University of Florida

Computer and Information Sciences

A Retrospective Analysis of Time Concepts in Temporal Databases

Seung-Kyum Kim and Sharma Chakravarthy

email: {skk,sharma}@cis.ufl.edu

UF-CIS-TR-92-044
(Submited for publication)

(This work was supported in part by the NSF Research Initiation Grant (IRI-9011216) and by a Grant from the Florida High Technology and Industrial Council (UPN# 900911013))

Department of Computer and Information Sciences
Computer Science Engineering Building
University of Florida
Gainesville, Florida 32611

Abstract

In this paper, we propose a formal definition of temporal validity which we believe is the first and the most comprehensive quantification of that notion. The temporal validity is developed into the notion of sort of validity with which the confusion among various time concepts introduced for temporal databases can be dispelled. Further, we discuss the problem of preserving multiple past states of a temporal database, which leads to the identification of a maximal set of time concepts and multidimensional representations. It is shown that the time concept, event time is needed to properly model retroactive and proactive updates, as it is not possible to model them using the valid and transaction times as thought earlier. We also show the adequacy of three time concepts (valid time, event time and transaction time) for completely preserving different past states generated by retroactive and proactive updates, error corrections, and delayed updates.

1 Introduction

The temporal database which incorporates time concept into the conventional database has been investigated by many researchers over a decade [Soo91]. The most predominant direction of the previous research has been the extension of the relational data model to a temporal one and/or the extension of the relational query languages SQL and Quel to temporal versions. Work on this area includes [CT85, Ari86, Sno87, Gad88, NA89, Sar90, SS91]. Besides the emphasis on the relational model, recently temporal extensions to the ER (or semantic) model have begun to emerge in the literature [Klo83, EW90, SS91, TL91]. The formalization of temporal data model [Ser80, CW83, TC90, GM91] and the physical data organization for temporal databases [LDE+84, SK86, Ahn86, EWK90, EJK92] have also been investigated.

In our view, earlier attempts at clarifying various time concepts introduced for enriching the semantics of temporal databases have only added more confusion. Early on, Snodgrass and Ahn [SA85] proposed three time concepts, valid time, transaction time, and user-defined time, based on the classification of temporal databases into rollback databases, historical databases, and temporal databases. However, the confusion concerning time concepts still persists and leads to the following problems:

1. Classification of time concepts into a dichotomy of reality vs. representation (or real-world vs. database-world).
2. Employing inappropriate time concepts to capture retroactive and proactive updates.

In this paper, we develop the notions of *temporal validity* and *sort of validity* to clarify (1). With these notions we are able to infer that there is no absolute real-world validity but everything is interpretation-dependent. Also we show that there may be multiple interpretations for temporal behavior (validity) of a class of data objects. This implies that the notion of valid time, irrespective of its name, is enough only as far as the validity is concerned.

As another aspect of time concept, we demonstrate that valid and transaction times are not sufficient to model retroactive and proactive updates (problem (2)), while they are still adequate for error corrections and delayed updates. In fact, the event time bearing a revised meaning is required to capture, in conjunction with the valid time, the precise meaning of retroactive and proactive updates. We discuss this issue in the context of preservation of multiple past states and argue that the preservation problem is independent of the temporal validity.

Combining the previous arguments, we conclude that only valid time is needed for a single sort of validity (one interpretation) and multiple valid times can be employed if multiple interpretations of temporal validity are required. Apart from this, the valid time needs to be augmented to two- or three-dimensional time by incorporating the event time and/or the transaction time to preserve multiple past states of databases. In summary, the valid, event, and transaction times are practically a maximal set of times needed to completely model the temporal database.

The rest of this paper is organized as follows. In section 2, we give a critical review of previous work dealing with time notions. In section 3, the notions of temporal validity and sort of validity are presented. In section 4, we discuss the problem of preserving multiple past states. Finally, in section 5, we make concluding remarks about our work.

2 Problems with previous work

Of numerous time concepts proposed to date, we select some and highlight the confusion among them.

Copeland and Maier [CM84] introduced two types of time, *event time* and *transaction time*. According to their description, the event time is the time an event, which gives rise to a change

to an object, happens in real world. And the transaction time is the time the change is recorded in a database. Although not explicitly mentioned, to our best comprehension, it seems that they were trying to explain two kinds of temporal validity. One is a real-world validity modeled by the event time. The other, an approximation of the former, is a database-world validity modeled by the transaction time. For example, let's assume that a faculty member, Smith was promoted to associate professor in Aug. 1985, and the fact was recorded in a database in Sep. 1985. If we adopt the event time, Smith's associate professorship becomes valid from Aug. 1985. If we take the transaction time, as they actually did, it becomes valid from Sep. 1985. We will call, tentatively, the former validity *effective validity* and the latter *registration-based validity.*

Lum and others [LDE+84] proposed other kinds of time, *logical time* and *physical time.* Basically, the logical time is the same as the event time in the sense that both are time of real world, i.e., the true time with which an object changes. However, the physical time is different from the transaction time of [CM84], even though the two times have been treated identically as both are the recording time of data. The physical time was intended to be a reference time for the logical time to model retroactive and proactive updates. For example, suppose Smith received a promotion to associate professor *in* Aug. 1985, but the promotion was retroactive from Jan. 1985. Then, the fact is recorded with logical time of Jan. 1985 and physical time of Aug. 1985 if indeed recorded then. We should be able to notice the problem with this approach. *What if the fact were recorded in Sep. or later instead of Aug.*? Does that mean Smith received a retroactive promotion in Sep. or later? In addition, the validity dealt with in [LDE+84] is in essence the same with the effective validity of [CM84], provided that the event time and the logical time are for modeling of the same real-world validity. The difference is that in [LDE+84] they used an additional time to capture the semantics of retroactive and proactive updates. We will discuss this problem further in section 4.

