus v_s \leq c \leq v_e-1\}\}$ (if $d \neq UNK$)

(iv) $\{\{c \in T^C \setminus v_s \leq c \leq v_a-1)\}, \{c \in T^C \setminus v_s \leq c \leq v_a)\}, \ldots, \{c \in T^C \setminus v_s \leq c \leq v_e-1\}\}$ (if $d=UNK$)

The semantics $Sem_t(r^{pn})$ of a pn-relation r^{pn} is the set resulting from the application of Sem_t to each one of the pn-tuples $x^{pn} \in r^{pn}$. ♦

The semantics of the temporal component of a pn-tuple depends on the reference time t and it is given by the Cartesian product of the semantics of its transaction time, specified by the function $SemTT_t$, and the semantics of its valid time, specified by the function $SemVT_t$. This approach grants for the orthogonality of TT and VT, as in most TDBs approaches, including BCDM and Clifford et al.'s semantics (see also Property 1 above).

The $SemTT_t$ function provides two cases: the transaction time is now-related, i.e., the tuple is current in the TDB (point (i)). In this case we consider the chronons from the transaction-time start t_s to the reference time t. On the other hand, if the transaction time is not now-related, i.e., the tuple has been deleted (point (ii)), the chronons of the transaction time cannot be greater than the chronon of the time when the tuple has been deleted (minus 1 because the interval is open to the right).

Also the $SemVT_t$ function provides two cases, depending on the latency of the tuple. If the latency is known (point (iii)) and has value d, the tuple certainly holds at all chronons from c_s to $t+d$ (or to v_a, in case $v_a>t+d$), but no longer than the bound v_e-1 (see the discussion in Section 3.2). Then, it possibly holds until v_e-1. It is worth noticing that, if the tuple is not now-related and thus $v_a=v_e$, the definition gives a singleton set of alternatives with the chronons that start at v_s and end at v_e-1 and it does not depend on the value of the reference time t.

If the latency is unknown (point (iv)), we cannot assume any persistence after the assertion time (consider the discussion in Section 3.1). Since, independently of the latency, the valid times from v_s to v_a-1 are certain, they are included in all the alternatives. On the other hand, the valid times from v_a to v_e-1 are possible and they correspond to the other alternatives. Notice that, in such a case, the semantics is independent of the value of the reference time t.

Notably, our representation is expressive enough to cope with all the different cases of now-related facts discussed in Section 2. As discussed in that section, our semantics for now-related tuples is indeed an extension of Clifford et al.'s one (see Property 1). Thus our representation is, indeed, a representational model for Clifford et al.'s semantics of NOW, as well as for the extension we have proposed. This is a first advance with respect to the current literature. A second, major, one is our proposal of an algebra operating on such a compact representation, respecting (i.e., consistent with) the above semantics.

4 Algebraic and Manipulation Operations

Our representational model is a compact 1NF implementation of the semantics in Section 2.4. In this section, we define new algebraic operators operating on such a representation. Our operators perform a "symbolic manipulation" on such a representation: the result is directly computed only on the basis of the compact representation, without resorting to its underlying semantics. This procedure is efficient since it only requires a symbolic manipulation of a compact representation, but demands a proof of correctness: we have to prove that the semantics of the output obtained through the symbolic manipulation is the same that would be obtained (although much less efficiently) by operating at the semantic level through the algebra in Section 2.5.

To operate on the representation in a correct way (but not resorting to a direct translation into the semantics, which would make our approach inefficient), we introduce the function *interprVTa*. *interprVTa* takes in input the valid-time values of a pn-tuple and a reference time t and returns a chronon representing the end of the "certain" part of the valid time at the reference time t, as implied by the representation.

Definition 9. InterprVTa.

```
interprVTa($v_a$,$v_e$,$d$,$t$)
    if ($d$ = UNK or $d$ = NR) then return $v_a$
    else return $min(max(t+d+1, v_a), v_e)$ ♦
```

This is indeed an implementation of part of the semantics in Definition 7. The returned value is respectively (i) the assertion time (v_a) if d=UNK or d=NR (notice that, in the case that the tuple is not now-related, v_a=v_e); (ii) the reference time shifted by the latency ($t+d$) (or the assertion time v_a if it is higher) if d is known.

4.1 Temporal extension of Codd's operators

Now we provide our temporal extension to Codd's relational algebra operators [19]. As, e.g., in BCDM and TSQL2, to grant reducibility, 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 the start/end of valid and transaction time. The definition of such operators is reported in the following, for the sake of completeness. Notice that, as in Clifford, queries indicate the reference time. When not specified, the current time c_{now} is used as default value for it.

Definition 10. Given two pn-relations r^{pn} and s^{pn}, defined over the schema (A_1, …, A_n | *TTs, TTe, VTs, VTa, VTe*, Δ), and reference time t, the relational union, projection and selection are defined as follows (we denote with A the attributes A_1, …, A_n).

$r^{pn} \cup_t^{pn} s^{pn} = \{x \setminus (x \in r^{pn} \vee x \in s^{pn})\}$

$\pi_{t\ X}^{pn}(r^{pn}) = \{x \setminus \exists x' \in r^{pn}\ x[X] = x'[X]\}$

$\sigma_{t\ P}^{pn}(r^{pn}) = \{x \setminus x \in r^{pn} \wedge P(x)\}$. ♦

Notably, as, e.g., in TSQL2 [3], the above operators do not modify the values of the temporal attributes.

On the other hand, as, e.g., in BCDM and TSQL2, our Cartesian product performs the intersection of valid and transaction time.

