# Data Models with Multiple Temporal Dimensions: Completing the Picture

Carlo Combi and Angelo Montanari

Department of Mathematics and Computer Science, University of Udine
Via delle Scienze 206, 33100 Udine, Italy
{combi, montana}@dimi.uniud.it

**Abstract.** There is a widespread recognition that valid and transaction times are the fundamental temporal dimensions of any fact relative to a database. There are, however, temporal aspects of facts that cannot be naturally modeled by means of them. A remarkable limitation of valid and transaction times is that they do not allow one to distinguish between retroactive and delayed updates. A third temporal dimension, called event time, has been proposed in the literature, which makes it possible to model retroactive, on-time, and proactive updates.
In this paper, we first refine the notion of event time by showing that one event time does not suffice to model relevant phenomena, and then we introduce a further temporal dimension, that we called availability time, which can be viewed as the information system counterpart of the real-world event time. We conclude the paper by outlining current and future work directions. In the appendix, we first propose a taxonomy of temporal databases endowed with valid, transaction, event, and availability times, and then we discuss some issues related to the design of query languages supporting databases with multiple temporal dimensions.

## 1 Introduction

As claimed by Jensen and Snodgrass [14], the study of temporal databases is a significant research topic in the database community. The ambitious goal of temporal databases is to manage time-varying information in a systematic and efficient way. Hence, it is not surprising that temporal database research covers a variety of areas, ranging from the conceptual level, where a number of different temporal extensions of the basic atemporal Entity-Relationship model has been proposed (cf. [10]), to the physical one, where indexing techniques suitable for dealing with data encompassing multiple temporal dimensions have been developed (cf. [26]). The focus of this paper is on the association of times with facts, which is at the basis of temporal data management.

There is a widespread recognition that valid and transaction times are the fundamental temporal dimensions of any fact relative to a database [14]. The *valid time* of a fact is the time when the fact is true in the domain, while the *transaction time* of a fact is the time when the fact is current in the database. Hence, valid and transaction times capture the time-varying states of the domain and of the database, respectively. Pairing valid and transaction times, one can recover past states of both the domain and the database. For instance, one can retrieve the real-world evolution of a given data item as recorded in the database at a given time point as well as the database evolution of a data item at a given time point. Jensen and Snodgrass [12] systematically analyze the semantics of all admissible combinations of valid and transaction times.

There are, however, meaningful temporal aspects of facts that cannot be naturally modeled by means of valid and transaction times. A remarkable limitation of these two temporal dimensions is that they do not allow one to distinguish between retroactive and delayed updates [17]. Kim and Chakravarthy [16] propose a third temporal dimension, called *event time*, that makes it possible to model retroactive, on-time, and proactive updates, and show how the combined use of valid, transaction, and event times allows one to maintain different past states generated by retroactive and proactive updates as well as by error corrections and delayed updates.

In this paper, we first refine the notion of event time, by showing that one event time does not suffice to model relevant phenomena. To overcome its limitations, we associate with each fact both the occurrence time of the event that initiates its validity interval and the occurrence time of the event that terminates it. Then, we introduce a further temporal dimension, that we called *availability time*, which can be viewed as the information system counterpart of the real-world event time. The availability time of a fact is the time interval during which the fact is available to and believed correct by the information system. As it will be shown, such an interval does not necessarily coincide with the transaction time interval of the fact.

The rest of the paper is organized as follows. In Section 2, we recall the basic notions of valid, transaction, and event times, and the related temporal database taxonomies. Section 3 illustrates the proposed refinement of the event time. Section 4 introduces the new concept of availability time and discusses the benefits of its usage. Section 5 outlines current and future work directions. Throughout the paper, we use examples taken from a medical scenario, which will be adopted as the target application domain [2]. However, the proposed temporal dimensions are completely general and can be used in other application contexts as well. In the Appendix, we first propose a taxonomy of temporal databases provided with valid, transaction, event, and availability times, focusing on the classifications induced by the new temporal dimensions; then, we discuss some issues related to the design of a query language for relational databases with multiple temporal dimensions.

## 2 Background

### 2.1 The basic temporal dimensions: valid and transaction times

Valid and transaction times are widely recognized as the two basic temporal dimensions of temporal databases. Furthermore, there exists a consolidated terminology about temporal databases provided with valid and/or transaction times, as witnessed by the Consensus Glossary of Temporal Database Concepts [11]. Valid and transaction times are defined as follows.

**Definition 1.** *Valid time: the* valid time *(VT) of a fact is the time when the fact is true in the modeled reality.*

**Definition 2.** *Transaction time: the* transaction time *(TT) of a fact is the time when the fact is current in the database and may be retrieved.*

Valid time is usually provided by database users, while transaction time is system-generated and supplied. Valid and transaction times are orthogonal dimensions: each of them can be independently recorded or not and has specific properties [12].

To explain the meaning of these two basic temporal dimensions, we consider a simple example, taken from a general medical scenario.

*Example 1.* On August 10, 1998, the physician prescribes a bipuvac-based therapy from 10:00 to 14:00. Data about the therapy is entered into the database at 9:00. Due to the unexpected evolution of the patient state, the bipuvac infusion is stopped at 11:15 and replaced by a diazepam-based therapy from 11:25 to 14:00. The new facts are entered at 12:00. (Bipuvac and diazepam are drugs commonly used in anesthesia.)

This example can be modeled as in Table 1. Each tuple is timestamped by valid and transaction times (hereafter, without loss of generality, we will use a temporal relational data model based on the tuple timestamping approach [14]). Both valid and transaction times are represented by intervals, where the occurrence of the special symbol $\infty$ as the ending point of an interval means that the interval includes the current time (the ending point of an interval that includes the current time is also denoted by `NOW` or `u.c.`, for `until changed`, in the literature). There exists a variety of possible relationships between valid and transaction times of a given tuple: the starting point of the transaction time can precede the valid time (as in the first tuple of Table 1), the starting point of the transaction time can follow the valid time (as in the second tuple), the starting point of the transaction time can follow the starting point of the valid time and precede its ending point (as in the third tuple), and so on. Depending

| Drug | VT | TT |
|---|---|---|
| bipuvac | [98Aug10;10:00, 98Aug10;14:00) | [98Aug10;9:00, 98Aug10;12:00) |
| bipuvac | [98Aug10;10:00, 98Aug10;11:15) | [98Aug10;12:00, $\infty$) |
| diazepam | [98Aug10;11:25, 98Aug10;14:00) | [98Aug10;12:00, $\infty$) |

**Table 1.** Database instance of the patient therapies.

on the modeled application domain, valid and transaction times can be related to each other in different ways. Jensen and Snodgrass [12] propose several classifications (*specializations*, in the authors' terminology) of (bi)temporal relations, based on the relationships between valid and transaction times of timestamped facts. They take into account both the relationships that exist between valid and transaction times of any given tuple of a relation as well as the relationships that exist between the valid and/or transaction times of different tuples of the relation. As an example, on the basis of the relationships that exist between the starting and/or ending points of the valid and transaction times of any given tuple of a relation, Jensen and Snodgrass characterize the following classes of relations: retroactive, delayed retroactive, predictive, early predictive, retroactively bounded, strongly retroactively bounded, delayed strongly retroactively bounded, strongly predictively bounded, early strongly predictively bounded, strongly bounded, predictively bounded, general, degenerate, retroactively determined, predictively determined [12]. For instance, given the starting point $VT_s$ of the valid time and the starting point $TT_s$ of the transaction time, a relation is *retroactive* if, for each tuple of the relation, the relationship $VT_s \leq TT_s$ holds, while a relation is *delayed retroactive with bound* $\Delta t \geq 0$ if, for each tuple of the relation, the relationship $VT_s \leq TT_s - \Delta t$ holds. According to Jensen and Snodgrass' classification, the relation of Table 1, taken as a whole, is *general*, because the first tuple has $VT_s > TT_s$, while the second and the third ones have $VT_s < TT_s$; if we apply the same criterion to each single tuple of the relation, we can classify the first tuple as *predictive* and the remaining two tuples as *retroactive*[1].

