Frame Time and Cardinality Indeterminacy in Temporal Relational Databases

Paolo Terenziani
Institute of Computer Science, DiSIT, Universita' del Piemonte Orientale, Viale teresa Michel 11, Alessandria, Italy

Keywords: Temporal Relational Databases, Data Model, Algebra, Temporal Indeterminacy, Reducibility.

Abstract: Time is pervasive of reality, and many relational database approaches have been developed to cope with it. However, in practical applications, *temporal indeterminacy* about the exact time of occurrence of facts and, possibly, about the number of occurrences, may arise. Coping with such phenomena requires an in-depth extension of current techniques. In this paper, we have introduced a new *data model*, and new definitions of *relational algebraic operators* coping with the above issues, and we have studied operator *reducibility*.

1 INTRODUCTION

Time is pervasive of our way of looking at reality. As a consequence, time is often modeled in databases. However, it is commonly agreed by the scientific community that time has a special status with respect to the other data, so that its treatment within a (relational) database context requires specific attention and dedicated techniques. <<*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 (Snodgrass, 1999), Section "Preface", Subsection: "A paradigm shift", page XVIII).

As a consequence, a plethora of dedicated approaches have been developed by the scientific community (see the cumulative bibliography in (Wu et al., 1997)). Most of the approaches in the literature focus on the case in which the *exact valid time* (Snodgrass, 1995) of facts is known. However, in real world applications, it is often the case that such an *exact* time is not known. In such a case, *temporal indeterminacy* (Dyreson, 2009) occurs. Temporal indeterminacy has various possible sources, including *scale*, *dating techniques*, *future planning*, *unknown or imprecise event times*, *clock measurements* (this list is not exhaustive, and is taken from TSQL2 book (Snodgrass, 1995; page 327)). Given its relevance, "support for temporal indeterminacy" was already one of the eight explicit goals of the data types in TSQL2 consensus approach: <<… *many applications require the storage of such "don't know exactly when" information*>> (Snodgrass, 1995). However, despite the importance of this phenomenon, relatively few approaches in the temporal relational database (TDB henceforth) area have faced it (consider, e.g., the approaches discussed in the survey in (Dyreson, 2009) and the recent approach in (Anselma et al., 2010; 2013)) and no general solution has been found yet (see the discussion in the Related Work Section).

Indeed, it is worth stressing that "temporal indeterminacy" regards cases in which *there is* indeed some form of information about the valid time of facts, but such pieces of information are *approximate*, in that they do not exactly locate facts on the timeline. Therefore, different forms of temporal indeterminacy can be faced, depending on the form of approximate temporal information one wants to model (see, e.g., the family of different temporal database approaches identified in (Anselma et al., 2010; 2013)). A very common one is what we call "*frame time*" temporal indeterminacy: the exact time of the occurrence of a fact is not known, and can only be approximated by a frame of time which contains it. For instance, in "*On October 10th 2011 John watched Cars 2 at the cimema*" October 10th is clearly not the exact time of occurrence of the fact that John watched a film: John watched the film at some (not known/specified) time interval within the *frame time* October 10th.

Frame time temporal indeterminacy is frequent and relevant. For instance, it naturally rises in all cases in which time is not perfectly monitored. For instance, when a person is asked to elicit her activities, and the time when she performed them, she usually provides only frame time approximations. It is actually quite unusual that a person can provide accurate temporal descriptions such as "*On October 10th I watched Cars 2 (*or: *I had headache;* or: *I did his homework;* or: *I had the meeting) from 17:12 to 18:44*". Even the usual TDB example about employee promotions is indeed a case of temporal indeterminacy: we rarely have the exact time, so that it is usually approximated by the day in which the promotion occurred. It is worth noticing that the treatment of temporal granularities is a related phenomenon: for instance, whenever a data set expressed (with exact times of occurrences) at a given granularity is converted to a different, coarser granularity, frame time indeterminacy arises. For instance, switching the fact "*On October 10th John watched Cars 2 from 17:12 to 18:44*" to the granularity of days naturally rises temporal indeterminacy: "*On October 10th John watched Cars 2*". Additionally, also the case in which one knows the occurrence of a fact, but has no information at all concerning its time of occurrence can be seen as a degenerate case of frame time indeterminacy: in such a case the frame of time stretches from the origin of time (for the database) to the latest possible time.