Definition 11. Cartesian product. Given two pn-relations r^{pn} and s^{pn} defined on the schemas R: (A_1, …, A_n | *TTs, TTe, VTs, VTa, VTe*, Δ) and S: (B_1, …, B_m | *TTs, TTe, VTs, VTa, VTe*, Δ) 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 | *TTs, TTe, VTs, VTa, VTe*, Δ) and it is defined as follows:

$$
\begin{aligned}
& r^{pn} \times_t^{pn} s^{pn} = \{x \setminus \exists x' \in r^{pn}\ \exists x'' \in s^{pn} \\
& \quad (x[A_1, \dots, A_n]=x'[A_1, \dots, A_n] \wedge \\
& \quad x[B_1, \dots, B_m]= x''[B_1, \dots, B_m] \wedge \\
& \quad x[TTs]=max(x'[TTs], x''[TTs]) \wedge \\
& \quad x[TTe]= \\
& \qquad\quad (if\ (x'[TTs] \neq x'[TTe] \wedge x''[TTs] \neq x''[TTe])\ then \\
& \qquad\qquad min(x'[TTe], x''[TTe]) \\
& \qquad\quad if\ (x'[TTs]=x'[TTe] \wedge x''[TTs] \neq x''[TTe])\ then\ x''[TTe] \\
& \qquad\quad if\ (x'[TTs] \neq x'[TTe] \wedge x''[TTs]=x''[TTe])\ then\ x'[TTe] \\
& \qquad\quad if\ (x'[TTs]=x'[TTe] \wedge x''[TTs]=x''[TTe])\ then \\
& \qquad\qquad max(x'[TTs],x''[TTs])) \wedge \\
& \quad x[VTs] = max(x'[VTs], x''[VTs]) \wedge \\
& \quad t'_a=interprVTa(x'[VTa, VTe, \Delta], t) \wedge \\
& \quad t''_a=interprVTa(x''[VTa, VTe, \Delta], t) \wedge \\
& \quad x[VTa]=max(min(t'_a, t''_a), x[VTs]) \wedge \\
& \quad x[VTe]=min(x'[VTe], x''[VTe]) \wedge \\
& \quad x[\Delta]=(if\ (x[VTa]=x[VTe]\ then\ NR\ else\ UNK) \wedge \\
& \quad x[TTs] \leq x[TTe] \wedge x[VTs]<x[VTe]))\}. \ \blacklozenge
\end{aligned}
$$

Cartesian product operates directly on the representation without resorting to the semantics. It manages the non-temporal attributes A_1, …, A_n, B_1, …, B_m in a standard way and, intuitively speaking, evaluates the intersection of the temporal parts of the paired tuples. Concerning transaction times, four cases are distinguished, depending on whether none, one or both the tuples are transaction-time now-related, and following the POINT representation (e.g., the condition $x'[TTs] \neq x'[TTe]$ is used to ascertain that x' is not transaction-time now-related). On the other hand, concerning valid time, intersection is computed by exploiting the *interprVTa* function and latency is set to UNK unless the result is not now-relative; in such a case the latency is not relevant.

As in some approach to indeterminate time (see, e.g., [20]), we choose to propose two different algebraic operators for difference: the *certain difference* $-_t^{pn\_cert}$, and the *possible difference* $-_t^{pn\_poss}$. In the certain difference, we are interested only in certain results. A chronon is certainly in the result of difference if it is certain in the minuend, and if it does not appear (neither as certain nor as possible) in the subtrahend. The certain difference uses the *interprVTa* function to determine the end of the certain part of the valid time.

Definition 12. Certain difference. Given two pn-relations r^{pn} and s^{pn} defined on the same schema R: ($A_1, \ldots, A_n \mid TTs, TTe, VTs, VTa, VTe, \Delta$) and a reference time t, the cert_difference $r^{pn} -_t^{pn\_nec} s^{pn}$ has schema ($A_1, \ldots, A_n \mid TTs, TTe, VTs, VTa, VTe, \Delta$) and it is defined as follows:

$$
\begin{aligned}
& r^{pn} -_t^{pn\_nec} s^{pn} = \{x \setminus \exists x' \in r^{pn}\ \neg\exists x'' \in s^{pn} \\
& (x'[A_1, \ldots, A_n]=x''[A_1, \ldots, A_n] \wedge x=x') \vee \\
& \exists x' \in r^{pn}\ \exists! x''_1, \ldots, x''_k \in s^{pn} \\
& (x'[A_1, \ldots, A_n]=x''_1[A_1, \ldots, A_n]=\ldots=x''_k[A_1, \ldots, A_n] \wedge \\
& x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge \\
& t'=interprVTa(x'[VTa, VTe, \Delta], t) \wedge \\
& x[T] \in (\{<x'[TTs],x'[TTe],x'[VTs],t',t',x'[\Delta]>\} -^* \\
& \{<x''_1[TTs],x''_1[TTe],x''_1[VTs],x''_1[VTe],x''_1[VTe],x''_1[\Delta]>, \\
& \ldots, <x''_k[TTs],x''_k[TTe],x''_k[VTs],x''_k[VTe],x''_k[VTe], \\
& x''_k[\Delta]>\}))\}. \blacklozenge
\end{aligned}
$$

As already pointed out by the BCDM model, for each tuple $x \in r^{pn}$, the times of all the tuples $x_1, \ldots, x_k \in s^{pn}$ that are *value-equivalent* to it must be subtracted from the time of x. The uniqueness operator $\exists!$ is used to identify 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 one of the elements in the first set. Since we are evaluating certain difference, we consider only the "certain" valid time for the minuend (so that the end of its valid time is *interprVTa(x'[VTa,VTe,Δ], t)*, and "possible" valid times for subtrahends (so that the end of their valid time is *x'[VTe]*). Each element has the form $<t_s,t_e,v_s,v_a,v_e,d>$, where $v_e=v_a$. The operation computes binary difference between two elements $<t1_s,t1_e,v1_s,v1_e,v1_e,d1>$ and $<t2_s,t2_e,v2_s,v2_e,v2_e,d2>$ as follows:

(1) for transaction time, (i) it computes the difference between the two time intervals $[t1_s,t1_e)$ and $[t2_s,t2_e)$, considering that both intervals are represented using the POINT representation for 'now'. Zero, one or two intervals (in the POINT representation) are provided as output. Let *TT_diff_set* be the set of such intervals. Moreover (ii) it computes the intersection between them. At most one intersection interval is returned. Let *TT_inters_set* the set of such intervals;

(2) for valid time, it computes the standard difference between the two time intervals $[v1_s,v1_e)$ and $[v2_s,v2_e)$. Zero, one, or two intervals are provided as output. Let *VT_set* be the set of such intervals;

(3) for each interval $[t_s,t_e) \in TT\_diff\_set$, it adds $\{<t_s,t_e,v1_s,v1_e,v1_e,NR>\}$ to the set of results;

(4) for each interval $[t_s,t_e) \in TT\_inters\_set$ and for each interval $[v_s,v_e) \in VT\_set$, it adds $\{<t_s,t_e,v_s,v_e,v_e,NR>\}$ to the set of results.