### 2.2 Single vs. multiple valid times

A problem which has been debated to some extent in the literature is whether a fact can always be associated with a single valid time or there exist situations in which multiple valid times are needed [13, 18]. Kokkotos and his colleagues [18] and Jensen and Snodgrass [13] provide some examples of facts which are apparently endowed with several valid times. The proposed solutions, however, are completely different: while Kokkotos and his colleagues propose the adoption of many "labeled" valid times, Jensen and Snodgrass conclude their analysis by asserting that one valid time is enough and the problem is that of properly identifying, at the conceptual level, what constitutes a fact. If a fact has many valid times, probably we are merging together different facts, associated with different (correlated) valid times. To better explain the problem of dealing with multiple valid times and the solution proposed by Jensen and Snodgrass [13], we discuss the following example[2].

*Example 2.* A patient, after a traumatic brain injury, must undergo a rehabilitation therapy, composed of two treatments: the first one "intensive", the second one "for maintenance". The family doctor of the patient prescribes the therapy; the prescription has a validity period, after which the patient cannot use the prescription anymore. At the hospital, rehabilitation treatments are structured in sessions, where several patients are followed by specialized people. The patient

[1] Notice that different classifications of the same tuple can be obtained by comparing different endpoints of its valid and/or transaction times.

[2] Example 2 is a reformulation in the medical domain of the example proposed by Kokkotos and his colleagues [18] concerning airline tickets.

has to fix the dates for these sessions; scheduled therapies can be changed by the hospital staff and/or by the patient for organizational/technical problems.

In the previous example, there are several times related to the rehabilitation therapy: (i) the (possibly several) scheduled times of the treatments, (ii) the actual times of the treatments, (iii) the validity of the prescription, and finally (iv) the time at which data is stored into the database. Table 2 depicts a possible state for the database, where all data items belong to a single table. Apparently, all these times are, in some way, valid times related to treatment data (or to parts of it). Kokkotos and his colleagues [18] argue that, in real-world situations like the previous one, the choice of keeping all the times related to a single fact in a single table is the best one; they propose to model all the time dimensions, but the transaction time, as "labeled" valid times. In the previous example, the attributes *S_I_TrD*, *S_M_TrD*, *A_I_TrD*, *A_M_TrD*, and *P_Val* of Table 2 would be labeled valid times representing the scheduled date of the intensive treatment, the scheduled date of the maintenance treatment, the actual date of the intensive treatment, the actual date of the maintenance treatment, and the validity of the prescription, respectively.

PAT_THERAPY

| Pr_Id | P_Val | S_I_Tr | S_I_TrD | S_M_Tr | S_M_TrD | A_I_Tr | A_I_TrD | A_M_Tr | A_M_TrD | TT |
|---|---|---|---|---|---|---|---|---|---|---|
| $p2$ | [Jul1, Dec1) | $i1$ | [Jul3, Jul6) | $m1$ | [Aug3, Aug6) | $i1$ | [Jul3, Jul6) | $m1$ | [Aug3, Aug6) | [Jul4, Sep6) |
| $p1$ | [Jul1, Dec1) | $i1$ | [Jul3, Jul6) | $m1$ | [Aug3, Aug6) | $i1$ | [Jul3, Jul6) | $m1$ | [Aug3, Aug6) | [Aug10, $\infty$) |
| $p2$ | [Jul1, Dec1) | $i1$ | [Jul3, Jul6) | $m1$ | [Aug3, Aug6) | $i3$ | [Jul7, Jul10) | $m1$ | [Aug3, Aug6) | [Sep8, $\infty$) |

**Table 2.** Database instance of rehabilitation therapies.

We show now how the above example can be dealt with a single valid time, provided that we properly identify and distinguish the relevant facts. There exist at least three different facts that came into play: the treatment session, which is identified by a session code; the rehabilitation therapy, which is composed by an intensive treatment and a maintenance one and is identified by the prescription code; the prescription, which is also identified by the prescription code, but refers to administrative aspects of the medical domain. Table 3 shows an alternative modeling of Example 2, that provides each fact with a distinct table. Notice that the introduction of three facts allows us to solve some problems, which occur when we use a single table with multiple valid times:

- time dimensions of facts are suitably modeled by single transaction and valid times;
- data redundancy is avoided (for example, the temporal validity of a given prescription is not repeated in several tuples, as well as the valid times of treatment sessions are not repeated for each prescription and more than one time in a single tuple);

– modeling of the temporal aspects is clearer and more precise (for example, therapies can be re-scheduled several times, maintaining the history of schedules and distinguishing between the previous schedules and the current, i.e., actual, schedule).

THERAPY_SCHEDULES

| Pr_Id | I_Tr | M_Tr | VT | TT |
|---|---|---|---|---|
| $p2$ | $i1$ | $m1$ | [Jul1, $\infty$) | [Jul4, Sep6) |
| $p1$ | $i1$ | $m1$ | [Jul2, $\infty$) | [Aug10, $\infty$) |
| $p2$ | $i1$ | $m1$ | [Jul1, Jul3) | [Sep8, $\infty$) |
| $p2$ | $i3$ | $m1$ | [Jul3, $\infty$) | [Sep8, $\infty$) |

TREATMENT_SESSIONS

| Tr | VT | TT |
|---|---|---|
| $i1$ | [Jul3, Jul6) | [Jul4, $\infty$) |
| $m1$ | [Aug3, Aug6) | [Jul4, $\infty$) |
| $i3$ | [Jul7, Jul10) | [Sep8, $\infty$) |

PRESCRIPTIONS

| Pr_Id | VT | TT |
|---|---|---|
| $p2$ | [Jul1, Dec1) | [Jul4, $\infty$) |
| $p1$ | [Jul3, Dec3) | [Aug10, $\infty$) |

**Table 3.** Database instance of the patient rehabilitation treatments.

Finally, it is worth noting that we assume the valid time domain to be linearly ordered, thus excluding branching valid times. Branching valid time is needed to model situations where different perceptions of the reality exist (one for each distinct timeline). An example of this kind of situations comes from historians who hardly agree on how the world evolved. The linear vs. branching valid time alternative is, however, orthogonal to our characterization of the relevant temporal dimensions of data models, that is, the dimensions we propose in the following, can be directly applied also to data models with branching valid times.

### 2.3 A third temporal dimension: the event time

In [17], Kim and Chakravarthy point out the inability of valid and transaction times to distinguish between retroactive and delayed updates and propose the addition of a third temporal dimension to solve this problem.

*Example 3.* Consider the following two scenarios: in the first one, an on-time promotion event, that increases the rank of a physician, occurs on October 1, 1997, but its effects are recorded in the database on November 1 (delayed update); in the second one, a retroactive promotion event, whose effects are immediately recorded in the database, occurs on November 1, but the increase in rank of the physician becomes valid since October 1 (retroactive update).

In the first scenario, the increase in rank is known (and thus it produces its effects) as soon as it becomes valid (October 1), while, in the second scenario, even though the time at which the increase in rank becomes valid precedes the time of the promotion, the increase in rank is not known until the promotion event occurs (November 1). Since two-dimensional (valid and transaction) temporal databases treat retroactive and delayed updates in the same way, they are not able to discriminate between the two scenarios. To overcome these problems, Kim and Chakravarthy [15] introduced the notion of *event time*, which was refined in [16, 17].

**Definition 3.** *Event time: the* event time *(ET) of a fact is the occurrence time of the real-world event that generates the fact.*

Whenever the event time coincides with the starting point of the valid time, it can actually be represented by the valid time. However, such a reduction is not possible when the two times are distinct. Let us consider, for example, the hospitalization of a patient. The valid time is related to the time interval during which the patient stays in the hospital, while the event time is related to the time at which the family doctor decides the hospitalization of the patient or the patient himself decides to go to the hospital. The event time can either coincide with the starting point of the hospitalization, whenever it happens immediately after the decision, or precede the starting point of the hospitalization, in case of scheduled hospitalizations. In [17], Kim and Chakravarthy provide two different classifications of events, based on their relationships with the corresponding valid and transaction times, respectively.