The *valid time* and *transaction time* proposed by Snodgrass and Ahn are also confusing [SA85]. They distinguished the two times based on the classification of temporal databases into rollback databases, historical databases, and temporal databases[1]. That is, the valid time is the time

[1]This "temporal database" has a narrow meaning in their work that it is a database made by consolidating the rollback database and the historical database. This term should not be confused with our general meaning of temporal

employed by the historical database, which is described as a database storing history of relations as it is best known. The transaction time is described as the time employed by the rollback database, which is a collection of all the snapshot databases. The historical and rollback databases are distinguished by error corrections also. That is, errors in historical databases can be rectified, whereas such corrections are not allowed in rollback databases. Here, we can see that the distinction between historical databases and rollback databases is not so clear in the sense that a collection of snapshot databases can be regarded as a history of relations. Further, prohibiting error correction in rollback databases is not meaningful because erroneous database states should be able to get corrected anyhow. Accordingly, the definitions of valid and transaction times as attributes to historical and rollback databases are also unclear.

On the other hand, the role of transaction time in the rollback database is different from that of their temporal counterpart, which employs both valid and transaction times to support retroactive and proactive updates as well as preservation of all past database activities. The transaction time in rollback databases is equivalent to the transaction time of [CM84] in that rollback databases are in fact temporal databases using the registration-based validity. In contrast, the transaction time in their temporal databases is the same with the physical time of [LDE+84] in the sense that both times are used to model retroactive and proactive updates.

Also, it is noticeable in [SA85] that the user-defined time was treated just as a data type requiring only input/output functions. However, as we shall show in section 3, the user-defined time models another validity just as effective or registration-based validity.

Yet another confusion arises in the work of Gadia and Yeung [GY88], in which the valid and transaction times of [SA85] were adopted for the demonstration of their n-dimensional, symmetric time concept. However, the valid and transaction times of [GY88] were in fact not those of [SA85], but the event and transaction times of [CM84], respectively.

To summarize, we identify several problems with the previous approaches. First, we think there has been a dichotomous classification of time concepts, i.e., reality vs. representation (or real-world

database.

vs. database-world) [SA85]. That dichotomy, together with the absence of a proper quantification of temporal validity, presumably misled us to the point that there are one genuine real-world validity and other possible database-world validities, if any, such as the registration-based validity. Second, we can see that the problem of preserving multiple past states has been intermingled with the validity problem. They need to be separated. Lastly, unlike the previous view [LDE+84, SA85], combination of valid and transaction times cannot model retroactive and proactive updates.

3 Temporal validity

In this section, we present the notions of *temporal validity* and *sort of validity*. With these notions, the validity problem partially discussed in the previous section can be clearly understood.

Throughout this paper we assume an equi-distant discrete time domain T which is used to represent the temporal validity and is isomorphic to the Gregorian calendar time domain. We call the time domain valid time domain and call time in the domain valid time. A time interval is defined to be consecutive time points in T. *Now* represents the current time and ∞ denotes a sufficiently remote future time point.

3.1 A simple data model

Before presenting the formal definition of temporal validity, we first define a simple data model from which some necessary concepts are drawn. The simple data model is not a complete one, its sole objective being to give a set-theoretic definition of data value.

Object An object is an entity or a relationship among entities.

Object set An object set is a set of objects. However, not all arbitrary object sets are meaningful to us. We are interested in only certain object sets which are pre-classified from the universal object set, consisting of all conceivable objects in the world, according to some criteria.

For example, we may have three object sets, S_1, S_2, and S_3, each of which is a subset of the universal object set U. The subsets might be pre-classified and named Person, Faculty, and Furniture, respectively. Then, an unrelated object, such as a stone, will not be included in set S_1.

We call such a name a *property* of the object set. From now on, whenever we say an object set we mean an object set with property. Also, it is assumed that object sets are pre-classified or identified by database designers.

Name and value A name is regarded in our work as an identifier for a property. Accordingly, two different properties have different names. However, an object set may happen to have several properties, each of which is identified by a distinct name. Further, we do not make any distinction between name and data value in the simple data model. A value such as 2 may be treated as a name, and a name such as Smith treated as a value.

Property The definition of property is given below.

- A property P of a given object set A is a subset (including empty set) of A, i.e., $P \subseteq A$.[2]
- An object e, $e \in A$, has a property P if $e \in P$.

Property set A *property set*, $\mathcal{P}$ is a meaningful set of properties that can be associated with a given object set A.

A property set is called a *disjoint* property set when no element (property) in the set overlaps the others. That is, if $e \in P_i$, then $e \notin P_j$, for any $P_i, P_j \in \mathcal{P}$ such that $i \neq j$. If a property set is not disjoint, it is called an *overlapping* property set.