A more detailed definition of $-^*$ and of binary difference is provided in the supplementary materials. The definition of the *possible difference* ($-_t^{pn\_poss}$) is omitted since it is analogous to the definition of the certain difference. Notably, in the possible difference we want as output possible chronons, i.e., those chronons which are possible in the minuend, and are not certain in the subtrahend (possible chronons in the subtrahend are not considered by possible difference, since the subtrahend tuple may not hold in such chronons).

4.2 Additional algebraic operations

New operators, which are not an extension of Codd's ones, can be introduced to cope with the temporal aspects. In particular, since we consider both "certain" and "possible" valid times for now-related tuples, it is worth introducing the *to_poss* and *to_nec* operators, which coerce pn-relations (which contain a certain degree of indeterminacy) into "standard" determinate-time relations. Such operators retain the transaction time, and set the valid time to the possible times (*to_poss*) or certain valid times (*to_cert*) of the tuple respectively, and are useful to enhance the integration between pn-relations and "standard" temporal relations.

Definition 13. Given a pn-relation r^{pn} defined on the schema R: ($A_1, \ldots, A_n \mid TTs, TTe, VTs, VTa, VTe, \Delta$) and a reference time t, $to\_poss_t(r^{pn})$ and $to\_nec_t(r^{pn})$ have schema ($A_1, \ldots, A_n \mid TTs, TTe, VTs, VTe$) and are defined as follows:

$to\_poss_t(r^{pn})=\{x \setminus \exists x' \in r^{pn}\ (x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge x[TTs]=x'[TTs] \wedge x[TTe]=x'[TTe] \wedge x[VTs]=x'[VTs] \wedge x[VTe]=x'[VTe])\}$

$to\_cert_t(r^{pn})=\{x \setminus \exists x' \in r^{pn}\ (x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge x[TTs]=x'[TTs] \wedge x[TTe]=x'[TTe] \wedge x[VTs]=x'[VTs] \wedge x[VTe]=interprVTa(x'[VTa,VTe,\Delta], t))\}$. ♦

For instance, considering Table 1 and RT=30, to_poss_{30}(Table 1) = {<Bill, Cardiac Surgery | 18,42,16,32>, <John, ICU | 21,21,10,c_{max}>, <Tom, ER | 5,5,4,7>} and to_cert_{30}(Table 1) = {<Bill, Cardiac Surgery | 18,42,16,32>, <John, ICU | 21,21,10,29>, <Tom,ER | 5,5,4,5>}.

Additional operators, such as temporal selection, can be easily introduced.

4.3 Examples of queries

Let us consider the relation PWARD in Table 1 and the relation PSYMPT in Table 2 storing patients' symptoms. In the following we provide some examples of queries, asked, e.g., at reference time 20 (see the subscript of all the algebraic operators).

TABLE 2. The pn-relation PSYMPT

Patient	Symptom	TTs	TTe	VTs	VTa	VTe	Δ
Bill	chest pain	18	18	14	32	32	NR
John	headache	21	21	8	15	c_{max}	-1
Tom	abdominal pain	5	5	3	5	c_{max}	UNK

Q1) Which patients were in cardiac surgery while John was in ICU?

$\pi_{20}{}^{pn}{}_{Patient}(\sigma_{20}{}^{pn}{}_{Ward=Cardiac\ Surgery}(\mathrm{PWARD}) \times_{20}{}^{pn}$
$\pi_{20}{}^{pn}{}_{Ward}(\sigma_{20}{}^{pn}{}_{Patient=John\ \wedge\ Ward=ICU}(\mathrm{PWARD})))$

Q2) When was Bill (certainly) in cardiac surgery while John was not in ICU?

$\sigma_{20}{}^{pn}{}_{Patient=Bill\ \wedge\ Ward=Cardiac\ Surgery}(\mathrm{PWARD}) -_{20}^{pn\_cert} \sigma_{20}{}^{pn}{}_{Patient=John\ \wedge}$

$_{Ward=ICU}$(PWARD)

Q3) What symptoms did John have while he was in ICU?

$\pi_{20}^{pn}{}_{Symptom}(\sigma_{20}^{pn}{}_{Patient=John}$(PSYMPT)) $\times_{20}^{pn}$ $\sigma_{20}^{pn}{}_{Patient=John\ \wedge\ Ward=ICU}$(PWARD))

Q4) Did (possibly) John have headache while he was not in ICU, and when specifically?

$\pi_{20}^{pn}{}_{Patient}(\sigma_{20}^{pn}{}_{Patient=John\ \wedge\ Symptom=headache}$(PSYMPT)) $-_{20}^{pn\_poss}$ $\pi_{20}^{pn}{}_{Patient}(\sigma_{20}^{pn}{}_{Patient=John\ \wedge\ Ward=ICU}$(PWARD))

Q5) When was John (certainly) in ICU?

$to\_cert_{20}(\sigma_{20}^{pn}{}_{Patient=John\ \wedge\ Ward=ICU}$(PWARD))

4.4 Manipulation operations

Here we define insertion and deletion manipulation operations for our representation model (updates can be defined on top of them).

Definition 14. Insertion and deletion. Given a relation *r*, instance of the schema ($A_1, \ldots, A_n$ | *TTs, TTe, VTs, VTa, VTe*, Δ), the insertion and the deletion of a tuple at time c_{now} in the relation *r* are defined as follows.