On the basis of the relationship between the event time and the starting point of the corresponding valid interval, one can partition events in three classes:

- On-time events: the validity interval starts at the occurrence time of the event (e.g., hospitalization starts immediately after the family doctor decision).
- Retroactive events: the validity interval starts before the occurrence time of the event (e.g., on December 21, 1997, the manager of the hospital decided an increase of 10% of the salary of the physicians of the Pathology Department, starting from December 1, 1997).
- Proactive events: the validity interval starts after the occurrence time of the event (e.g., on January 29, 1998, the family doctor decides the patient hospitalization on February 15, 1998).

When a database is updated with a fact generated by a retroactive event, Kim and Chakravarthy call the update a *retroactive update*, while, when the generating event is a proactive event, they call the update a *proactive update* [17]. This terminology is somehow misleading, because it may happen that the database update of a fact generated by a proactive event actually occurs after the beginning of the validity interval of the corresponding fact.

A different classification can be obtained by considering the relationships between event time and transaction time:

- On-time update: the transaction time coincides with the event time. This situation happens when data values are inserted in the database as soon as they are generated.

- Delayed update: the transaction time is greater than the event time. This is the case when data values are inserted some time after their generation.
- Anticipated update (which has not been taken into account by Kim and Chakravarthy): the transaction time is less than the event time. This is the case when data values are entered into the database before the occurrence time of the event that generates them. Such a notion of anticipated update is useful to model hypothetical courses of events.

It is not difficult to realize that the two classifications are orthogonal, and thus all possible combinations of them are admissible.

## 3 One event time is not enough

The original notion of event time suffers from an intrinsic limitation: it implicitly assumes that it is sufficient to associate a single event (the so-called generating event) with each valid fact. The generating event is the event that initiates the validity interval of the fact. In many situations, such an assumption is acceptable. Consider the case of a relation that keeps track of the temporal evolution of the rank of a physician. Each event that either increases (promotion) or decreases (downgrading) the rank not only initiates a period of time during which the physician has the new rank, but also implicitly terminates the validity interval of the current rank. In such a case, the event time of each tuple in the rank relation is the occurrence time of the promotion or downgrading event that initiates its validity interval, while the occurrence time of the event that terminates the validity interval is the event time of the tuple describing the next rank of the physician. On the contrary, whenever there are gaps in temporal validity (consider the case of a relation modeling admissions to hospital, which only records information about inpatients), or only incomplete information about the effects of an event is available (this is the case when we know that an event that changes the rank of the physician occurred, but we do not know if it is a promotion or a downgrading event), or, finally, the expected termination of a validity interval is (must be) revised, while its initiation remains unchanged (suppose that a prescribed therapy needs to be stopped due to an unexpected evolution of the patient state), we need to distinguish between the occurrence times of the two events that respectively initiate and terminate the validity interval of the fact.

The following example, which slightly revises Example 1, should clarify the point.

*Example 4.* On August 10, 1998, at 8:00, the physician prescribes a bipuvac-based therapy from 10:00 to 14:00. Data about the therapy is entered into the database at 9:00. Due to the unexpected evolution of the patient state, at 11:00 the physician decides a change in the patient therapy. Accordingly, the bipuvac infusion is stopped at 11:15 and replaced by a diazepam-based therapy from 11:25 to 14:00. The new facts are entered at 12:00.

This example can be modeled as in Table 4. The combined use of valid, transaction, and event times allows us to characterize the times at which the physician takes his/her decisions about the therapy, the time intervals during which therapies are administered, and the times at which facts are entered into the database. However, the proposed representation suffers from a major weak-

| Drug | VT | ET | TT |
|---|---|---|---|
| bipuvac | [98Aug10;10:00, 98Aug10;14:00) | 98Aug10;8:00 | [98Aug10;9:00, 98Aug10;12:00) |
| bipuvac | [98Aug10;10:00, 98Aug10;11:15) | 98Aug10;8:00 | [98Aug10;12:00, $\infty$) |
| diazepam | [98Aug10;11:25, 98Aug10;14:00) | 98Aug10;11:00 | [98Aug10;12:00, $\infty$) |

**Table 4.** Database instance of the patient therapies with one event time (case 1).

ness: it seems that, at 8:00, the physician prescribes a bipuvac-based therapy from 10:00 to 11:15 and that, at 11:00, he/she prescribes a diazepam-based one. This is not correct because at 8:00 the physician actually prescribes a bipuvac-based therapy from 10:00 to 14:00, which is revised at 11:00[3]. The alternative representation given in Table 5 also does not work. According to such a representation, it seems that, at 11:00, the physician prescribes a bipuvac-based therapy from 10:00 to 11:15, thus making the prescription a retroactive event (which is clearly meaningless is the considered domain). The problem illustrated by the

| Drug | VT | ET | TT |
|---|---|---|---|
| bipuvac | [98Aug10;10:00, 98Aug10;14:00) | 98Aug10;8:00 | [98Aug10;9:00, 98Aug10;12:00) |
| bipuvac | [98Aug10;10:00, 98Aug10;11:15) | 98Aug10;11:00 | [98Aug10;12:00, $\infty$) |
| diazepam | [98Aug10;11:25, 98Aug10;14:00) | 98Aug10;11:00 | [98Aug10;12:00, $\infty$) |

**Table 5.** Database instance of the patient therapies with one event time (case 2).

above example motivated us to suitably refine the concept of event time. We distinguish between the occurrence time of the event that initiates the validity interval of a fact (*initiating event time*) and the occurrence time of the event that terminates the validity interval of a fact (*terminating event time*).

| Drug | VT | $\mathrm{ET}_i$ | $\mathrm{ET}_t$ | TT |
|---|---|---|---|---|
| bipuvac | [98Aug10;10:00, 98Aug10;14:00) | 98Aug10;8:00 | 98Aug10;8:00 | [98Aug10;9:00, 98Aug10;12:00) |
| bipuvac | [98Aug10;10:00, 98Aug10;11:15) | 98Aug10;8:00 | 98Aug10;11:00 | [98Aug10;12:00, $\infty$) |
| diazepam | [98Aug10;11:25, 98Aug10;14:00) | 98Aug10;11:00 | 98Aug10;11:00 | [98Aug10;12:00, $\infty$) |

**Table 6.** Database instance of the patient therapies with two event times.

**Definition 4.** *Event time (revisited): the* event time *of a fact is the occurrence time of a real-world event that either initiates or terminates the validity interval of the fact.*

[3] Notice that the first tuple cannot be used to explain the situation, because it has been logically deleted, and thus it does not model the current state of the world.

For any fact, we denote its initiating and terminating event times by $ET_i$ and $ET_t$, respectively. Obviously, it holds that $ET_i \leq ET_t$. The possible relationships between event and valid times of a fact are depicted in Figure 1. Providing each fact with these two event times makes it possible to correctly model the considered example, as shown in Table 6. Notice that, unlike what has been pointed out in [14], there are more than one event time per fact (the initiating and terminating event times), even when there is only one decision maker (the physician in the above example).