For example, given an object set, Person = $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, three properties can be defined as follows.

Faculty = $\{e_1, e_2, e_3, e_4\}$, Student = $\{e_5, e_6\}$, *Unknown* = $\{e_7\}$.

The property Faculty (an object set) is further specialized into two ways as follows.

Instructor = $\emptyset$, Assistant = $\{e_1\}$, Associate = $\{e_2, e_3\}$, Full = $\{e_4\}$, Tenured = $\{e_3, e_4\}$.

Jane = $\{e_1\}$, Smith = $\{e_2\}$, John = $\{e_3\}$, Miller = $\{e_4\}$.

[2]To be more specific, property P is a name of a subset $A' \subseteq A$. However, such a subtle distinction between a name of the set and the set itself is not so critical as long as the name is an identifier of the set.

The object set Person has a property set, $\mathcal{P}_{Occupation}$ = {Faculty, Student, *Unknown*}, while the object set Faculty has two property sets, $\mathcal{P}_{F\_Rank}$ = {Instructor, Assistant, Associate, Full, Tenured} and $\mathcal{P}_{F\_Name}$ = {Jane, Smith, John, Miller}.[3] An object e_2 in object sets, Person and Faculty, has three properties, Faculty, Associate, and Smith, each of which belongs to property sets, $\mathcal{P}_{Occupation}$, $\mathcal{P}_{F\_Rank}$, and $\mathcal{P}_{F\_Name}$, respectively. $\mathcal{P}_{F\_Name}$ is a disjoint property set, whereas $\mathcal{P}_{F\_Rank}$ is an overlapping one. Actually, a property set corresponds to a domain of attribute in terms of the traditional definition.

3.2 Definition of temporal validity

Informally, the temporal validity of a data value for a given object is the object's possession of the value along the time dimension. For example, the validity of associate professorship for Smith's rank is a state of whether or not Smith possesses the associate professorship with time.

Temporal validity The *temporal validity* of property P in an object set A is a *characteristic function* of P,

$$\mathcal{V}_P : (A \times T) \rightarrow \{0, 1\}$$

given by

$$\mathcal{V}_P(e,t) = \begin{cases} 1 & \text{if } e \in P \text{ at } t \\ 0 & \text{if } e \notin P \text{ at } t \end{cases}$$

where e is an object in A, t is a time point in the time domain T, P is in $\mathcal{P}$, and $\mathcal{P}$ is a property set of A.

If (e,t) is mapped onto 1 by a characteristic function $\mathcal{V}_P$, then property P is said to be *valid* for object e at time t by $\mathcal{V}_P$. It is assumed that for each property P_i in property set $\mathcal{P}$ of object set A, there exist one or more characteristic functions, such as $\mathcal{V}_{1P_i}$, $\mathcal{V}_{2P_i}$, and so on.

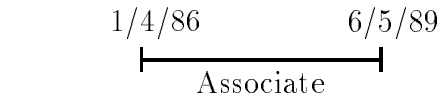
The use of a characteristic function provides a flexible mechanism for defining temporal validity. Noted that in a characteristic function the way to decide if e is in P does not matter, as far as the temporal validity is concerned. The only necessary assumption is that such a function does exist.

[3] It should be noted that the property Faculty (also an object set) is different from the property set $\mathcal{P}_{F\_Name}$ denotationally, as well as semantically. That is, the former is denoted by $\{e_1, e_2, e_3, e_4\}$ while the latter by $\{\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}\}$.

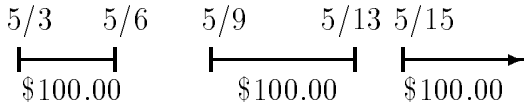
Valid period A *valid period*, τ of property P for object e, characterized by a characteristic function $\mathcal{V}_P$, is defined to be a set of time points t, $t \in T$, such that

$\{t \mid \mathcal{V}_P(e,t) = 1\}$.

In other words, a valid period is a time interval or a set of disjoint time intervals over which a given property is valid. A valid period can be pictorially represented as shown in Figure 1. Figure 1(a) shows a valid period of a property, associate professorship for an object, Smith, while Figure 1(b) shows a recurring valid period of a property, \$100 (which is in a property set, Stock-price) for an object, say IBM stock.



(a)



(b)

Figure 1: Valid periods

Due to the possibility of multiple characteristic functions for a property, an object may have several different valid periods for a property. To exhibit this aspect, we define two characteristic functions, $\mathcal{V}_{1P_1}$ and $\mathcal{V}_{2P_1}$, for a property P_1, where P_1 is *associate professorship*.

$\underline{\mathcal{V}_{1P_1}(e,t)}$:

1. Map to 1 if e receives a promotion letter to associate professor and t is equal to the promotion *date written on the letter*.
2. Map to 0 if e receives a promotion letter to other rank than associate professor or a letter of dismissal, and t is equal to the date written on either letter.

3. Map to 1 if e is an associate professor at $t-1$ and t is different from the date of (2) if e gets either letter.

$\underline{\mathcal{V}_{2P_1}(e,t)}$:

1. Map to 1 if e receives a promotion letter to associate professor and t is equal to the *date the letter was signed.*
2. Map to 0 if e receives a promotion letter to other rank than associate professor or a letter of dismissal, and t is equal to the date either letter was signed.
3. Map to 1 if e is an associate professor at $t-1$ and t is different from the date of (2) if e gets either letter.