$insert^{pn}(r^{pn}, <a_1, \ldots, a_n>, v_s, v_a, v_e, d>) =$
$\quad r^{pn} \cup \{<a_1, \ldots, a_n \mid c_{now}, c_{now}, v_s, v_a, v_e, d>\}$

$delete^{pn}(r^{pn}, <a_1, \ldots, a_n>, v_s, v_a, v_e) =$
$\quad r^{pn} - \{<a_1, \ldots, a_n \mid t_s, t_e, v_s, v_a, v_e, d>\} \cup \{<a_1, \ldots, a_n \mid t_s, c_{now}, v_s, v_a, v_e, d>\}$
$\quad\quad$ (if $\exists t_s,t_e,d\ (<a_1, \ldots, a_n \mid t_s,t_e,v_s,v_a,v_e,d> \in r^{pn} \wedge t_s = t_e)$)
$\quad r^{pn}$ (otherwise) ♦

The *insert*^{pn} function inserts a new tuple in the relation r^{pn}. The new tuple, following the POINT representation, has the chronon c_{now} as both transaction time start and end.

The *delete*^{pn} function deletes an existing tuple from a relation r^{pn}. This is done by changing its transaction-time end to c_{now} (provided that the tuple has not been deleted).

4.5 Properties

We can now analyze the theoretical properties of our algebra. Two properties are of paramount importance in this context: *correctness* with respect to the semantics and *reducibility*.

As regards *correctness*, it is worth noting that we have introduced a compact representation model to represent and query now-related data, based on the semantics in Section 2. All the algebraic relational operators defined on pn-relations perform symbolic manipulations, working at the representation level on pn-tuples (thus not resorting to their underlying semantics) and providing as a result pn-relations. This procedure, although efficient, requires a proof of correctness.

Property 2. Correctness of pn-relational algebra. For each operator op_t^{pn} working on pn-relations, we have to prove that the semantics of the result of op_t^{pn} applied to pn-relations is equivalent to the corresponding operation performed at the semantic level. Formally speaking, for a binary operator, we have to prove that

$$Sem_t(r^{pn}\, op_t^{pn} s^{pn}) = Sem_t(r^{pn})\, op^{ref} Sem_t(s^{pn})$$

where op^{ref} is the semantic operator in the reference approach [5] equivalent to op_t^{pn} (the proof for unary operators is similar). ♦

The algebraic operators that do not manipulate time (non-temporal selection, projection, union) are trivially correct. On the other hand, the correctness of our Cartesian product is proven by showing that, for each $c \in T^C$,

$$Sem_t(r^{pn} \times_t^{pn} s^{pn}) = Sem_t(r^{pn}) \times^{ref} Sem_t(s^{pn})$$

The difference is similar.

The *reducibility* to the standard non-temporal algebra is widely recognized as a "must" for temporal algebras (since it supports. e.g., interoperability with pre-existent non-temporal databases; see e.g., [21][3]). We prove that our algebra is reducible to TSQL2's one (notice that, in turn, TSQL2 is reducible to the standard non-temporal algebra). To prove reducibility, we first introduce the pn-slice operator.

Definition 15. pn-slice operator. Let r^{pn} be a pn-relation defined over the schema ($A_1, \ldots, A_n$ | *TTs, TTe, VTs, VTa, VTe*, Δ) and *t* a reference time. The result of the pn-slice operator $\rho_t^{pn}(r^{pn})$ is a standard TSQL2 relation over the schema ($A_1, \ldots, A_n$ | *Ts, Te, Vs, Ve*), where *Ts, Te, Vs* and *Ve* are the attributes representing the start and end of the transaction and valid time, defined as follows:

$\rho_t^{pn}(r^{pn}) = \{x \setminus \exists x' \in r^{pn}$
$\quad (x[A_1, \ldots, A_n]=x'[A_1, \ldots, A_n] \wedge$
$\quad x[Vs] = x'[VTs] \wedge x[Ve]=interprVTa(x'[VTa, VTe, \Delta], t) \wedge$
$\quad x[Ts]=x'[TTs] \wedge$
$\quad x[Te]=(if\ x'[TTs]=x'[TTe]\ then\ UC$
$\quad\quad if\ x'[TTs]\neq x'[TTe]\ then\ x'[TTe]))\}$. ♦

The pn-slice operator, given a pn-relation and a reference time, removes the indeterminate part and retains only the certain part giving as a result a standard (possibly transaction-time now-related) TSQL2 relation.

Property 3. Reducibility of pn-relational algebra to TSQL2 algebra. Pn-algebraic operators are reducible to TSQL2 valid-time algebraic operators, i.e., for each algebraic operator op_t^{pn} that extends a Codd's operator to cope with our model – and indicating with op^T the corresponding TSQL2 relational operator – for each $t \in T^C$ and for each pair of pn-relations r^{pn} and s^{pn} the following holds (the analogous holds for unary operators):

$\rho_t^{pn}(r^{pn}\ op_t^{pn}\ s^{pn}) = \rho_t^{pn}(r^{pn})\ op^T\ \rho_t^{pn}(s^{pn})$. ♦

5 Experimental Evaluation

In this section, we first discuss our implementation and then we experimentally evaluate the performance of our temporal algebra.