This distinction between initiating and terminating event times comes from ideas underlying classical formalisms in the area of reasoning about actions and change [6]. In particular, a similar classification of events is given in the model of change of Kowalski and Sergot's Event Calculus (EC) [19]. The notions of event, property, time point, and time interval are the primitives of the formalism: events happen at time points and initiate and/or terminate time intervals over which properties hold [5]. From a description of the events which occur in the real world and the properties they initiate or terminate, EC derives the validity intervals over which properties hold. In temporal deductive databases, this approach has been adopted, for example, by Sripada in [29], where EC is extended to manage the transaction time. The main differences between our database-oriented approach and the EC ontology are: (i) EC explicitly records event occurrences and derives validity intervals of properties at query time, while we directly record properties (i.e., facts) and their validity intervals, together with the occurrence times of the events that initiate and terminate them; (ii) EC only deals with on-time events, while we also consider proactive and retroactive events[4]. The

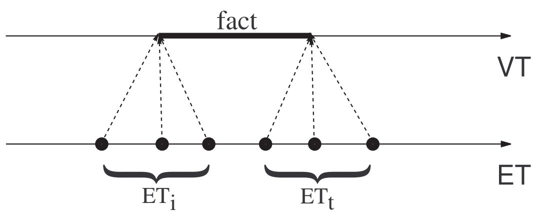


**Fig. 1.** On the relationships between event ($ET_i$, $ET_t$) and valid ($VT$) times of a fact.

choice of adding event time(s) as a separate temporal dimension has been extensively debated in the literature [24]. The basic issue is whether or not events and facts must be dealt with in a different way in temporal databases. If we consider events just as a special class of facts (instantaneous facts), we do not need event time(s) at all: events are explicitly recorded in the database, as all the other relevant facts, and their occurrence time is (captured by) the valid time. A major weakness of this approach is that the data model does not provide the relationships between events and initiated/terminated facts with any

[4] For the sake of simplicity, we only considered atomic events. Whenever the initiation (resp. termination) of the validity of a fact is the combined effect of a set of events, we need to replace atomic events by composite events. A preliminary formalization of a calculus of macro-events can be found in [4].

built-in semantics, but delegates such a task to the application. On the contrary, conventional relations model the reality relevant to a given domain as a set of temporally-extended facts, which are delimited by the occurrence times of their initiating and terminating events [13]. Events are not considered first-class citizens and thus are not recorded into the database. In such a case, the addition of event time(s) is needed to deal with retroactive and/or proactive events that occur at times which do not coincide with the starting (resp. ending) point of the valid time of the fact they initiate (resp. terminate). In our opinion, this second approach provides a model of the domain which is both more adequate and more concise. On the one hand, it explicitly represents the cause-effect relationships that capture the behaviour of the domain (adequacy). On the other hand, it completes the temporal description of a fact by representing the occurrence times of its initiating/terminating events, without imposing the explicit storage of other information about them, which is often neither needed nor possible (conciseness)[5].

## 4 A new temporal dimension: the availability time

Event times are strictly related to the modeled world, as they allow one to represent the occurrence times of events (decisions, happenings, actions, etc.) that initiate or terminate meaningful facts of the considered domain. In this section, we focus on the temporal dimensions relevant to the information system. By *information system* we mean the set of information flows of an organization and the human and computer-based resources that manage them. From such a point of view, we may need to model the time at which (someone/something within) the information system becomes aware of a fact as well as the time at which the fact is stored into the database. While the latter temporal aspect is captured by the transaction time, the former has never been explicitly modeled.

We first show that such a time neither coincides with any of the two event times nor with the transaction time. Let us consider the following scenario.

*Example 5.* Due to a trauma that occurred on September 15, 1997, Mary suffered from a severe headache starting from October 1. On October 7, Mary was visited by a physician. On October 9, the physician administered her a suitable drug. The day after, the physician entered acquired information about Mary's medical history into the database. On October 15, the patient told the physician that her headache stopped on October 14; the physician entered this data into the database on the same day.

This scenario involves several temporal dimensions: the time interval during which Mary suffered from headache, the occurrence time of the trauma, the occurrence time of the drug administration, the times at which the physician

[5] In those few cases where further information about initiating/terminating events is needed (for instance, with regards to Example 4, we may be interested in storing the name and the specialty of the physician(s), who prescribed the therapies), event times should be paired with other event-related attributes.

became aware of Mary's headache onset and cessation, and finally the times at which the database is updated. Valid and transaction times are respectively

| symptom | VT | $ET_i$ | $ET_t$ | TT |
|---|---|---|---|---|
| headache | [97Oct1, $\infty$) | 97Sept15 | null | [97Oct10, $\infty$) |

**Table 7.** Database instance after the first update.

used to record the time interval during which Mary suffered from headache and the database update times, while the occurrence times of trauma and drug administration are modeled by means of the initiating and terminating event times (cf. Tables 7 and 8). There is no way of representing the times at which the physician became aware of Mary's headache onset and cessation. Indeed, while information about Mary's headache cessation is entered in the database as soon as it became available, information about the headache onset became available three days before its registration into the database. In general, the time

| symptom | VT | $ET_i$ | $ET_t$ | TT |
|---|---|---|---|---|
| headache | [97Oct1, $\infty$) | 97Sept15 | null | [97Oct10, 97Oct15) |
| headache | [97Oct1, 97Oct14) | 97Sept15 | 97OCt9 | [97Oct15, $\infty$) |

**Table 8.** Database instance after the second update.

at which information becomes available precedes, but not necessarily coincides with, the time at which it is recorded in the database, and thus we cannot use transaction time to model it. Indeed, in many applications, where data insertions are grouped and executed in batches, possibly on the basis of previously filled report forms, transaction time cannot be safely taken as the time when a fact has been acquired by the information system. In some cases, it may happen that the order according to which facts are known by the information system differs from the order in which they are stored into the database. Since there are many application domains, including the medical one, where decisions are taken on the basis of the available information, no matter whether or not it is stored in the database, we need to introduce a new temporal dimension, that we call *availability time*, to deal with it. We informally define the *availability time* (AT) as the time at which a fact becomes available to the information system. Later on (cf. Definition 5), we will generalize and make more precise such a definition. Figure 2 shows how transaction and availability times are related to the database and information systems, respectively. Notice that the information system includes, but does not necessarily coincide with, the database system. In our medical example, for instance, the information system includes both the database and the physician.

By exploiting the availability time, we are able to completely model the previous example. As shown in Table 9, the availability time of the first tuple, which is the time at which the physician learns about Mary's headache onset (October 7), strictly precedes the beginning of the transaction time interval (October 10), while the database time and the beginning of the transaction time interval of the second tuple coincide (October 15).

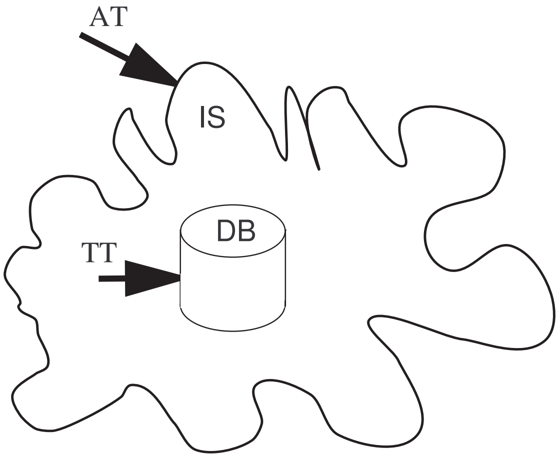


**Fig. 2.** Availability ($AT$) and transaction ($TT$) times and their relationships with the database (DB) and information systems (IS).

Let us suppose that we are interested in evaluating the quality of the care provided by the physician. In case we lack information about the availability time (cf. Table 8), we only know that Mary's headache started on October 1, due to some event occurred on September 15 (as of October 10), and stopped on October 14, thanks to (presumably) some therapeutic action occurred on October 9 (as of October 15). As of October 10, we can conclude only that the physician became aware of Mary's headache some time in between October 1 (the starting point of the validity interval) and October 10 (the starting point of the transaction time). This uncertainty makes it difficult to assess the quality of the care, because it is not possible to unambiguously relate clinical data (headache onset and cessation) to therapeutic strategies (to decide the proper therapeutic action, on the basis of the available knowledge about the patient history). In particular, if we assumed that the physician became aware of the

| symptom | VT | ET$_i$ | ET$_t$ | AT | TT |
|---|---|---|---|---|---|
| headache | [97Oct1, $\infty$) | 97Sept15 | null | 97Oct7 | [97Oct10, 97Oct15) |
| headache | [97Oct1, 97Oct14) | 97Sept15 | 97Oct9 | 97Oct15 | [97Oct15, $\infty$) |

**Table 9.** Database instance after the second update (revisited).

headache on October 1, we would observe a delay of more than one week between the awareness of Mary's problem and the execution of the action that solved the problem. On the contrary, if we assumed that the physician became aware of Mary's headache on October 10, we should conclude that Mary recovered from her headache without any intervention from the physician (the event that terminates the validity time interval of the headache happened on October 9).