For the two different characteristic functions shown above, we may have different valid periods of Smith's associate professorship as shown in Figure 2.

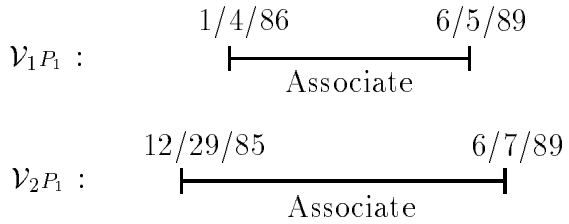


Figure 2: Two different valid periods of associate-professorship

3.3 Sort of validity

While the temporal validity deals with each individual property, the sort of validity concerns the totality of all properties in a property set. A sort of validity is, in effect, an interpretation of temporal behavior of data objects.

Definition: Given a disjoint property set, $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ and an object set A, a *sort of validity*, $\mathcal{S}_\mathcal{P}$ for $\mathcal{P}$ is defined to be a set of characteristic functions,

$$\mathcal{S}_\mathcal{P} = \{\mathcal{V}_{i P_1}, \mathcal{V}_{j P_2}, \cdots, \mathcal{V}_{k P_n}\}$$

such that if $\mathcal{V}_{l P_m}(e,t) = 1$, $\mathcal{V}_{l P_m} \in \mathcal{S}_\mathcal{P}$, then for all $\mathcal{V}_{u P_w} \in \mathcal{S}_\mathcal{P}$, $\mathcal{V}_{u P_w} \neq \mathcal{V}_{l P_m}$, $\mathcal{V}_{u P_w}(e,t) = 0$.

As shown in the definition, for a set of characteristic functions to be a legitimate sort of validity, a couple of conditions are required to be satisfied.

1. At most one characteristic function maps (e, t) onto 1.
2. The property set $\mathcal{P}$ is disjoint.

Condition (1) is necessary to prevent an object from having more than one property simultaneously, which belong to a property set. For example, in general it does not make sense for a faculty member to have two different ranks, say, associate and full, at the same time. In fact, this condition is the minimum requirement that the characteristic functions defined should satisfy.

Condition (2) is imposed because if the property set is indeed overlapping, the first condition cannot be satisfied semantically. For example, if the property set is $\mathcal{P}_{F\_Rank}$ defined previously, an object with property Tenured most likely possesses another property, Associate or Full, too. Thus, if the characteristic functions are defined so as to fulfill the first condition over such an overlapping property set, the resultant validity will be semantically wrong. The second condition is actually a necessary condition for the first one.

In addition, it should be noted that none of the characteristic functions in a sort of validity may map (e, t)'s to 1. In that case, it is assumed that the object e does not exist at time t.[4]

Single-sort/multisort validities The definition of sort of validity and the possibility of multiple characteristic functions for a property lead to multiple sorts of validity. For example, assume that given a property set $\mathcal{P} = \{P_1, P_2\}$, four characteristic functions, $\mathcal{V}_{1P_1}$, $\mathcal{V}_{2P_1}$, $\mathcal{V}_{1P_2}$, and $\mathcal{V}_{2P_2}$ are defined. Then, there may be four sorts of validity out of the characteristic functions, provided that each combination satisfies the conditions of sort of validity. Two of them are given below.

$$\mathcal{S}_{1\mathcal{P}} = \{\mathcal{V}_{1P_1}, \mathcal{V}_{1P_2}\}$$
$$\mathcal{S}_{2\mathcal{P}} = \{\mathcal{V}_{2P_1}, \mathcal{V}_{2P_2}\}$$

When an object set has multiple sorts of validity, as shown above, we say the object set has *multisort* validity. If only one sort of validity is defined, the object set is said to have a *single-sort*

[4]This assumption might be an over-simplification in that the existence of an object does not depend on the fact whether the object has a certain property. The simplification, however, keeps the discussion manageable at the expense of modeling accuracy.

validity. For example, if we define $\mathcal{V}_{1P_2}$ and $\mathcal{V}_{2P_2}$ in a similar manner to $\mathcal{V}_{1P_1}$ and $\mathcal{V}_{2P_1}$, respectively, where P_2 is *full professorship*, then we may have two different sequences of valid periods for two sorts of validity, $\mathcal{S}_{1\mathcal{P}}$ and $\mathcal{S}_{2\mathcal{P}}$, as shown in Figure 3.

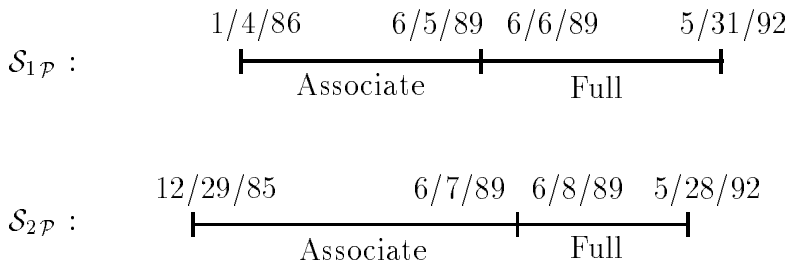


Figure 3: Two sorts of validity

On the other hand, either of the remaining combinations, $\{\mathcal{V}_{2P_1}, \mathcal{V}_{1P_2}\}$ and $\{\mathcal{V}_{1P_1}, \mathcal{V}_{2P_2}\}$ are not likely to be a sort of validity since as shown in Figure 4 during some time period an object may have more than one property at the same time.