5.1 Implementation of algebraic operators

We have developed a prototypical implementation of our approach using PL/SQL. As an example, we describe our implementation of Cartesian product between two pn-relations *r* and *s*, given a reference time *t*.

```
procedure CartesianNow
Input: relation r with schema
        R:(A1,…,An|TTs,TTe,VTs,VTa,VTe,Δ),
        relation s with schema
        S:(B1,…Bm|TTs,TTe,VTs,VTa,VTe,Δ),
        reference time t
Output: relation res of schema
```

```
      RES:(A1,…An,B1,…Bm|TTs,TTe,VTs,VTa,
      VTe,Δ)
cursor curs_r is select A1,…An,
TTs,TTe,VTs,VTa,VTe,Δ from r;
cursor curs_s is select B1,…Bm,
TTs,TTe,VTs,VTa,VTe,Δ from s;
open curs_r;
loop
  fetch curs_r into a1,…an,
  tt1s,tt1e,vt1s,vt1a,vt1e,d1;
  exit when curs_r%notfound;
  open curs_s;
  loop
    fetch curs_s into b1,…bm,
    tt2s,tt2e,vt2s,vt2a,vt2e,d2;
    exit when curs_s%notfound;
    vts := greatest(vt1s, vt2s);
    vta := greatest(least(
      interprVTa(vt1a,vt1e,d1,t),
      interprVTa(vt2a,vt2e,d2,t)),vts);
    vte := least(vt1e,vt2e);
    tts := greatest(tt1s, tt2s);
    if (tt1s <> tt1e and
    tt2s <> tt2e) then
      tte := least(tt1e,tt2e);
    else if (tt1s = tt1e and
    tt2s <> tt2e) then
      tte := tt2e;
    else if (tt1s <> tt1e and
    tt2s = tt2e) then
      tte := tt1e;
    else if (tt1s = tt1e and
    tt2s = tt2e) then
      tte := tts;
    if (vta = vte) then
      d := 'NR';
    else
      d := 'UNK';
    if (tts ≤ tte and
    vts < vte) then
      insert into res (A1,…,An,B1,…Bm,
      TTs,TTe,VTs,VTa,VTe,Δ) values
      (a1,…,an,b1,…bm,tts,tte,
      vts,vta,vte,d);
  end loop;
  close curs_s;
end loop;
close curs_r;
```

In the CartesianNow procedure above, for each pair of tuples x′ (belonging to r) and x″ (belonging to s) the variables *tts, tte, vts, vta, vte, d* represent the temporal attributes of the resulting tuple. First, the algorithm evaluates the values of such variables accordingly to Definition 8 (notice that, to perform temporal intersection, CartesianNow uses the function *interprVTa* already described in Definition 9). Then, the algorithm checks whether the transaction-time and the valid time of the resulting tuple are not empty (i.e., tts ≤ tte and vts< vte). If so, it adds a new tuple with *tts, tte, vts, vta, vte, d* as temporal attributes and the original non-temporal attributes of x′ and x″ as non-temporal attributes (as in Codd's Cartesian product) to the output relation *res*; otherwise, no tuple is added to the output.

5.2 Indexing

In our previous work [9] we have shown that spatial indexing techniques are very efficient to address now-related tuples in relational queries when they are modeled with the POINT approach (the same approach that we followed in this paper to cope with transaction time). However, since the experiments for Cartesian product consider all tuples from both relations, indexing the relations would not bring benefits. Regarding difference, since it subtracts only value-equivalent tuples (considering, in our example, the attribute 'Patient'), we could index the attribute 'Patient' with a B+-tree index. Notice that, since in the experiments about Cartesian product and difference no temporal selection operation is performed on the temporal attributes, indexing on temporal attributes would not be useful to improve the experimental results.

5.3 Experimental evaluation

We are not aware of any other algebra explicitly coping with now-related facts (except the POINT approach [10][18], which, however, is only partially based on Clifford et al.'s semantics; indeed, it does not explicitly provide any semantics for NOW). Therefore, we have chosen to compare the performance of our approach with an "ideal" (but 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. To enhance the quality of the comparison, in our tests we implemented the TSQL2 algebraic operators using the same algorithmic structure we adopted for our operators (e.g., using the nested loop structure for both Cartesian products). Of course, the "ideal" approach involves omniscience, which is not a realistic assumption. However, it can be used to highlight what is the extra-effort we introduce to cope with 'now' with respect to an ideal case in which no treatment for 'now' is required.

In the following, we provide a detailed description of the context and modalities we adopted for our experiments, discussing the setup, the data set types and sizes, the data distributions as well as the adopted measures and schemas.

Setup of the experimental evaluation. 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. Due to the compatibility issues with our previous experiments and with previously developed work, we have chosen to run experiments on an older version of Oracle (version 10.2.0). However, our initial testing on Oracle 12c release 1 has shown that, when the databases have the same startup parameters, the experimental results for Cartesian product and difference are the same as on the 10.2.0 version, since in our experiments we do not query temporal periods and we do not exploit the built-in support for Valid, Transaction, and Decision time that version 12.c offers. 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.

Datasets: We considered different types of datasets (third column of Tables 3, 4 and 5). For the ideal approach we used a standard dataset of TSQL2-like bitemporal tuples (with no now-related tuples); for our approach, in order to investigate different data options and their influence on performance, we considered five different types of datasets, specifically:

(1) "not now" - without now-relative tuples,
(2) "TT now" - where all tuples are transaction-time now-related, but not valid-time related,
(3) "VT unk" - where all tuples are valid-time now-related but no transaction-time related and with unknown latency,
(4) "VT delta" - where all tuples are valid-time now-related but no transaction-time related and with a known latency, and
(5) "Mix" - consisting of a mixture of the different types of tuples. In particular, as regard valid time, 60% of the tuples are not now-related, 20% are now-related with unknown latency, and 20% are now-related with known latency; as regards transaction time, 40% are not now-related and 60% are now-related.

For difference, we generated the tuples in such a way that 10% of the tuples in the subtrahend relation is value-equivalent to a tuple in the minuend relation.

We also considered different sizes for the datasets (first column of the Tables). In particular, notice that the size of the datasets in the experiments on Cartesian product is relatively small and it has been limited to a maximum of 3,000 tuples. This is due to the fact that, since Cartesian product pairwise combines tuples and has a quadratic complexity, it generates a large answer size (indeed, in the case of 3,000 tuples is up to 4,307,488 tuples). However, we estimate that the CPU usage increases linearly with the answer size (see Table 3 and Figure 1). We obtained similar conclusions for difference when we performed experiments with bigger datasets, up to one million tuples (see Table 4).