**Remark**. *The concept of availability time allows us to clarify the relationships between Kim and Chakravarthy's (initiating) event time and the related notion of decision time, which has been originally proposed by Etzion and his colleagues in [8] and later refined in subsequent work, e.g., in [9]*[6]. *The* decision time *of a fact is the time at which the fact is decided in the application domain of discourse. More precisely, it can be defined as the occurrence time of a real-world event, whose happening induces the decision of inserting a fact into the database. In the conceptual framework we propose, the decision time models the temporal aspects of an event, initiating the validity interval of a fact, that occurs within the information system, and thus is immediately known by it. In these circumstances, the initiating event time and the availability time coincide (and are before than or equal to the starting point of the transaction time interval).*

Up to now, we have used the availability time to model the time at which relevant knowledge becomes available to the system, without considering the possibility that it acquires erroneous data. In the following, we will show how the notion of availability time can be generalized to take into account such a possibility. Let us now consider the following variant of the previous example.

*Example 6.* Due to an insertion mistake (or to an imprecision in Mary's talk), the trauma has been registered as happened on September 5, 1997. Only on October 20, the mistake was discovered. The day after, the physician entered the correct data into the database.

| symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|
| headache | [97Oct1, $\infty$) | 97Sept5 | null | 97Oct7 | [97Oct10, 97Oct15) |
| headache | [97Oct1, 97Oct14) | 97Sept5 | 97Oct9 | 97Oct15 | [97Oct15, 97Oct21) |
| headache | [97Oct1, 97Oct14) | 97Sept15 | 97Oct9 | 97Oct20 | [97Oct21, $\infty$) |

**Table 10.** Database instance after error correction.

The database instance resulting from the error correction is shown in Table 10. A limitation of this representation is that information about the time interval during which any fact is known and *believed correct* cannot be obtained from the corresponding tuple. As an example, to conclude that the second tuple was considered correct until October 20, we need to access the availability time of the third tuple. Unfortunately (cf. [3]), it is not possible, in general, to determine how a given tuple of a relation has been updated, whenever several modifications and/or insertions are performed by means of a single transaction (in such a case, all the updated tuples share the same transaction time). Even worse, when the discovery of an error forces us to (logically) delete some tuples, which are not replaced by new ones (suppose that, on October 20, we discover that the trauma did not happen at September 5, but we do not know yet when it actually happened), there is no way of keeping trace of the time at which the error has been discovered.

[6] Indexing techniques for temporal databases supporting the decision time have been proposed by Nascimento and his colleagues [21, 22].

That leads us to consider the availability time as an interval for a fact. The starting point of the availability time is the time at which the fact becomes available to the information system, while its ending point is the time at which the information system realizes that the fact is not correct (as for transaction time, being the ending point equal to $\infty$ means that the fact is currently believed correct). By modeling availability time as an interval, we are able to correctly represent the previous example, as shown in Table 11.

| symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|
| headache | [97Oct1, $\infty$) | 97Sept5 | null | [97Oct7, 97Oct15) | [97Oct10, 97Oct15) |
| headache | [97Oct1, 97Oct14) | 97Sept5 | 97Oct9 | [97Oct15, 97Oct20) | [97Oct15, 97Oct21) |
| headache | [97Oct1, 97Oct14) | 97Sept15 | 97Oct9 | [97Oct20 , $\infty$) | [97Oct21, $\infty$) |

**Table 11.** Database instance after error correction (revisited).

**Definition 5.** *Availability Time: the* availability time *(AT) of a fact is the time interval during which the fact is known and believed correct by the information system.*

As for the relationships between database and transaction times of a fact, it holds that $AT_s \leq TT_s$ and $AT_e \leq TT_e$: a fact can be inserted into the database only when it is known, or later, and it can be logically removed from the database only when it is recognized incorrect, or later.

**Remark.** *Notice that the availability time can be viewed as the transaction time of the information system. If the information system outside the database is considered a database (in general, this is not case; in our medical examples, for instance, the information system includes both databases and physicians), then the availability time of a fact is when the fact is inserted into (deleted from) the information system. This parallel between availability time and transaction time makes it immediately clear why the availability time has to be an interval. This point of view on the availability time resembles the notion of temporal generalization (cf. [12]) which allows several levels, and thus several transaction times, in a database.*

Figure 3 summarizes the relationships between valid, transaction, event, and availability times, the modeled reality, and the information system.

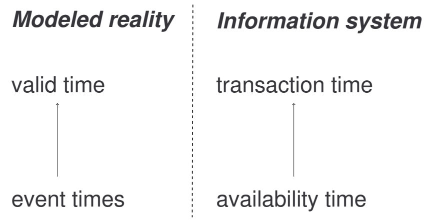


**Fig. 3.** Temporal dimensions, modeled reality, and information system.

Transaction time is append-only: facts previously stored into the database cannot be changed. The acquisition of new knowledge about the domain results into the addition of new facts, whose transaction time interval includes the current time, to the database; the (logical) deletion of incorrect knowledge is obtained by "closing" the transaction times of the corresponding facts; modifications can be performed by combining the two previous actions. In contrast, valid and event times, being related to times of the represented real world, can be located either in the past or in the future, and they can be modified freely. As regards the availability time, it is append-only in its nature, because facts previously known and believed correct by the information system cannot be changed. However, from the point of view of the database system, availability time would be really append-only only if there were no errors in data entry. Since we cannot exclude such a possibility, previous states of the information system, according to the availability time, can be revised by entering new data. Furthermore, even assuming data entry without errors, database and transaction times may "append" facts according to two different orders.

## 5 Conclusions and further work

In this paper, we proposed a new, fully symmetric conceptual data model with multiple temporal dimensions, that revises and extends existing temporal models. We first described a refinement of the concept of event time, that replaces the single event time, originally proposed by Kim and Chakravarthy, by a pair of *initiating* and *terminating event times*. In such a way, we are able to represent real-world situations where two different events, with (possibly) different occurrence times, are related to the starting and ending points of the validity interval of a fact. Then, we introduced a new temporal dimension, named *availability time*, which captures the time interval during which a fact is known and believed correct by the information system. The availability time allows us to distinguish the time at which a fact becomes available to the information system from the time at which it is entered into the database. This capability can be exploited, for example, to analyze the quality of decision making in information systems where data insertion is performed in batches and thus some delay is possible between data availability and data insertion.

Whether a special support for event and availability times must be provided or not is debatable. This issue can be considered as an instance of the more general problem of deciding whether or not built-in supports for temporal dimensions in databases must be provided (since the proposal of valid and transaction times, there has been a dispute about that). In this paper, we have shown that, from a conceptual point of view, event and availability times are temporal dimensions that capture some general aspects of applications which cannot be naturally managed by using valid and transaction times, and thus event and availability times cannot be reduced to simple user-defined times. We believe that capturing the application-independent meaning of temporal information is a main step towards the conceptual modeling of temporal information systems [27]. Highlighting these aspects at the conceptual level is, indeed, a basic task both when designing information systems by standard (atemporal) methodologies and when designing and adopting fully-fledged temporally-oriented methodologies [10]. The design issues, we are faced with when dealing with the problem of defining a query language for temporal databases (cf. Appendix B), confirm that all the four temporal dimensions must somehow be taken into account, also in the case we decide to manage them by a classical atemporal database system.

As for current and future research, we are working at a generalization of the temporal logic reconstruction of valid and transaction temporal databases outlined in [20], that copes with both event and availability times. Another problem we are dealing with is the identification and characterization of suitable temporal normal forms. Many temporal dependencies among data may indeed arise when relations are provided with multiple temporal dimensions [25]. Finally, it goes without saying that suitable indexing techniques are needed to effectively manage temporal databases with four temporal dimensions. To this end, we are currently analyzing existing indexing techniques for multidimensional data. The existence of general constraints among the different temporal dimensions hints at the possibility of tailoring existing indexing techniques and search algorithms to our specific context.