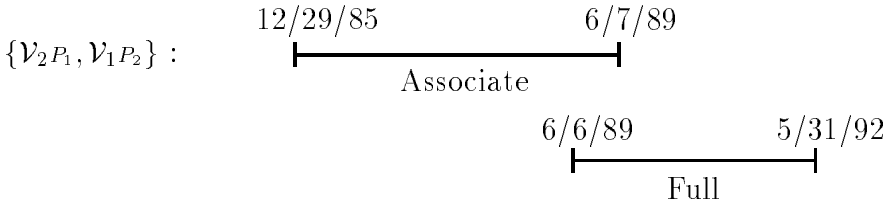


Figure 4: A disallowed combination of characteristic functions

Lastly, we want to emphasize that the sort of validity (as well as the temporal validity) is developed for an object set. In other words, a sort of validity defined for an object set, in general, does not apply to other object sets. For example, the sort of validity $\mathcal{S}_{1\mathcal{P}}$ is not meaningful for an object set Furniture.

4 Preservation of multiple past states

As the temporal database is intended to capture all past database states including the current one, preserving past states is essential. However, as we shall see, the preservation of past states is not

guaranteed by just not deleting previously entered data. There are two independent causes that have a bearing on the preservation. The first is the retroactive update, and the second is the error correction and delayed update. The reason why the preservation can be incomplete is that these operations generate multiple pasts. If a temporal database system cannot record and retrieve all the past states, we will get to lose some past states, that is, the preservation of the past will be incomplete.

[Sno87] is regarded as the first comprehensive treatment of the preservation of past states, although the aspect of retroactive update was mentioned earlier in [LDE+84]. A common problem in their work is that they made use of the transaction time (the physical time) to deal with retroactive and proactive updates. In this section, we show that retroactive and proactive updates cannot be captured by the transaction time, and define another time concept, event time to model these operations.

4.1 Event time

An event can be viewed as an abrupt change[5] of database state(s). In our work the event has a slightly tailored meaning that it generates certain relevant facts, or more specifically, it is a cause of changes to properties of database objects. Thus, when an event happens, it is naturally assumed that the event changes certain properties of database objects. For example, a promotion event for a faculty member generates a new fact, i.e., a new rank, and a relocation event generates a new address. In other words, the two events change properties of a faculty member, to new ones. Now we are interested in the time when such an event happens. We call the time an event happens *event time*.

Incidentally, it should be pointed out that a fact (property) is not necessarily generated by only one event. That is, two or more different events can generate the same fact. For example, an address, Gainesville, may be generated by an event of district adjustment such as annexation, as well as an ordinary relocation event. As a matter of fact, the heterogeneity of events may interfere with the interpretation of events recorded in a database. However, we do not pursue that issue any

[5]Such a change need not be an instant one. An event may happen through a fairly long period of time. Or a seemingly instant event may be viewed as a long term one if the time granularity is sufficiently magnified.

further in our current work. We simply assume that all events changing a property of an object are of the same kind.

4.1.1 Time difference between an event and its accompanying facts

Although it is preferred that facts generated by an event become valid as soon as the event happens, there may be situations in which the time a fact becomes valid differs from the time the event happens (i.e., event time). Based on this time difference, we can classify events into three categories, on-time events, retroactive events, and proactive events. When a database is updated with a fact generated by a retroactive event, we call the update a *retroactive update*. Similarly, if a fact is generated by a proactive event, we call the update a *proactive update*.

4.1.2 Classification of events

On-time events This is the most general case. For an event classified into this category, its relevant facts become valid immediately after the event happens. Alternatively, it may be thought of as facts become valid simultaneously with the event. For example, if a promotion event which promotes a faculty member Smith to an associate professor happened on 11/89, unless otherwise stated, his rank of associate professor would become valid on 11/89.[6]

Retroactive events For some events, the time a fact becomes valid may be prior to the time at which the fact is generated, i.e., the time an event generating the fact occurs. We call such events *retroactive events*. For example, Smith's promotion might be a retroactive one so that the promotion became valid from 8/89 while the promotion itself was proclaimed in 11/89. The time difference is depicted in Figure 5(a). It should be noted that even though the time the fact becomes valid precedes the time of the event, the fact is never known until the event occurs.

Proactive events In contrast to retroactive events, a fact may become valid at a later time than the time at which the event generating the fact occurs. We call such events *proactive events*. As an example, when it is decided that a research grant is to be awarded to an institute (i.e., an event

[6]Note that the interpretation concerning when the event happened is determined by a characteristic function discussed in section 3.