Despite the practical relevance of the frame time indeterminacy, and the fact that several approaches have faced it in other areas of Computer Science, such as, e.g., Artificial Intelligence, to the best of our knowledge no TDB approach has directly faced this phenomenon, providing both a 1NF data model and an algebra coping with it (see the Related Work Section). In this paper, we aim at overcoming such a limitation of the current literature. We provide (i) a *data model* to represent frame time indeterminacy, and (ii) a *temporal relational algebra* to query it, and (iii) we prove its *reducibility* to the conventional non-temporal algebra. Indeed, as in most DB approaches we know, we regard the development of a query language (we choose to operate at the algebraic level) as an essential contribution, and a fundamental desiderata for our approach (as well as for all the DB approaches we know) is that the data model must be expressive enough in order to represent the results of the queries (technically speaking, we aim at achieving the *closure* of the *relational algebra* with respect to the *data model*). As we will see in more detail in Section 2, such a goal leads to important implications about the data model we propose. Indeed, to be expressive enough to represent the output of the application of relational algebraic operators, our data model must also include the possibility of coping with an additional form of indeterminacy: the indeterminacy about the *cardinality of the occurrences* of facts within their frame time. As an example, consider "*On October 10th 2011, John had two or three headache attacks*", where *October 10th* is the frame of time containing the intervals of occurrences of John's headache, and the number of occurrences is only approximated by a *minimum* and *maximum bound*. The treatment of such a form of indeterminacy concerning the number of occurrences of facts is an additional contribution of our work, and has never been provided, to the best of our knowledge, by any approach in the literature.

The paper is organized as follows. In Section 2 we discuss the key problems and challenges we had to face, and informally sketch our solutions. Section 3 formally introduces our data model, and Section 4 describes our extended relational algebra. Section 5 presents related works, and Section 6 contains conclusions. The Appendix contains proofs.

2 PROBLEMS AND SOLUTIONS

The treatment of temporal indeterminacy involves in-depth extensions to the standard TDB approaches. To substantiate this claim, we introduce an example.

Example 1. *On September 10th 2011, John had an headache episode.* Frame time temporal indeterminacy occurs in Example 1, since the exact temporal location of the headache episode is not provided: we only know that the episode occurred in a time interval contained (properly or not) in September 10th 2011. First, let us suppose to deal with Example 1 using a typical TDB representation. As a representative, let us choose a TSQL2 (Snodgrass, 1995) representation, shown in Table 1. In the example, we use the granularity of minutes.

Table 1: Relation PAT_HISTORY: an approximate representation of Example 1 in TSQL2.

Patient	Symptom	T_{start}	T_{end}
John	headache	10/9/2011 0:00	10/9/2011 23:59

At a first glance this representation looks reasonable. However, we stress that, to cope with the exact content of Example 1, we cannot interpret the relation PAT_HISTORY above using the usual semantics adopted by TSQL2 and by most TDB approaches (such a semantics has been specified by the BCDM model (Jensen & Snodgrass, 1996)). In such a semantics, the so-called *snapshot* semantics, the meaning of the tuple in Table 1 would be the following:

$$\forall t\ 10/9/2011\ 0{:}00 \leq t \leq 10/9/2011\ 23{:}59 \quad \text{Holds\_at}(\text{<John,headache>},t) \tag{1}$$

where $Holds\_at(<a_1,\ldots,a_n>,t_i)$ is a predicate (that we have imported from (Galton, 1990)) stating that the fact $<a_1,\ldots,a_n>$ occurs (is true) at the time t_i. In other word, in TSQL2, the tuple in Table 1 would mean that John had headache continuously, in all the time granules (temporal *snapshots*) in September 10th. On the other hand, the intended semantics of Example 1 is different, as shown below:

$$\exists t_{start}\ \exists t_{end}\ (t_{start} \leq t_{end})\ \forall t\ t_{star} \leq t \leq t_{end} \quad \text{Holds\_at}(\text{<John,headache>},t) \tag{2}$$

Intuitively, Example 1 means that there is an interval of time (starting at t_{start} and ending at t_{end}) during September 10th in which John had (continuously) headache. Adopting such a *new* semantics has a deep impact in the definition of relational algebraic operators. Indeed, since we demand that a data model must be expressive enough to cope with the results of the application of algebraic operators, and since we want to enforce the above semantics, the TSQL2 data model is inadequate here. As a simple example, let us consider intersection. Let us suppose to have another relation, PAT_HISTORY', which is identical to PAT_HISTORY in Table 1, and to perform the intersection PAT_HISTORY $\cap$ PAT_HISTORY'. Trivially, the non-temporal component of the two tuples is equal. But what about the intersection of their temporal components? Remember that the underlying semantics of our representation is not (1), but is the one shown at point (2) above. Thus, the tuple <John, headache| 10/9/2011 0:00, 10/9/2011 23:59> in PAT_HYSTORY states that there is a (convex) time interval in which John had headache, which is located somewhere in September 10th, and the tuple in PAT_HISTORY' behaves in the same way. There is no support for the conclusion that the time intervals of the two tuples are the same, or temporally intersect. Indeed, they may intersect (or even be the same), but may also be disjoint! Stated in other words, the two relations may denote two different episodes of John's headache, or the same episode. Due to the intrinsic indeterminacy of the inputs, both cases are possible. As a consequence, the output of intersection may be empty, in case the two episodes are disjoint, or contain an episode occurring on September 10th otherwise.