For all the relations, we used a fixed value for the reference time (RT=300) and we distributed the values of the other temporal attributes (except for the Δ attribute) in the following way:

1. "ideal" - the distribution of TTe and VTe is a Gaussian distribution centered at RT = 300 with values ranging from 250 to 350 while the distribution of TTs and VTs is the following: TTs (VTs) are x units of time before TTe (VTe), where x is randomly distributed between 1 and 100.
2. "not now" - we used exactly the same distributions as for the "ideal" approach. VTa is always equal to VTe and Δ is 'NR'.
3. "TT now"- for VTs, VTe and TTs we used the same distributions as the "ideal" approach. VTa is always equal to VTe and TTe is equal to TTs. Δ is 'NR'.
4. "VT unk"- for TTs and TTe we used the same distributions as the "ideal" approach. The distribution of VTa is a Gaussian distribution centered at RT = 300 with values ranging from 250 to 350. VTs is x before VTa, where x is a random value between 1 and 100. VTe is y after VTa, where y is a random value between 1 and 100. Δ is 'UNK'.
5. "VT delta" - TTs, TTe, VTs, VTa and VTe are as in the "VT unk" approach. The distribution of Δ is a Gaussian distribution centered at 0 and ranging from -10 to +10.
6. "mix" - the different types of tuples are inserted in the relations, according to the proportions discussed above. Depending on the type of the tuple, its temporal attributes are valued according with the distributions discussed in (2)-(5).

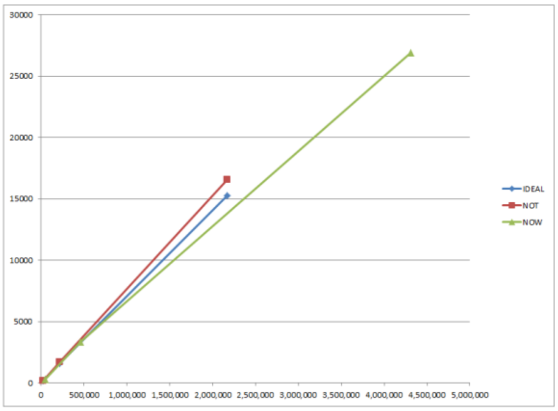

Figure 1. Linear increase in CPU usage as a factor of the answer size.

Measures: In each execution, we have measured the answer size (number of tuples), physical disk I/O and CPU time (units of computation; fourth, fifth and sixth columns of Tables 3 and 4 respectively).

Structure of the evaluation: The experimental evaluation is quite articulated, to consider the different aspects covered by our approach. We present comparisons regarding the most complex operations, i.e. Cartesian product and difference, which require quite complex operations on the temporal attributes (see Definitions 11 and 12).

Schema of the temporal relations: The schema of the temporal relations in both the "ideal" approach and our approach simply consists of a non-temporal attribute plus the temporal ones, as shown in Tables 1 and 2.

5.4 Results and Discussion

The results of the experimental evaluations are shown in Table 3 (Cartesian product) and 4 (Certain difference, the results for possible difference are similar). Our approach is indicated by 'PN' in column 2. For different dataset sizes (first column of Tables 3 and 4), we compare the "Ideal" approach and our approach ("PN" approach), considering Answer size, I/O and CPU time.

TABLE 3. Cartesian product

Dataset size	*Approach*	*Dataset type*	*Answer size*	*I/O*	*CPU time*
300	Ideal	-	20,810	11	158
	PN	Not now	20,810	11	165
		VT unk	20,004	11	165
		TT now	43,710	11	260
		VT delta	23,620	11	189
		Mix	25,638	11	219
1000	Ideal	-	216,660	13	1,562
	PN	Not now	216,660	13	1,713
		VT unk	243,002	13	1,185
		TT now	460,816	13	3,360
		VT delta	230,686	13	1,829
		Mix	250,530	13	2,114
3000	Ideal	-	2,170,998	28	15,242
	PN	Not now	2,170,998	28	16,563
		VT unk	2,169,498	28	16,524
		TT now	4,307,724	28	26,932
		VT delta	2,166,994	28	16,715
		Mix	2,809,488	28	21,563

Considering Cartesian product, Table 3 shows that our approach behaves like the "ideal" one as concerns the I/O. Some overhead is added to the CPU time, due to the growth of the answer size. For instance, if all tuples are current ("TT now" relations), the size of Cartesian product increases since there are more intersections between bitemporal tuples. As a consequence, the answer size and the CPU usage of our approach is larger than the one of the "ideal" approach. In Figure 1 we show that the CPU usage linearly increases with the increase of answer size, in the ideal approach ("ideal") and in our approach, both with the dataset without now-related tuples ("not") and in the one with them ("now").

Results concerning certain difference are reported in Table 4 (organized as for Cartesian product).

TABLE 4. Certain difference

Dataset size	*Approach*	*Dataset type*	*Answer Size*	*I/O*	*CPU Time*
10,000	Ideal	-	10,472	74	130
	PN	Not now	10,472	74	132
		VT unk	10,604	74	160
		TT now	10,481	74	151
		VT delta	10,640	74	134
		Mix	10,571	74	134
100,000	Ideal	-	104,725	714	1,172
	PN	Not now	104,725	714	1,149
		VT unk	106,302	649	1,542
		TT now	104,407	667	1,149
		VT delta	105,894	682	1,191
		Mix	105,223	732	1,137
1,000,000	Ideal	-	1,047,416	6,342	11,605
	PN	Not now	1,047,416	6,337	11,486
		VT unk	1,063,418	6,480	13,308
		TT now	1,044,034	6,385	11,524
		VT delta	1,058,768	6,482	11,841
		Mix	1,054,418	6,383	11,515

Also such results clearly indicate that our approach behaves like the "ideal" one. There are some small variations with regard to the CPU time. However, a closer look can reveal that such variations are also caused by different answer sizes, which are influenced by the actual data.

6 Comparisons with related works

Several approaches have faced the treatment of now-related data in relational TDBs. Some recent approaches have provided coalescing [22], or have focused on indexing [23]–[25] or on timestamping [26], [27]. As stressed along this paper, the approach by Clifford et al. [6] is a milestone, since it first pointed out the semantics of 'now' in TDBs (see Section 2.2). The relationship between our semantics and Clifford et al.'s one has been already discussed throughout the paper. In particular, Property 1 shows that, if we neglect the cases in which latency is 'UNK', and we consider only the "certain" part of valid time, our semantics of now-related tuples reduces to Clifford et al.'s one. Thus, our approach extends Clifford et al.'s semantics considering also *unknown latency* and providing a *relational algebra* for the (semantic) data model. We also propose a *representational implementation* for both the data model and the algebra, *experimentally* proving its efficiency (with respect to the "ideal" approach)[3].