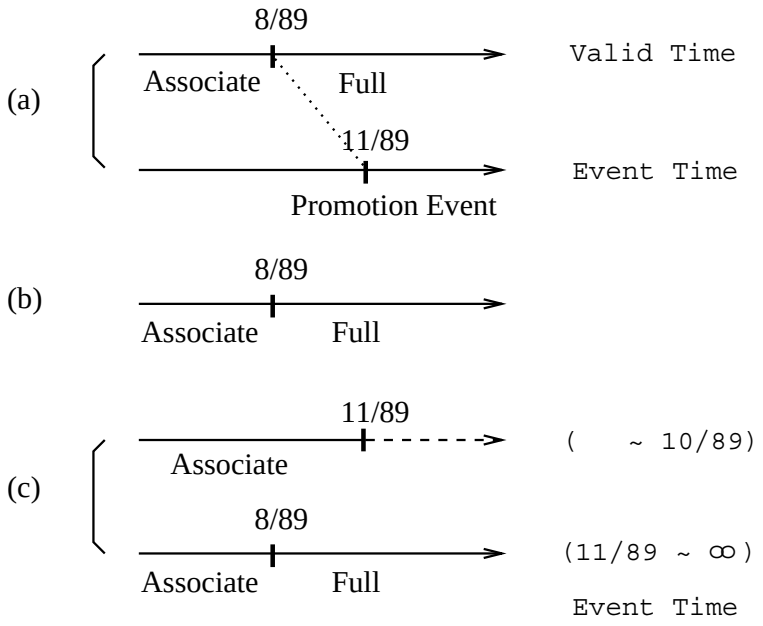


Figure 5: Retroactive event

has now happened), the grant usually will not be effective until a later time. The time difference for the proactive event is shown in Figure 6. Other examples for the proactive event may be drawn from a temporal database for weather forecasing. Whenever a forecasting is made (or a fact is generated), the fact will not be valid until some time later.

4.1.3 Multiple pasts generated by the time difference

The time difference inherent to a retroactive event is one reason for the generation of multiple pasts. Figure 5(a) depicts a retroactive promotion of Smith. Until 8/89, Smith's rank is no doubt Associate. After the promotion on 11/89, his rank is Full. But, what about from 8/89 to 10/89? Before the promotion, the rank during that period was Associate, whereas after the promotion, the rank was revised to Full. As a result, we have two different pasts during that time period. Figure 5(b) and (c) show possible representations of the retroactive update.

Figure 5(b) represents the resultant database state after the update, and is in fact the most

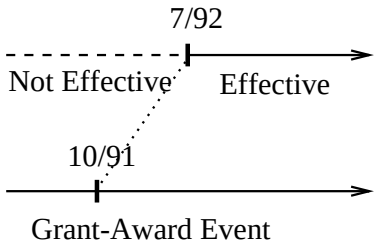


Figure 6: Proactive event

reasonable representation for one-dimensional temporal databases. A shortcoming of this representation is that the intermediate database state, Associate, during 8/89 through 10/89 disappears and cannot be retrieved. Moreover, the fact that the promotion was retroactive cannot be recalled.

Figure 5(c) shows an alternative in which two histories are kept, one each for the updated and the previous state. In this approach, the updated state can be retrieved as usual, and the previous state also can be retrieved if necessary. In addition, the fact that the recorded event was a retroactive one can be represented in some way and recalled.

A problem with the second approach shown in Figure 5(c) is that *one* time (or one-dimensional time) is not sufficient to uniquely qualify a data value at a given time point. For example, at 10/89 Smith's rank may be interpreted as either Associate or Full. However, taking a closer look at the multiple histories, we can see each history has its own *disjoint* effective period in terms of event time. For example, until 11/89 in event time, only the history of the upper figure in Figure 5(c) will be effective. Such an event time period is shown on the right side of each history. As such, the event time is now able to be a qualifier to single a history out of multiple histories. Once a history is selected, then we can examine it to get a data value at a specific time point. Consequently, if two time values, one each in the valid time and the event time, are given, we are able to retrieve a unique data value from a temporal database without losing past states generated by the time difference of retroactive events.

4.2 Transaction time

In section 2, we discussed the meaning of transaction time and the different roles it played in temporal databases. The transaction time, in our work, is simply the time at which data values (facts, properties) are recorded in a database and its role is just to supplement the valid time to

preserve multiple pasts generated by error corrections and delayed updates.

4.2.1 Error corrections

The error correction operation may be the most conspicuous one among those operations generating multiple pasts. Assume that Smith was promoted to associate professor on 1/85, but the rank was wrongly recorded as full professor. The error was found later and corrected on 9/85. Figure 7(a) illustrates that situation. The error correction induces two different pasts. During 1/85 through 8/85 we have an incorrect past in which Smith's rank is viewed as full professor, whereas from 1/85 up to now we have a correct past[7] where the rank is viewed as associate professor. Figure 7(b) and (c) show two possible representations for the correction.

Figure 7(b) represents the corrected, most up-to-date state of database, but nothing else. What we lost in the representation is the incorrect past, i.e., rank of full professor during 1/85 through 8/85, which cannot be retrieved after the correction. Moreover, there is no way to recall the fact that an error was corrected, unless it is stored somewhere else. For many applications, this loss of past information may not matter at all. However, applications requiring strict audit need to keep even erroneously created states.