## Acknowledgments

We would like to thank Alberto Policriti, Richard Snodgrass, and the anonymous reviewers for their useful suggestions and comments on the work, Alberto Pasetto, chair of the Anesthesiology and Intensive Care Unit of the University of Udine, and his staff, who helped us to select meaningful clinical examples and gave us the opportunity to test a prototype of a database system with multiple temporal dimensions in a real-world, data-intensive, clinical unit, and our former students Ivan Andrian, Silvia Riva, and Fabio Valeri that contributed to the development of the temporal data model, to the study of temporal functional dependencies, and to the definition and implementation of the temporal query language T4SQL, respectively.

## References

1. J.F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26: 832–843, 1983.
2. I. Andrian. Temporal Dimensions in the Relational Data Model: a New Characterization with Applications to Clinical Databases in Intensive Care Units. (In Italian). Master Degree in Computer Science. Under the supervision of C. Combi, A. Montanari, and A. Pasetto, 1998.
3. G. Bhagrava and S.K. Gadia. The Concept of Error in a Database: an Application of Temporal Databases. Proc. of the International Conference on Management of Data (COMAD), McGraw-Hill, New York, 106–121, 1990.
4. I. Cervesato and A. Montanari. A Calculus of Macro-Events: Progress Report. In A. Trudel, S. Goodwin (eds.), Proc. of the 7th International Workshop on Temporal Representation and Reasoning (TIME). IEEE Computer Society Press, Los Alamitos, 47–58, 2000.

5. L. Chittaro and A. Montanari. Efficient Temporal Reasoning in the Cached Event Calculus. *Computational Intelligence*, 12: 359–382, 1996.
6. L. Chittaro and A. Montanari. Temporal representation and reasoning in artificial intelligence: Issues and approaches. *Annals of Mathematics and Artificial Intelligence*, 28(1-4): 47–106, 2000.
7. C. Combi, C. Cucchi, and F. Pinciroli. Applying Object-Oriented Technologies in Modeling and Querying Temporally-Oriented Clinical Databases Dealing with Temporal Granularity and Indeterminacy. *IEEE Transactions on Information Technology in Biomedicine*, 1(2): 100–127, 1997.
8. O. Etzion, A. Gal, and A. Segev. Temporal Support in Active Databases. In Proc. of the Workshop on Information Technologies & Systems (WITS), 245–254, 1992.
9. O. Etzion, A. Gal, and A. Segev. Extended Update Functionality in Temporal Databases. In O. Etzion, S. Jajodia, and S. Sripada (eds.), *Temporal Databases - Research and Practice*, LNCS 1399, Springer, Berlin Heidelberg, 56–95, 1998.
10. H. Gregersen and C.S. Jensen. Temporal Entity-Relationship Models: a Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3): 464–497, 1999.
11. C. Jensen, C. Dyreson (Eds.) et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In O. Etzion, S. Jajodia, and S. Sripada (eds.), *Temporal Databases - Research and Practice*, LNCS 1399, Springer, Berlin Heidelberg, 367–405, 1998.
12. C. Jensen and R. Snodgrass. Temporal Specialization and Generalization. *IEEE Transactions on Knowledge and Data Engineering*, 6: 954–974, 1994.
13. C. Jensen and R. Snodgrass. Semantics of Time-Varying Information. *Information Systems*, 21(4): 311–352, 1996.
14. C. Jensen and R.T. Snodgrass. Temporal Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 11: 36–44, 1999.
15. S.K. Kim and S. Chakravarthy. Semantics of Time-Varying Information and Resolution of Time Concepts in Temporal Databases. In R.T. Snodgrass (ed.), Proc. of the International Workshop on an Infrastructure for Temporal Databases, Arlington, TX, G1–G13, 1993.
16. S.K. Kim and S. Chakravarthy. Modeling Time: Adequacy of Three Distinct Time Concepts for Temporal Databases. In Proc. of the 12th International Conference of the Entity-Relationship Approach, LNCS 823, Springer, Berlin Heidelberg, 475–491, 1993.
17. S.K. Kim and S. Chakravarthy. Resolution of Time Concepts in Temporal Databases. *Information Sciences*, 80: 91–125, 1994.
18. S. Kokkotos, E.V. Ioannidis, T. Panayiotopoulos, and C.D. Spyropoulos. On the Issue of Valid Time(s) in Temporal Databases. *SIGMOD Record*, 24(3): 40–43, 1995.
19. R. Kowalski and M. Sergot. A Logic-Based Calculus of Events, *New Generation Computing*, 4: 67–95, 1986.
20. A. Montanari and B. Pernici. Towards a Temporal Logic Reconstruction of Temporal Databases. In R.T. Snodgrass (ed.), Proc. of the International Workshop on an Infrastructure for Temporal Databases, Arlington, TX, BB1–BB12, 1993.
21. M. Nascimento and M. Dunham. Indexing a Transaction-Decision Time Database. In Proc. of ACM Symposium of Applied Computing (SAC), ACM Press, New York, 166–172, 1996.
22. M. Nascimento and M. H. Eich. On Decision Time for Temporal Databases. In S. Goodwin and H. Hamilton (eds.), Proc. of the 2nd International Workshop on Temporal Representation and Reasoning (TIME), 157–162, 1995.

23. S.B. Navathe and R. Ahmed. Temporal Extensions to the Relational Model and SQL. In A. Tansell, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.), *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings Pub. Co., Redwood City, 92–109, 1993.
24. G. Özsoyoglu and R. T. Snodgrass. Temporal and Real-Time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4): 513–532, 1995.
25. S. Riva. Functional dependencies in relational databases with multiple temporal dimensions. (In Italian). Master Degree in Computer Science. Under the supervision of C. Combi and A. Montanari, 2000.
26. B. Salzberg and V.J. Tsotras. Comparison of Access Methods for Time Evolving Data. *ACM Computing Surveys*, 31(2): 158–221, 1999.
27. Y. Shahar and C. Combi. Editors' Foreword: Intelligent Temporal Information Systems in Medicine. *Journal of Intelligent Information Systems (JIIS)*, 13(1-2): 5–8, 1999.
28. R. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, Boston, 1995.
29. S.M. Sripada. A logical framework for temporal deductive databases. In Proc. 14th Very Large Data Bases Conference, Morgan Kaufmann, San Francisco, 171–182, 1988.
30. F. Valeri. T4-SQL: a Query Language for Relational Databases with Multiple Temporal Dimensions. (In Italian). Master Degree in Computer Science. Under the supervision of C. Combi and A. Montanari, 1999.

## Appendix A: a taxonomy of multidimensional temporal databases

In this appendix, we address the problem of classifying a temporal relation according to the relationships among the proposed temporal dimensions of a fact. Let a temporal database be defined as a set of *temporal* relations.

**Definition 6.** *Temporal relation a* temporal relation *is a relation where each tuple (representing a fact) is fully timestamped, that is, it is provided with valid time (VT), initiating and terminating event times ($ET_i$, $ET_t$), availability time (AT), and transaction time (TT).*

As pointed out in [21, 26], the task of classifying temporal relations according to the relationships between their temporal dimensions is an important step towards the efficient application of indexing techniques. Moreover, such a classification can be exploited to assign event and availability times to those relations which are provided only with valid and transaction times.

In the following, we do not consider the classification of temporal databases based on the relationships between valid and transaction times, because such a classification has been already proposed and systematically investigated in the literature. We refer the interested reader to [12] for a detailed analysis of the relationships between these temporal dimensions.

### 5.1 Event and transaction times

As for the relationships between event and transaction times, the following classification is a straightforward generalization of the taxonomy proposed in [17] (cf. Section 2.3), that allows us to manage two event times.