However, as stressed already in the introduction, we aim at providing a data model in which the output of algebraic queries can be expressed. We must thus move towards a different representation with respect to the one in Table 1, since it cannot capture the indeterminacy about the output of intersection described above. Our idea is simple, and starts from the consideration that the above indeterminacy can be interpreted as an indeterminacy about the number of occurrences (zero or one, in the example) of the described episode. Thus, we propose to extend the data model in order to explicitly model the *minimum* and *maximum number of occurrences* of facts.

It is worth emphasizing that such a number of occurrences cannot be coped with as a "standard" numeric attribute, to be managed directly by users/developers. We will show soon that relational algebraic operators have to be carefully re-defined to correctly deal with such numbers. Such a definition must be provided once-and-for-all, and cannot be demanded to users/developers (analogous motivations have been provided in Section 1 of the TSQL2 book (Snodgrass, 1995) for the specialized treatment of *valid time* in temporal relational databases).

Also, the cases in which the exact number of occurrences of facts is known (see, e.g., example 1 in the introduction) can be easily modeled, and constitute a specific case (in which the minimum and maximum cardinality are set to be equal) of our general model. Indeed, example 1 above shows that, even in case the exact input cardinalities are known, the cardinalities obtained by the application of relational operators may only be bounded by a minimum and a maximum value. It is worth stressing that such a behavior is not due to our choice of the data model, but is an *intrinsic feature* of the phenomena we want to model.

Finally, we stress that our approach is even more expressive with respect to the initial requirements discussed in the introduction, since it copes with basic relations (i.e., primitive relations, not obtained as a result of any query) both in the case when the number of occurrences of facts is known, and in the case when it can only be approximated by a minimum and maximum bound. For instance, it can cope with Example 2 in the following.

Example 2. *On September 10th 2011, John had two or three headache episodes.* In the rest of the paper, the above solutions are detailed.

3 DATA MODEL

In our data model, tuples are associated with valid time (*transaction time* (Snodgrass, 1995) is not considered in this paper). The timeline is partitioned into granules of a chosen basic granularity. As is BCDM (Jensen and Snodgrass, 1996) and TSQL2 (Snodgrass, 1995), our time domain is totally ordered and is isomorphic to the subsets of the domain of natural numbers. The domain of valid times D_{VT} is given as a set $D_{VT}=\{t_1,t_2,\ldots,t_k\}$ of granules. The number of repetitions of a fact in a frame time is encoded by two cardinality attributes N and M, defined on the domains of natural numbers and of positive natural numbers respectively, with the constraint that the minimal cardinality is less or equal that the maximum cardinality. We propose a 1-NF representation of facts. The schema of an indeterminate temporal relation $R=(A_1,\ldots,A_n|N,M,T_{start},T_{end})$ consists of an arbitrary number of non-temporal attributes $A_1,\ldots,A_n$, encoding some fact, of a minimal cardinality attribute N, of a maximal cardinality attribute M, and of two timestamp attributes T_{start} and T_{end}, with domain D_{VT}. Thus, a tuple $x=<a_1,\ldots,a_n|n_1,n_2,t_1,t_2>$ (where $n_1 \leq n_2$, $n_2>0$, and $t_1 \leq t_2$) in a relation r(R) on the schema R consists of a n-tuple of values for the non-temporal attributes, associated with a minimum cardinality n_1, a maximum cardinality n_2, and two timestamps $t_1,t_2 \in D_{VT}$, and represents the fact that *there are between n_1 and n_2 occurrences of the fact $a_1,\ldots,a_n$ during the time interval starting at t_1 and ending at t_2*. For instance, Table 2 shows the relation modeling Example 2 in our data model.

Table 2: Relation PAT_HISTORY_INDET: representation of Example 2 in our model.

Pat.	symptom	N	M	T_{start}	T_{end}
John	headache	2	3	10/9/2011 0:00	10/9/2011 23:59

Notation 1. Given a tuple x defined on the schema $R=(A_1,\ldots,A_n|N,M,T_{start},T_{end})$, we denote by A the set of attributes $A_1,\ldots,A_n$. Then x[A] denotes the values in x of the attributes in A, x[T] denotes the time interval of x, $x[T_{start}]$ and $x[T_{end}]$ its starting and ending time respectively, and x[N] and x[M] denote the minimum and maximum cardinality respectively.

Our data model is a consistent extension of the TSQL2 data model, as specified by Property 1 below. This fact grants that we can cope with the same content than TSQL2 (and BCDM) valid time relations, which are, indeed, a restriction of our relations, in which both minimum and maximum cardinalities of tuples must be set to the value 'one'.