Figure 7(c) shows an alternative, similar to Figure 5(c), where both pasts are maintained. With this approach we can retrieve the incorrect past information if needed, in addition to the corrected one. Also, we can discern whether and when an error correction was made. Again, one-dimensional time is not sufficient to uniquely qualify a data value at a given time point. For example, on 5/85 Smith's rank may be construed as either Full or Associate.

4.2.2 Delayed updates

As a fact is generated in real world and recorded in a database, there exists inevitable time delay between the generation time and the recording time. This delay may vary from a few milliseconds to several months or more. Due to the delay we, by no means, can avoid temporal inconsistency or inaccuracy between real world and a modeled database-world, as long as information is retrieved

[7] Note that the currently correct past is not necessarily indeed correct. It may be corrected again later.

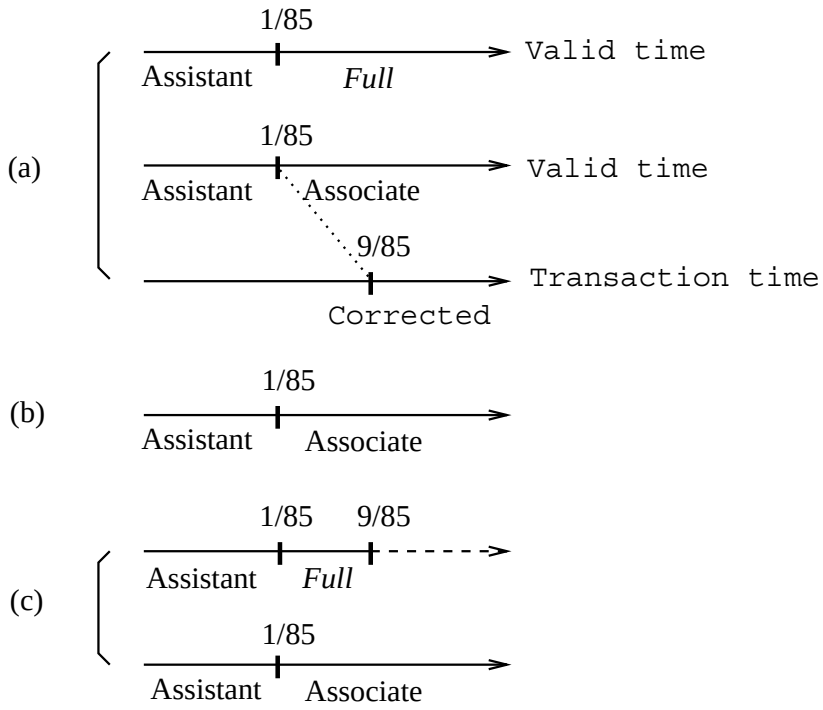


Figure 7: Error correction

from the stored data. Suppose that Smith was promoted to full professor in 8/89 and the fact was not recorded until 11/89. Then, until 11/89 we would never know the fact of his promotion, no matter how ingenious methods were employed. The best we can do is to either minimize such a delay or keep all the past incorrect database states so as to be brought back whenever an inspection or audit is necessary.

Figure 8 shows such a delayed update. An event (or a fact) happened in 8/89, which promoted Smith to full professor, is recorded in 11/89. As a result, during 8/89 through 10/89 the database keeps an out-dated data value, Associate. When one records the delayed update as if the event has been occurred just now, the resultant valid period will be Figure 8(b). The drawback of this approach is obvious. The discrepancy between the intended valid period and the valid period of stored data will be perpetuated.

Figure 8(c) is an alternative in which the fact is recorded as valid from 8/89 even if the recording is carried out in 11/89. This is similar to the approach taken in error correction shown in Figure 7(b). An advantage of this approach is that once recorded, the resultant database correctly reflects the intended validity. However, as in the case of error correction, we will lose an incorrect past state of