- On-time update. For each fact of the relation, the initiating and terminating event times coincide with the starting point of the transaction time ($ET_i = TT_s \wedge ET_t = TT_s$).
- Delayed update. For each fact of the relation, the initiating and terminating event times precede the starting point of the transaction time ($ET_t < TT_s$).
- Anticipated update. For each fact of the relation, the event times follow the starting point of the transaction time ($ET_i > TT_s$).
- Delayed-on-time update. For each fact of the relation, the initiating (resp. terminating) event time precedes (resp. coincides with) the starting point of the transaction time ($ET_i < TT_s \wedge ET_t = TT_s$).
- On-time-anticipated update. For each fact of the relation, the initiating (resp. terminating) event time coincides with (resp. follows) the starting point of the transaction time ($ET_i = TT_s \wedge ET_t > TT_s$).
- Delayed-anticipated update. For each fact of the relation, the initiating (resp. terminating) event time precedes (resp. follows) the starting point of the transaction time ($ET_i < TT_s \wedge ET_t > TT_s$).

### 5.2 Event and valid times

As for the relationships between event and valid times, temporal relations can be classified according to the taxonomy proposed in [17] (cf. Section 2.3). However, unlike the event classification provided in [17], we are able to capture situations in which both retroactive and proactive events are related to the valid time of a fact (e.g., a proactive event initiates the valid time interval and a retroactive one terminates it).

- Proactive event-valid time relations: for each fact of the relation, the initiating and terminating events are proactive events, with respect to the starting and ending points of the valid time interval, respectively ($ET_i < VT_s \wedge ET_t < VT_e$).
- On-time event-valid time relations: for each fact of the relation, the initiating and terminating events are on-time events, with respect to the starting and ending points of the valid time interval, respectively ($ET_i = VT_s \wedge ET_t = VT_e$).
- Retroactive event-valid time relations: for each fact of the relation, the initiating and terminating events are retroactive events, with respect to the starting and ending points of the valid time interval, respectively ($ET_i > VT_s \wedge ET_t > VT_e$).
- Pro-retro event-valid time relations: for each fact of the relation, the initiating (resp. terminating) event is a proactive (resp. retroactive) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i < VT_s \wedge ET_t > VT_e$).
- Retro-pro event-valid time relations: for each fact of the relation, the initiating (resp. terminating) event is a retroactive (resp. proactive) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i > VT_s \wedge ET_t < VT_e$).
- Pro-on-time event-valid time relations: for each fact of the relation, the initiating (resp. terminating) event is a proactive (resp. on-time) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i < VT_s \wedge ET_t = VT_e$).
- Retro-on-time event-valid time relations: for each fact of the relation, the initiating (resp. terminating) event is a retroactive (on-time) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i > VT_s \wedge ET_t = VT_e$).
- On-time-pro event-valid time relations: for each fact of the relation, the initiating (terminating) event is an on-time (resp. proactive) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i = VT_s \wedge ET_t < VT_e$).
- On-time-retro event-valid time relations: for each fact of the relation, the initiating (resp. terminating) event is an on-time (retroactive) event, with respect to the starting (resp. ending) point of the valid time interval ($ET_i = VT_s \wedge ET_t > VT_e$).

A finer classification can be obtained by taking into account the relationships between the initiating and/or terminating event times and both the endpoints of the valid time interval, as well as by adding quantitative requirements to the previous constraints (e.g., the initiating event must occur exactly 3 minutes before the starting point of the valid time interval).

### 5.3 Availability and transaction times

To classify relations according to the values of availability and transaction times, we use the relationships between intervals defined by Allen in [1], expressed as relationships between the starting and ending points of intervals.

- Immediate update: for each fact of the relation, the availability time coincides with the transaction time ($AT_s = TT_s \wedge AT_e = TT_e$).
- Delayed start and delayed end update: for each fact of the relation, the availability time overlaps the transaction time ($AT_s < TT_s \wedge TT_s < AT_e \wedge AT_e < TT_e$).
- Delayed end update: for each fact of the relation, the availability time is started by the transaction time ($AT_s = TT_s \wedge AT_e < TT_e$).
- Delayed start update: for each fact of the relation, the availability time is finished by the transaction time ($AT_s < TT_s \wedge AT_e = TT_e$).

### 5.4 Availability and valid times

As for the availability and valid times, we can exploit the extended taxonomy provided by Jensen and Snodgrass in [12]. The only difference is that, in this case, the terms *retroactive*, *predictive*, and so on, express the relationships between the valid time of a fact and the time interval during which the fact is known and believed correct by the information system, disregarding the time at which it is entered into the database.

### 5.5 Availability and event times

The classification of temporal relations on the basis of the relationships between event and availability times closely resembles the classification based on the relationships between event and transaction times. The only difference is that, instead of delayed update, anticipated update, on-time update, and so on, we have delayed data availability, anticipated data availability, on-time data availability , and so on.

## Appendix B: towards a query language for multidimensional temporal databases

The management of multiple temporal dimensions has an impact on many aspects of query languages, ranging from the definition of DDL statements, which allow one to express constraints among the different temporal dimensions, to the definition of DML statements, supporting both the specification of temporal search conditions and the definition of the temporal dimensions of the query result. A query language for relational databases with multiple temporal dimensions is described in detail in [30]. In this section, we restrict our attention to the aspects specifically related to queries involving the four temporal dimensions. In particular, we discuss some issues related to the design of a query language for temporal databases with multiple temporal dimensions, including the definition of suitable clauses to query about past states of both the database system and the information system (`AS OF`, `AS KNOWN`), the addition to the `SELECT` clause of qualifiers that allow one to compute the value of the temporal attributes of the resulting relation (`WITH ... AS VALID, ... AS INITIATING`), and the addition of some clauses and/or of some keywords to support the definition of temporal joins, temporal conditions, and so on (`TJOIN ON ..., WHEN ....`).

Without loss of generality, we start with the standard `SELECT` command of SQL and introduce, step by step, the specific extensions/modifications needed to deal with multiple temporal dimensions.

In order to allow comparisons between temporal dimensions, two choices are possible: the first one is to deal with temporal and non-temporal data in the `WHERE` clause [28]; the second one is to constrain purely temporal conditions to appear in a specific clause `WHEN` [23][7]. In case we opt for the second alternative (this allows one to exploit specific tools to manage temporal data, at both the logical and physical level), another design issue is the specification of the temporal dimensions we can refer to in the `WHEN` clause. The most general option is to allow all the four temporal dimensions to be used in the `WHEN` clause. According to this choice, a query that returns all and only the patient's symptoms that appeared no later than 3 days after their initiating event (cf. Table 11) can be formulated as follows:

```
SELECT symptom
FROM sympt_re S
WHEN BEGIN(VALID(S)) - INITIATING_ET(S) ≤ 3
```

where `sympt_re` is the relation name, and `INITIATING_ET(·)`, `VALID(·)`, and `BEGIN(·)` are functions that return the initiating event time of a tuple, the valid time interval of a tuple, and the starting point of a given interval, respectively.

By default, a query is evaluated with respect to the current state of the database, i.e., on the tuples whose transaction time is an interval open to the right. Obviously, there must be the possibility of evaluating the query over both the current and the past states of the database and/or of the information system; this requirement can be managed by adding suitable qualifiers to the `FROM` clause. If we are interested in querying the database about its past states, we can use

[7] Obviously, to make it possible to check conditions mixing temporal and non-temporal data, we must allow temporal dimensions to occur, whenever necessary, in the WHERE clause.

the well-known `AS OF` clause, which has been added to many temporal query languages proposed in the literature (cf. [28]). To allow the user to query the database about the knowledge which is available to the information system at given time points, we added the `AS KNOWN` clause. Let us consider, for example, the following queries on the relation depicted in Table 11

```
SELECT *
FROM sympt_re S
AS OF 97Oct20
```

```
SELECT *
FROM sympt_re S
AS KNOWN 97Oct20
```

which return the contents of the relation *sympt_re* as of October 20, 1997, and as known on October 20, 1997, respectively. The results of these queries are reported in Tables 12 and 13, respectively. It is worth noting that the separated use of the `AS OF` and `AS KNOWN` clauses allow the user to execute queries focusing on (possibly) different aspects of stored information. In the proposed example, by comparing the results of the two queries, it is possible to conclude that on October 20, 1997 the database was out-of-date with respect to the facts known by the information system at that time.

| symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|
| headache | [97Oct1, 97Oct14) | 97Sept5 | 97Oct9 | [97Oct15, 97Oct20) | [97Oct15, 97Oct21) |

**Table 12.** Tuples of the database instance of Table 11 as of October 20, 1997.

| symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|
| headache | [97Oct1, 97Oct14) | 97Sept15 | 97Oct9 | [97Oct20 , $\infty$) | [97Oct21, $\infty$) |

**Table 13.** Tuples of the database instance of Table 11 as known on October 20, 1997.

The joined use of both clauses allow the user to obtain further meaningful information, as shown by the following example.