Property 1. TSQL2 valid-time relations can be modeled by indeterminate temporal relations in our approach.

4 RELATIONAL ALGEBRA

Codd designated as complete any query language that is as expressive as his set of five relational algebraic operators: relational union ($\cup$),difference ($-$), selection (σ_P), projection (π_A), and Cartesian Product ($\times$) (Codd, 1972). We propose an extension of Codd's operators to query the data model we introduced in Section 3. Several temporal extensions have been provided to Codd's operators in the TDB literature (McKenzie et al., 1991); (Snodgrass, 1995). In many cases, such extensions behave as standard non-temporal operators on the non-temporal attributes, and may involve the application of set operators on the temporal attributes. This approach ensures that the temporal algebraic operators are a consistent extension of Codd's operators and are *reducible* to them when the temporal dimension is removed. For instance, in TSQL2, temporal Cartesian Product involves pairwise concatenation of the values of non-temporal attributes and pairwise intersection of their temporal values. Analogously, in TSQL2, relational difference, union, and projection behave in a standard way on non-temporal attributes, and difference performs the difference on the temporal component of *value-equivalent* tuples.

4.1 Relational Algebra for Frame Time Indeterminacy

We ground our approach on such a "consensus" background, extending the algebraic operators to cope with the new implicit attributes.

Def. 1: *Temporal Algebraic Operators*. Let r and s denote relations having the proper schema.

$\mathbf{r \cup^I s} = \{ < v|n,m,t_s,t_e> |$
$\exists n_1,m_1,t1_s,t1_e\ (<v|n_1,\ m_1,\ t1_s,t1_e> \in r \wedge n=n_1 \wedge m=m_1 \wedge t_s= t1_s \wedge t_e= t1_e) \vee$
$\exists n_2,m_2,t2_s,t2_e\ (<v|n_2,m_2,\ t2_s,t2_e> \in s \wedge n=n_2 \wedge m=m_2 \wedge t_s= t2_s \wedge t_e= t2_e) \}$

$\mathbf{r -^I s} = \{ <v|\ n,m,t_s,t_e> |$
$(\exists n_1,m_1,t1_s,t1_e\ (<v|n_1,m_1,t1_s,t1_e> \in r \wedge$
$\neg \exists n_2,m_2,t2_s,t2_e\ (<v|n_2,m_2,t2_s,t2_e> \in s \wedge$
$[t1_s,t1_e] \cap [t2_s,t2_e] \neq \emptyset) \wedge$
$n=n_1 \wedge m=m_1 \wedge t_s= t1_s \wedge t_e= t1_e)\ ) \vee$
$(\exists n_1,m_1,t1_s,t1_e\ (<v|n_1,m_1,t1_s,t1_e> \in r \wedge$
$\exists n_2,m_2,t2_s,t2_e\ (<v|n_2,m_2,\ t2_s,t2_e\ > \in s \wedge$
$[t1_s,t1_e] \cap [t2_s,t2_e] \neq \emptyset) \wedge$
$n=0 \wedge m=m_1 \wedge t_s= t1_s \wedge t_e= t1_e)\}$

$\mathbf{r \times^I s} = \{ <v_1 \cdot v_2|n,m,t_s,t_e> |$
$\exists n_1,m_1,t1_s,t1_e\ (<v_1|n_1,m_1,t1_s,t1_e> \in r \wedge$
$\exists n_2,m_2,t2_s,t2_e\ (<v_2|n_2,m_2,t2_s,t2_e> \in s) \wedge$
$[t_s,t_e]= [t1_s,t1_e] \cap [t2_s,t2_e] \wedge [t_s,t_e] \neq \emptyset \wedge n=0 \wedge m=\min(m_1,m_2))\}$

$\boldsymbol{\pi^I_A}\ \mathbf{(r)} = \{ <v|n,m,t_s,t_e> |$
$\exists v_1,n_1,m_1,t1_s,t1_e\ (<v_1\ |n_1,\ m_1,t1_s,t1_e\ > \in r \wedge$
$v = \pi_A(v_1) \wedge t_s= t1_s \wedge t_e= t1_e \wedge n=n_1 \wedge m=m_1\}$

$\boldsymbol{\sigma^I_P}\ \mathbf{(r)} = \{(<v|n_1,m_1,t_s,t_e> | (<v|n_1,\ m_1,t_s,t_e> \in r \wedge P(v)\}$

For the sake of brevity, in the definition of difference $[t_s,t_e]=[t1_s,t1_e]-[t2_s,t2_e] \wedge [t_s,t_e] \neq \emptyset$ may denote one or two (in case $[t2_s,t2_e]$ is properly contained into $[t1_s,t1_e]$) time intervals.