[3]It is also worth pointing out that Clifford et al.'s semantics also models temporal indeterminacy not related to NOW. The "full" treatment of temporal indeterminacy sharply increases the computational complexity of the approaches (see the complexity analysis in [5], which considers a wide family of alternative approaches). In our paper, we only focus on the treatment of now-related data, thus we consider only the degree of indeterminacy strictly needed to cope with (the semantics of) NOW. In such a way, as widely discussed in Section 5, we have experimentally demonstrated that our approach roughly behaves like the "ideal" one, not adding any significant overhead to cope with now-relative data (while a substantial increase of complexity would be required for a full treatment of temporal indeterminacy [5]). We have already proposed a work considering temporal indetermi-

Several *representational models* to cope with 'now' in the relational context have been proposed. MIN, MAX and NULL approaches [8] support now-related tuples by representing NOW as a special value for the valid-time end, i.e., the minimum chronon (MIN approach) and 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: at query time the special value is replaced with the current time. More recently, the POINT approach has been proposed [18], which outperforms the MAX, MIN and NULL approaches. Even more recently, a relational algebra has been defined to fully support querying NOW-related data (i.e., without resorting to the instantiation to the current time) in all MAX, MIN, NULL and POINT approaches [10]. All such approaches are (implicitly) based on Clifford et al.'s semantics, assuming latency equal to zero.

Considering the representational model (Section 3), our approach proposed the addition of two temporal attributes ("VTa" and "Δ"). While the former resembles the additional attributes used in [5] to distinguish between possible and certain times (although there are subtle differences), the explicit treatment of latency (through the "Δ" attribute) constitutes an original and innovative contribution of our approach, leading to deep implications in the data semantics (consider, e.g. the differences between Definition 2 and Definition 4 in Section 3), as well as in the definition of algebraic operators (consider, e.g., the adoption of the "*interprVTa*" function in the definitions of Cartesian product and of difference). Notably, there is only one other relational algebra coping with 'now', the one we devised in [10] for the NULL, MIN, MAX and POINT approaches, and the new algebra proposed in this paper is radically different from it, due to its treatment of different types of latencies. Specifically, the algebra in [10] is based on the notion of "*binding*" of the value of 'now' to the current time (taken as the reference time). On the other hand, no binding is used in the newly proposed algebra, to support the possibility of coping also with the case in which latency is unknown (only the case of latency equal to zero was considered –although implicitly- in the previous algebra). The main "practical" differences between our approach and the above ones are graphically highlighted in Figure 2. In the figure, we consider a now-related fact and we compare its valid time (transaction time is not shown, for the sake of readability) at different reference times in the different approaches. Notably, MAX, MIN, NUL, and POINT approaches only support the case in which latency is zero. Clifford et al. also supports different latencies (e.g., latency -2 in the figure), while our approach supports latency zero, latency different from zero and also *unknown* latency, which is not supported by any other approach. Indeed, we stress that unknown latency constitutes the general case. Notably, our approach, differently from the others, also supports a future *bound* for 'now'.

nacy in general (see [5]) and in our future work we aim at integrating both results.

The other main innovative contribution of our ap-

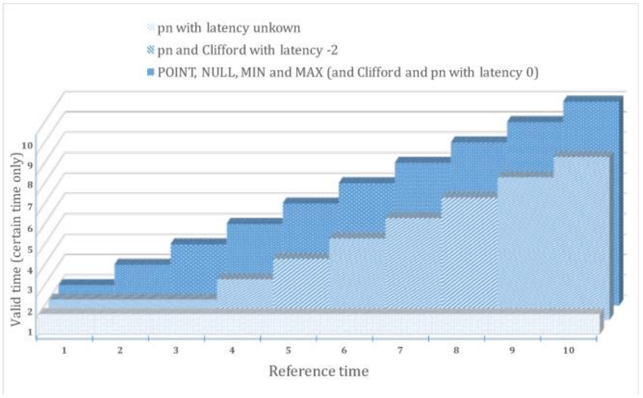

Figure 2. Graphical representation for the fact "John stays in ICU from 1 to NOW. The fact is asserted at 1". The certain valid times of the now-related example are represented at different reference times in the various approaches.

proach is that we provide the only approach to 'now' in TDB that homogeneously takes into consideration all the different issues (1) - (5) mentioned in the introduction. Clifford et al. only focused on (1) (and partly on (2)), NULL, MIN, MAX and POINT approaches on (2), (4) and (5) only, though in a recent work [10] we have provided an algebra for the NULL, MIN, MAX and POINT approaches, covering also issue (3).

Before ending, it is worth mentioning that also some commercial systems are starting to provide temporal support, based on TSQL2 seminal approach. For instance, Oracle database since version 12c supports valid time. However, it does not explicitly cope with now-related data, but it only allows users to set end and start valid times to NULL in order to represent facts valid at all time values [28].

7 Conclusions

Now-related temporal data play an important role in many applications. In the area of temporal relational databases, several approaches have faced in isolation different issues concerning now-related data (see the introductory section and Section 6). In this work, we first propose a comprehensive approach, starting from the semantics, then moving towards a compact 1NF representation, and finally providing an experimental evaluation, considering both data model and algebraic and manipulation operators. The main advances with respect to the current approaches in literature are:

(i) we first propose an integrated approach considering the different aspects (the approach is "deeply" integrated, since we also prove the correctness of our representation with respect to the semantics),

(ii) we extend current approaches considering new phenomena. In particular, we also cope with cases in which there is a future bound for the validity of now-related tuples and, above all, with cases in which the latency of updates is unknown.

(iii) we experimentally demonstrate that our representational approach does not add any significant overhead to cope with now-relative data (roughly behaving like the "ideal" approach).

References

[1] L. Liu and M. T. Özsu, Eds., *Encyclopedia of database systems*. Springer, 2009.