*Example 7.* In Table 14 (relation *pat_sympt*), we reported information about the symptoms of two patients: Mary (`P_id`=1) and Sam (`P_id`=2). Data about the headache of Mary is that of Example 6. As for the symptoms of Sam, on September 3, 1997, the physician was informed that Sam had vertigo episodes from August 8 to August 15, 1997. The event that initiated the episodes happened on August 7, while the event that terminated them happened on August 12. Data about Sam's vertigo was entered into the database on October 15, 1997. On October 17, the physician discovered that information about vertigo episodes was incorrect and, later, on October 19, he became aware that vertigo episodes actually took place from August 10 to August 15, 1997. Finally, on October 21, the physician entered the correct data items into the database.

| P_id | symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|---|
| 1 | headache | [97Oct1, $\infty$) | 97Sept5 | null | [97Oct7, 97Oct15) | [97Oct10, 97Oct15) |
| 2 | vertigo | [97Aug8, 97Aug15) | 97Aug7 | 97Aug12 | [97Sept3, 97Oct17) | [97Oct15, 97Oct21) |
| 2 | vertigo | [97Aug10, 97Aug15) | 97Aug7 | 97Aug12 | [97Oct19, $\infty$) | [97Oct21, $\infty$) |
| 1 | headache | [97Oct1, 97Oct14) | 97Sept5 | 97Oct9 | [97Oct15, 97Oct20) | [97Oct15, 97Oct21) |
| 1 | headache | [97Oct1, 97Oct14) | 97Sept15 | 97Oct9 | [97Oct20, $\infty$) | [97Oct21, $\infty$) |

**Table 14.** Database instance of patient symptoms: the relation *pat_sympt*.

Suppose that we want to determine which data is available to the physician on October 18, according to the database contents on October 20. To satisfy such a request, we need to combine the `AS OF` and `AS KNOWN` clauses:

```
SELECT *
FROM pat_sympt S
AS OF 97Oct20
AS KNOWN 97Oct18
```

The result of query evaluation is given Table 15. Notice that data about vertigo is still present on October 20, but the availability time allows us to conclude that it has been "conceptually" deleted on October 17, and thus on October 18 the physician was aware that data about vertigo was incorrect.

| P_id | symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|---|
| 1 | headache | [97Oct1, 97Oct14) | 97Sept5 | 97Oct9 | [97Oct15, 97Oct20) | [97Oct15, 97Oct21) |

**Table 15.** Tuples of the database instance of Table 14 as of October 20, 1997, as known on October 18, 1997.

Further issues arise when more than one relation comes into play in a query. As an example, we have to assign a proper meaning to the cartesian product of the relations that appear in the `FROM` clause. A possible choice is to impose that temporal tuples of different relations can be joined only when both their transaction times and their availability times overlap. As for the result of the cartesian product, the language should allow the user to choose among different ways of executing the cartesian product between temporal relations (e.g., one can opt for the usual cartesian product of relations, where the temporal attributes are treated as the standard ones). Furthermore, besides the standard (atemporal) join, the language should support some forms of temporal join, where the join condition allows the user to join tuples on the basis of different temporal features.

Finally, when several temporal relations are involved in a query, the language must provide some mechanisms to allow the user to obtain a consistent result, that is, a temporal relation endowed with valid, transaction, event, and availability times. While transaction and availability times of the resulting relation can be obtained (according to the standard definition of the cartesian product given above) as the intersection of the availability times and the transaction times of the considered tuples, respectively, valid and event times must be explicitly defined by the user, even though the system can provide some default rules.

As an example, consider a scenario where the physician has to pair data about patient symptoms, given in Table 14, with data about prescribed therapies, reported in Table 16.

| P_id | therapy | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|---|
| 1 | aspirin | [96Oct1, 96Oct20] | 96Sept25 | 96Oct16 | [97Oct7,$\infty$] | [97Oct10, $\infty$] |
| 2 | paracetamol | [97Aug11, 97Aug12] | 97Aug11 | 97Aug12 | [97Sept3, $\infty$] | [97Oct15, $\infty$] |

**Table 16.** Patients' therapy database instance: the relation *pat_ther*.

*Example 8.* On the basis of Tables 14 and 16, the physician wants to determine the symptoms of patients for which the valid time intervals of symptoms and therapies overlap.

The query of Example 8 can be formulated as follows:

```
SELECT symptom, S.P_id
FROM pat_sympt S, pat_ther T
WHEN NOT(VALID(S) BEFORE VALID(T)) AND
      NOT(VALID(T) BEFORE VALID(S))
WHERE S.P_id = T.P_id
```

where `BEFORE(·)` is the well-known relation of Allen's Interval Algebra [1].

The `WHEN` clause can actually be replaced by a suitable temporal join in the `FROM` clause, as shown by the following equivalent formulation of the query:

```
SELECT symptom, S.P_id
FROM pat_sympt S TJOIN pat_ther T ON NOT(VALID(S) BEFORE VALID(T))
      AND NOT(VALID(T) BEFORE VALID(S))
WHERE S.P_id = T.P_id
```

In order to properly evaluate the query, some default criteria must be specified to determine the values of the valid and event times of the resulting relation, whenever the user does not provide any explicit rule. As an example, the valid time of each tuple belonging to the resulting relation can be defined as the intersection of the valid times of the corresponding tuples belonging to the relations that occur in the `FROM` clause, while its initiating and terminating event times can be defined as the maximum of the initiating and terminating event times of the corresponding tuples, respectively.

According to such a criterion, the result of the query of Example 8 is the relation of Table 17.

However, we expect that, in most cases, the user will explicitly define the way in which the valid and event times of the resulting relation must be computed, taking into account the meaning he/she assigns to the query and the relative data. As an example, in the case of Example 8, the user can formulate the following query:

```
SELECT symptom, S.P_id WITH VALID(S) AS VALID, INITIATING_ET(S)
      AS INITIATING, TERMINATING_ET(S) AS TERMINATING
FROM pat_sympt S TJOIN pat_ther T ON NOT(VALID(S) BEFORE VALID(T))
      AND NOT(VALID(T) BEFORE VALID(S))
WHERE S.P_id = T.P_id
```

| P_id | symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|---|
| 2 | vertigo | [97Aug11, 97Aug12] | 97Aug11 | 97Aug12 | [97Oct19, $\infty$] | [97Oct21, $\infty$] |

**Table 17.** Patient symptoms that overlap therapies.

In such a way, the user assigns to the valid and event times of the resulting relation the values that these attributes have in the *pat_sympt* relation. Indeed, the valid time of the tuple belonging to Table 18 is the time interval during which Sam suffered from vertigo episodes, while the valid time of the tuple belonging to Table 17 represents the time interval during which both vertigo episodes and therapeutic actions occurred.

| P_id | symptom | VT | $ET_i$ | $ET_t$ | AT | TT |
|---|---|---|---|---|---|---|
| 2 | vertigo | [97Aug10, 97Aug15] | 97Aug7 | 97Aug12 | [97Oct19, $\infty$] | [97Oct21, $\infty$] |

**Table 18.** Patient symptoms that overlap therapies (revisited).

We conclude this section by observing that further specific clauses, which have already been proposed in the literature to explicitly support valid and transaction times in bitemporal databases [23, 7], such as `TIME SLICE` and `MOVING WINDOW`, can be easily generalized to take into account also event and availability times (cf. [30]).