As motivated above, our algebraic relational operators operate in the standard way on the non-temporal attributes. As in TSQL2, union, projection and selection operate in a standard way. On the other hand, Cartesian Product involves temporal intersection (as demanded by the *snapshot* semantics; see, e.g., the BCDM model). The starting (t_s) and ending (t_e) times of the resulting tuples are obtained through the intersection of the intervals on the two input tuples (i.e., $[t_s,t_e] =[t1_s,t1_e] \cap [t2_s,t2_e]$), and tuples are present in the output only when such an intersection exists (i.e., when $[t_s,t_e] \neq \emptyset$). The minimum output cardinality is zero, to represent the fact that it is possible that there is no intersection between the valid times of the input tuples. The maximum cardinality is the minimum of the input maximum cardinalities. This models the fact that, supposing, e.g., that $\min(m_1,m_2)=m_1$, all the occurrences of the first input tuple intersect with one occurrence of the second input tuple (or viceversa, if $\min(m_1,m_2)=m_2$). For instance, given a relation PAT_2 having the same schema of PAT_HISTORY_INDET, and containing the tuple <Ann, headache|1,1, 10/9/2011 12:01, 10/9/2011 23:59> (i.e., Ann had a headache episode on September 10th, after 12 o'clock), we have PAT_HISTORY_INDET $\times^I$ PAT_2= {<John, headache, Ann, headache |0,1, 10/9/2011 12:01, 10/9/2011 23:59>}.

In the operation of difference two cases must be distinguished. In case we perform $r -^I s$ and we have one tuple t=<v|n_1,m_1,$t1_s$,$t1_e$> in r which has no *value-equivalent* tuple in s (a temporal tuple t' is *value-equivalent* to t if it is equal to t as regard its non-temporal component (Snodgrass, 1995)), or such that value equivalent tuples in s hold in time intervals that do not intersect $[t1_s,t1_e]$, t must be reported in output unchanged. On the other hand, if a *value-equivalent* tuple t'=<v|n_2,m_2,$t2_s$,$t2_e$> exists in s, such that $[t1_s,t1_e] \cap [t2_s,t2_e] \neq \emptyset$, then the minimum number of occurrences is 0 (to represent the fact that all the occurrences in t may be fully contained into one or more occurrences in t'), and the maximum number of occurrences is m_1 (to represent the case when none of the occurrences in t is contained into the occurrences in t').

For instance, given a relation PAT_3 having the same schema of PAT_HISTORY_INDET, and containing the tuple <John, headache|1,4, 9/9/2011 0:00, 10/9/2011 23:59>, we have PAT_HISTORY_INDET $-^I$ PAT_3= {<John, headache, |0,3, 10/9/2011 0:00, 10/9/2011 23:59>}.

Finally, it is worth comparing the computational complexity of our algebraic operators with that of temporal operators already in the literature, e.g., with TSQL2's ones. As discussed and motivated at the beginning of Section 4, we follow a "consensus" approach in the TDB literature (followed also, e.g., by TSQL2). Thus, our definitions of temporal union, temporal projection, nontemporal selection and temporal selection are similar to TSQL2's ones (performing intersection and difference between valid times of tuples), except for the fact that, to cope with temporal indeterminacy, we also operate on minimum and maximum cardinality. As a consequence, only a constant (and limited) overhead is added with respect to TSQL2 (and many other TDB approaches).

4.2 Reducibility of the Algebra

Reducibility is fundamental, to grant that the semantics of extended operators reduces to that of non-temporal algebraic operators when time is disregarded (McKenzie et al., 1991); (Snodgrass, 1995). To prove it for our algebra, we define a family of slicing operators:

$\pi^I_{\varphi,t}(r)=\{z \mid \exists x\in r\ z[A]=x[A] \wedge \varphi(x[N],x[M]) \wedge x[T_{start}]\leq t\leq x[T_{end}]\}$

where $t\in D_{VT}$ is a temporal granule, and φ is a predicate on the minimum and maximum cardinalities (e.g., φ: x[N]>0). The result of slicing is a *non-temporal* relation defined over the schema A. Different slicing operations can be obtained using different instantiations of the φ predicate. For instance:

φ: x[M]>0 requires that the maximum cardinality is at least one. This condition is always satisfied by definition, and, intuitively speaking, corresponds to the case in which the fact described by the tuple is 'possible' at time t.

φ: x[N]>0 requires that the minimum cardinality is at least one. Intuitively speaking, this condition correspond to the case in which at least one instance of the fact described by the tuple 'necessarily' occurred at time t.

Reducibility does not hold for any definition of φ. E.g., it does not hold for φ: x[N]>0, while it holds for φ: x[M]>0 (the proof is in the Appendix).