[2] 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.

[3] TSQL2 Language Design Committee, *The TSQL2 temporal query language*. Kluwer, 1995.

[4] C. S. Jensen and R. T. Snodgrass, "Semantics of Time-Varying Information," *Inf. Syst.*, vol. 21, pp. 311–352, 1996.

[5] 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.

[6] 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.

[7] C. E. Dyreson, C. S. Jensen, and R. T. Snodgrass, "Now in Temporal Databases," in *Encyclopedia of Database Systems*, 2009, pp. 1920–1924.

[8] K. Torp, C. S. Jensen, and M. H. Böhlen, "Layered Temporal DBMS: Concepts and Techniques," in *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, 1997, pp. 371–380.

[9] 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.

[10] 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.

[11] C. S. Jensen and R. T. Snodgrass, "Semantics of Time-Varying Information," *Inf. Syst.*, vol. 21, pp. 311–352, 1996.

[12] C. S. Jensen and R. Snodgrass, "Temporal specialization and generalization," *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 6, pp. 954–974, Dec. 1994.

[13] J. Chomicki and D. Toman, "Temporal Logic in Information Systems," in *Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995)*, 1998, pp. 31–70.

[14] J. Dunn, S. Davey, A. Descour, and R. T. Snodgrass, "Sequenced subset operators: definition and implementation," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 81–92.

[15] C. Dyreson, "Temporal Indeterminacy," in *Encyclopedia of Database Systems*, L. Liu and M. T. Ozsu, Eds. Boston, MA: Springer US, 2009, pp. 2973–2976.

[16] K. Torp, C. S. Jensen, and R. T. Snodgrass, "Modification semantics in now-relative databases," *Inf. Syst.*, vol. 29, no. 8, pp. 653–683, Dec. 2004.

[17] T. Johnston and R. Weis, *Managing time in relational databases: how to design, update and query temporal data*. Amsterdam ; Boston: Morgan Kaufmann/Elsevier, 2010.

[18] 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.

[19] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[20] 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.

[21] L. E. McKenzie Jr. and R. T. Snodgrass, "Evaluation of Relational Algebras Incorporating the Time Dimension in Databases," *ACM Comput Surv*, vol. 23, no. 4, pp. 501–543, Dec. 1991.

[22] C. E. Dyreson, "Temporal coalescing with now granularity, and incomplete information," 2003, p. 169.

[23] D. Lomet, M. Hong, R. Nehme, and R. Zhang, "Transaction Time Indexing with Version Compression," *Proc VLDB Endow*, vol. 1, no. 1, pp. 870–881, Aug. 2008.

[24] L.-V. Nguyen-Dinh, W. Aref, and M. Mokbel, "Spatio-Temporal Access Methods: Part 2 (2003 - 2010)," *Cyber Cent. Publ.*, Jan. 2010.

[25] S. Saltenis and C. S. Jensen, "Indexing of now-relative spatio-bitemporal data," *VLDB J. Int. J. Very Large Data Bases*, vol. 11, no. 1, pp. 1–16, Aug. 2002.

[26] C. S. Jensen and D. B. Lomet, "Transaction Timestamping in (Temporal) Databases," in *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, 2001, pp. 441–450.

[27] K. Torp, C. S. Jensen, and R. T. Snodgrass, "Effective timestamping in databases," *VLDB J. Int. J. Very Large Data Bases*, vol. 8, no. 3–4, pp. 267–288, Feb. 2000.

[28] "CREATE TABLE," in *Oracle database SQL Language Reference 12c Release 1*, Oracle, 2015, p. 452.

Luca Anselma received the PhD degree in computer science from Università degli Studi di Torino in 2006 and the Master's Degree in computer science in the same university in 2002. He is an assistant professor of computer science at the Università di Torino, Italy. His main research interests include the areas of temporal reasoning, temporal databases and medical informatics. He is the author of more than 40 papers in international journals, books, and international refereed conferences.

Luca Piovesan received the PhD degree in computer science from Università degli Studi di Torino in 2016. He obtained the Bachelor Degree in Computer Science in 2010 and the Master's Degree in Computer Science 2012 at Università degli Studi di Torino. His research interests are artificial intelligence (knowledge representation, planning, decision support systems, temporal reasoning), medical informatics and temporal databases.

Abdul Sattar is founding Director of the Institute for Integrated and Intelligent Systems (IIIS), a research centre of excellence at Griffith University established in 2003. He has been an academic staff member at Griffith University since February 1992 as a lecturer (1992-95), senior lecturer (1996-99), and professor (2000-present) within the School of Information and Communication Technology. He holds a BSc (Physics, Chemistry and Mathematics) and an MSc (Physics) from the University of Rajasthan, India, an MPhil in Computer and Systems Sciences from the Jawaharlal Nehru University, India, and an MMath in Computer Science from the University of Waterloo, Canada, and a PhD in Computer Science (with specialization in Artificial Intelligence) from the University of Alberta, Canada. His research interests include knowledge representation and reasoning, constraint satisfaction, rational agents, propositional Satisfiability, temporal reasoning, temporal databases, and bio-informatics. He has supervised over 30 PhD graduates, and published over 200 technical papers in refereed conferences and journals in the field.

Bela Stantic is a Deputy Head of School of Information and Communication Technology within the Griffith University. His area of research is efficient management of complex data structures including Big Data, Spatio-temporal and High dimensional data. He successfully applied his research interdisciplinary and published more than 90 peer-reviewed conference and journal papers. He presented many invited and Keynotes talks and served on Program Committees of more than 100 conferences and was/is doing the editorial duties of many Journals.

Paolo Terenziani received his Laurea degree in 1987 and his PhD in computer science in 1993 from Università di Torino, Italy. He is full professor in computer science with DiSIT, Institute of Computer Science, Università del Piemonte Orientale "Amedeo Avogadro", Alessandria, Italy. His research interests include artificial intelligence (knowledge representation and temporal reasoning), databases and medical informatics. He has published more than 150 papers on these topics in refereed journals and conference proceedings.