Property 2. Reducibility. Our algebra reduces to the standard (non-temporal) algebra, i.e., for each algebraic unary operator Op^I in our model, and indicating with *Op* the corresponding Codd operator, for each relation *r*, and for any granule *t*, and for φ*: x[M]>0*, the following holds (the analogous holds for binary operators):

$$\pi^I_{\varphi,t}(Op^I(r)) = Op(\pi^I_{\varphi,t}(r))$$

5 RELATED WORK

In many applications, the exact temporal location of facts cannot be determined, so that some form of temporal indeterminacy must be managed. As a consequence, many approaches to temporal indeterminacy have been devised, e.g., within the Artificial Intelligence (AI) field. Many different forms of temporal indeterminacy have been considered in AI, including qualitative and quantitative constraints between events (see, e.g., the survey (Allen, 1991)). As concerns specifically "*frame time*" indeterminacy, it can be coped with, e.g., in the STP framework (Dechter et al., 1991). In STP, *bounds on differences* of the form $c_1\leq X-Y\leq c_2$ are used in order to state that the distance between two points *Y* and *X* is between c_1 and c_2 time units. In STP, *frame time* indeterminacy can be represented. For instance, Example 1 can be represented as f(10/9/2011 0:00)$\leq JHA_s-X_0 \wedge JHA_e-X_0\leq$f(10/9/2011 23:59) $\wedge\ 0\leq JHA_e - JHA_s$, where JHA_s and JHA_e represent the starting and ending points of the episode of John's headache, and X_0 represents the chosen origin of time, and *f(d)* is a function that evaluates the number of time units occurring from X_0 to the timestamp *d*. In STP, constraint propagation mechanisms are provided in order to perform temporal reasoning of a set (conjunction) of constraints. Moving to the area of AI temporal logics, Galton (Galton, 1990), in his reified temporal logic, has introduced a specific predicate to model frame time indeterminacy. Specifically, the predicate *HOLDS-IN(f,t)*, where *f* represents a fact, and *t* is a time period, models the situation in which the fact *f* occurred somewhere in the time interval *t*.

On the other hand, when moving to the area of (relational) databases, the number of approaches coping with temporal indeterminacy becomes more restricted. A survey of TDB approaches to temporal indeterminacy has recently been provided in (Dyreson, 2009). In the earliest TDB work on temporal indeterminacy, an indeterminate instant was modeled with a set of possible chronons (Snodgrass, 1982). A fuzzy set approach was introduced by Dutta (1989). Gadia et al. have proposed a model to support value and temporal incompleteness (Gadia et al., 1992). In particular, Gadia et al. also cope with values that are known if they occurred, thus considering a limited form of indeterminacy about the number (zero or one) of occurrences.

Dyreson and Snodgrass (1998) and Dekhtyar et al., (2001) have proposed probabilistic approaches coping with different forms of temporal indeterminacy. Dekhtyar et al. introduce temporal probabilistic tuples to cope with data such as "*data tuple d is in relation r at some point of time in the interval* $[t_i,t_j]$ *with probability between p and p'* ". They also provide algebraic relational operators for their data model. However, they restrict their attention to facts that are instantaneous, while our approach also considers facts with duration. In Dyreson's and Snodgrass' paper (1998), valid-time indeterminacy is coped with by associating a period of indeterminacy with a tuple. A period of indeterminacy is a period between two indeterminate instants, each one consisting of a range of granules and of a probability distribution over it. Additionally, they impose the constraint that the ranges of granules defining the starting and ending points of a period cannot overlap, so that each tuple has a "necessary" period of existence. Even disregarding probabilities, one may notice that frame time indeterminacy cannot be coped with by periods of indeterminacy, since they require additional pieces of information about the maximum possible starting time and the minimum possible ending time, which are not available if only the frame time is known. Additionally, it is worth noticing that in (Dyreson and Snodgrass, 1998) no relational algebra is proposed (and, indeed, it is easy to show that their 'periods of indeterminacy' formalism cannot be closed with respect to the relational operator of difference).

On the other hand, Brusoni et al., (1999) have faced indeterminacy in the context of dealing with temporal constraints between tuples. In (Brusoni et al., 1999), bounds on differences are used in order to represent temporal constraints between tuples. The notion of *conditional interval* is introduced, to cope with the indeterminacy involved by temporal constraints in the relational context. Also, a relational algebra has been devised to cope with conditional intervals.

Recently, Anselma et al., (2010, 2013) have introduced and compared a family of algebraic approaches to cope with different forms of temporal indeterminacy. Each approach is characterized by the possibility of expressing some form of temporal indeterminacy in a compact way. Indeed, though some of the approaches in (Anselma et al., 2010; 2013) can model frame time indeterminacy, none of them can do that in 1NF and in a compact way (as we do in this paper): the only way they can cope with it is by explicitly listing all the possibilities (i.e., all the possible locations for all the time intervals contained in a frame time).

6 DISCUSSION AND CONCLUSIONS

In this paper, we consider the problems of (1) *frame time temporal indeterminacy*, and of (2) *indeterminacy* about the *number of occurrences of facts*. To the best of our knowledge, only the approaches in (Brusoni et al., 1999) and (Anselma et al., 2010; 2013) have proposed *both* a relational *data model* and *algebra* which might (although not directly) support frame time indeterminacy. However, the data models of both approaches are not in 1NF, and, more interestingly, both approaches do not support *cardinality indeterminacy*. Indeed, to the best of our knowledge, cardinality indeterminacy has not been faced by any approach in the Artificial Intelligence and Temporal Database areas.

To deal with both *frame time* and *cardinality indeterminacy*, we have introduced a new *data model*, an a new definition of *relational algebraic* operators, and we have studied their properties. In particular, we have proved that our data model is a *consistent extension* of the TSQL2 data model (Property 1; in turn, TSQL2 data model is a *consistent extension* of standard (non-temporal) data model) and that our algebra *reduces* to the standard algebra (Property 2). Such properties are important, since they are the basis to grant for the *implementabilty* and the *interoperability* of our approach. Specifically, the above properties guarantee that (1) our approach can be *implemented* on top of standard relational databases (or of TSQL2) as a support to cope with frame time indeterminacy, and (2) the *interoperability* of our approach with pre-existent TSQL2 and standard relational data.

We are currently developing a prototypical implementation of our approach. Experimental evaluations will follow, to analyse the overhead we added with respect to temporal database approaches not managing temporal and cardinality indeterminacy. However, we conjecture that such an overhead will be almost negligible, due to the reasons discussed at the end of section 4.1.

Finally, even if in this paper we have only discussed valid time, our approach also supports transaction time. Indeed no indeterminacy is possible on transaction time, so that we manage it as proposed in the "consensus" TSQL2 approach.

ACKNOWLEDGEMENTS

This research is supported by the Compagnia di San Paolo, GINSENG project.

REFERENCES

J. Allen. Time and time again: the many ways to represent time. *International Journal of Intelligent Systems* 6(4). pp. 341-355. 1991.

L. Anselma, P. Terenziani and R. T. Snodgrass. Valid-Time Indeterminacy in Temporal Relational Databases: A Family of Data Models. *Proc. TIME 2010*, pp. 139-145, 2010.

L. Anselma, P. Terenziani and R. T. Snodgrass. Valid-Time Indeterminacy in Temporal Relational Databases: Semantics and Representations. *IEEE*

TKDE, to appear (IEEE Digital Library, ISSN: 1041-4347).

Brusoni V., Console L., Terenziani P., Pernici B. Qualitative and Quantitative Temporal Constraints and Relational Databases: Theory, Architecture, and Applications. *IEEE Trans. On Knowledge and Data Engineering* 11(6), pp. 948-968, 1999.

E. F. Codd, *Relational Completeness of Data Base Sublanguages*. In: R. Rustin (ed.), Database Systems 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California, 1972.

R. Decther, I. Meiri, J. Pearl. Temporal Constraints Networks. *Artificial Intelligence* 49. pp. 61-95. 1991.

Dekhtyar A., Ross R., and V. S. Subrahmanian. Probabilistic temporal databases, I: algebra. *ACM Trans. On Database Systems 26*(1), pp. 41-95, 2001.

C. Dyreson. Temporal Indeterminacy. In L. Liu and M. Tamer Özsu (2009). Encyclopedia of Database Systems, Springer, DOI: 10.1007/978-0-387-39940-9, ISBN: 978-0-387-39940-9.

C. E. Dyreson and R. T. Snodgrass, “Supporting Valid-time Indeterminacy”, *ACM Transactions on Database Systems*, 23(1), pp. 1-57, 1998.

S. Dutta, “Generalized Events in Temporal Databases,” in Prcoc. *Fifth International Conference on Data Engineering*, pp. 118-126, 1989.

S. K. Gadia, S. S. Nair, and Y.-C. Poon. “Incomplete Information in Relational Temporal Databases,” in *Proc. of the International Conference on Very Large Databases*, Vancouver, Canada, pp. 395-406, 1992.

A. Galton. A Critical Examination of Allen’s Theory of Action and Time. *Artificial Intelligence* 42 (2-3), 159-188, 1990.

C. S. Jensen, R. T. Snodgrass, Semantics of Time-Varying Information, *Information Systems*, 21(4), 311–352, 1996.

L. E. McKenzie, R. T. Snodgrass: Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Comput. Surv. 23*(4): 501-543 (1991).

R. T. Snodgrass. Monitoring Distributed Systems: A Relational Approach. *PhD thesis*, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1982.

R. T. Snodgrass (Ed.), The TSQL2 Temporal Query Language. Kluwer 1995.

R.T. Snodgrass, Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann Publishers, Inc., San Francisco, July, 1999.

Y. Wu, S. Jajodia, X. Sean Wang: Temporal Database Bibliography Update. *Temporal Databases*, Dagstuhl: 338-366, 1997.

APPENDIX

Property 1. Proof. TSQL2 (valid-time) valid time relations contain tuples of the form $<a_1,\ldots,a_n|t_1,t_2>$. The same content can be mapped in a tuple $<a_1,\ldots,a_n|1,1,t_1,t_2>$ in our approach.

Property 2. Reducibility. Proof. We prove the property considering Cartesian Product (the proof of other operators is similar). Let r and s be event relations of schema $(A|N,M,T_{start},T_{end})$ and $(B|N,M,T_{start},T_{end})$ where A and B stand for the non-temporal attributes $\{A_1,\ldots,A_n\}$ and $\{B_1,\ldots,B_m\}$, let $\times$ be the standard (non-temporal) Cartesian Product and $\times^I$ be our temporal Cartesian Product, and $\pi^I_{\varphi,t}$ the slicing operator (defined in Section 4.2), with φ: $x[M]>0$. Then,

$$\pi^I_{\varphi,t}(r\times^I s) = \pi^I_{\varphi,t}(r) \times \pi^I_{\varphi,t}(s)$$

We show the equivalence by proving two inclusions separately, i.e., we prove (1) that if a tuple belongs to $\pi^I_{\varphi,t}(r\times^I s)$ it also belongs to $(\pi^I_{\varphi,t}(r) \times \pi^I_{\varphi,t}(s))$, and viceversa (2).

(1).Let $x'' \in \pi^I_{\varphi,t}(r\times^I s)$. Then, by the definition of $\pi^I_{\varphi,t}$, there exists a tuple $x'\in(r\times^I s)$ such that $x'=x''[A,B]$ and $t\in x'[T_{start},T_{end}]$ and $x'[M]>0$. By the definition of $\times^I$, there exist tuples $x_1\in r$ and $x_2\in s$ such that $x_1[A]=x'[A]$ and $x_2[B]=x'[B]$ and $t\in x_1[T_{start},T_{end}]$ and $t\in x_2[T_{start},T_{end}]$ (since $t\in x'[T_{start},T_{end}]$ and $x'[T_{start},T_{end}]= x_1[T_{start},T_{end}]\cap x_2[T_{start},T_{end}]$) and $x_1[M]>0$ and $x_2[M]>0$ (since $x'[M]=\min(x_1[M]=x_2[M])$ and $x'[M]\geq 0$). Then, by the definition of $\pi^I_{\varphi,t}$, there exists a tuple $x_1'\in\pi^I_{\varphi,t}(r)$ such that $x_1'[A]=x_1[A]=x'[A]$, and there exists a tuple $x_2'\in\pi^I_{\varphi,t}(s)$ such that $x_2'[B]=x_2[B]=x'[B]$. Thus, by the definition of $\times$, there exists $x_{12}''\in(\pi^I_{\varphi,t}(r) \times \pi^I_{\varphi,t}(s))$ such that $x_{12}''[A]=x_1'[A]=x_1[A]=x'[A]=x''[A]$, and $x_{12}''[B]=x_2'[B]=x_1[B]=x'[B]=x''[B]$.

(2).Assume $x''\in (\pi^I_{\varphi,t}(r) \times \pi^I_{\varphi,t}(s))$. Then, by definition of $\times$, there exist tuples $x_1'\in\pi^I_{\varphi,t}(r)$ and $x_2'\in\pi^I_{\varphi,t}(s)$ such that $x_1'[A]=x''[A]$ and $x_2'[B]=x''[B]$. By the definition of $\pi^I_{\varphi,t}$, there exists a tuple $x_1\in r$ such that $x_1[A]=x_1'[A]$, $t\in x_1[T_{start},T_{end}]$ and $x_1[M]>0$, and there exists a tuple $x_2\in s$ such that $x_2[B]=x_2'[B]$, $t\in x_2[T_{start},T_{end}]$ and $x_2[M]>0$. Then, by definition of $\times^I$, there is a tuple $x'\in(r\times^I s)$ such that $x'[A]=x_1[A]$, $x'[B]=x_2[B]$, $t\in x'[T_{start},T_{end}]$ and $x'[M]>0$. Thus, by definition of $\pi^I_{\varphi,t}$, there exists a tuple $x_{12}''\in\pi^I_{\varphi,t}(r\times^I s)$ such that $x_{12}''[A,B]=x'[A,B]$. By construction, $x_{12}''=x''$.