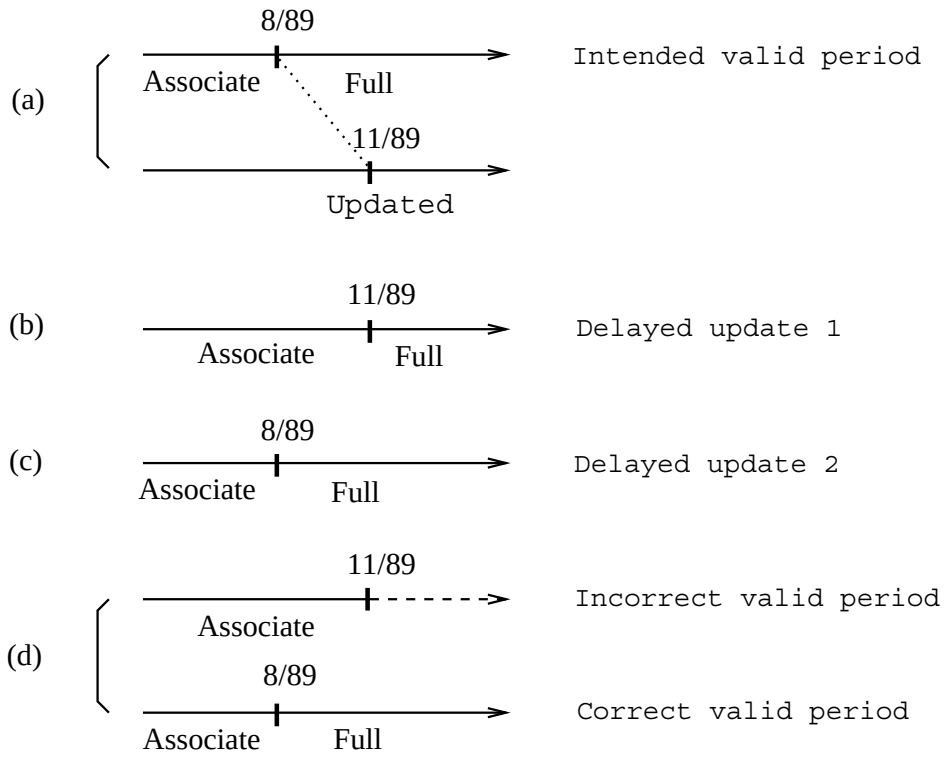


Figure 8: Delayed update

the database; i.e., the incorrect state during 8/89 through 10/89 will disappear after the update.

Figure 8(d) presents another alternative, in which we have two histories. We keep the incorrect past state as well as the correct one. By doing so, we are able to alleviate, to a large extent, the problem of delayed update. Until the delayed data is recorded, we cannot help getting incorrect information from the database. However, once it is recorded, we will be able to retrieve the correct one. In addition, we can always access and roll back to the incorrect past state. But, analogous to the previous cases, one-dimensional time is not sufficient in this approach to uniquely qualify a data value at a given time point. For example, in 9/89 Smith's rank may be interpreted as either Associate or Full.

4.2.3 Consolidation of error correction and delayed update

Even though the error correction and the delayed update may seem to have slightly different temporal behavior, they have one thing in common. That is, the time of error correction and the time of (delayed) update are the transaction time. Needless to say, the time when an error is

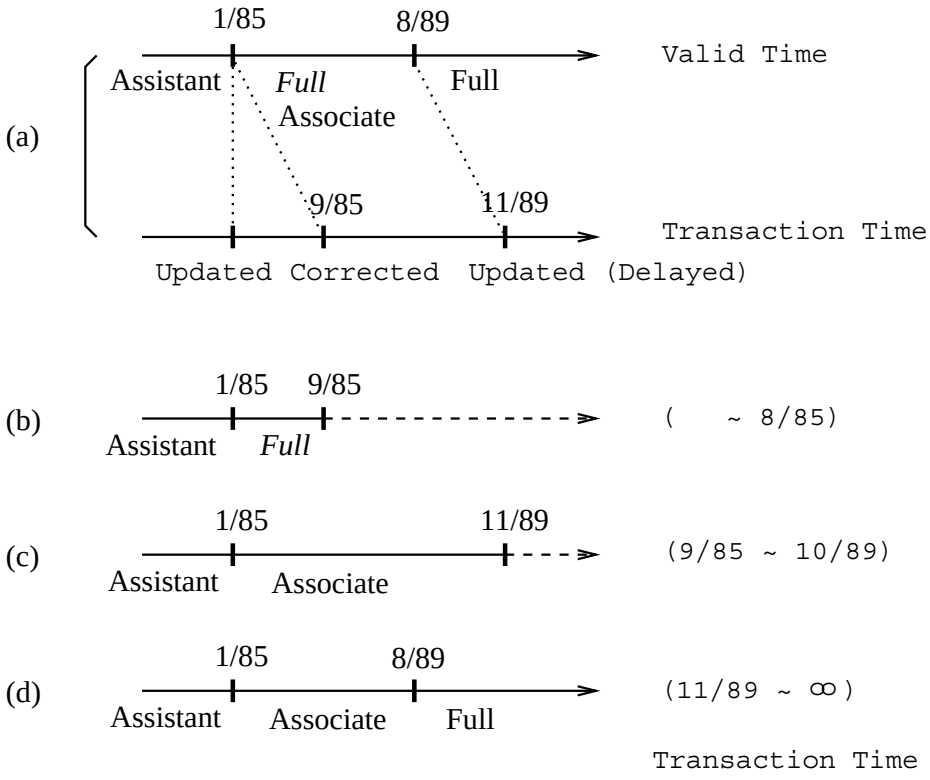


Figure 9: Consolidation of error correction and delayed update

corrected or a data value is updated is the time when either operation is performed on the database. This sharing leads to consolidation of error correction and delayed update over the transaction time, as depicted in Figure 9(a).

Figure 9(a) shows a series of update, correction, and delayed update operations. In 1/85, the rank was updated to a wrong value, Full. The wrong value was corrected to Associate in 9/85. Again, the rank was changed to Full in 8/89, but recording of the value was delayed until 11/89. Figure 9(b), (c), and (d) show multiple histories generated by the operations. Figure 9(b) and (c) reflect the effect of the correction made in 9/85. Before the correction the value (rank) is viewed as Full. After the correction the value is rectified to Associate as shown in Figure 9(c). The effect of the delayed update is reflected in Figure 9(c) and (d). Before the update at 11/89, the data value is wrongly seen (from 8/89) as Associate. After the update a new history shown in Figure 9(d) is obtained.

Similarly to the earlier discussion on event time, in Figure 9 we can see that each history has its own *disjoint* effective period in terms of transaction time. For example, during 9/85 through

10/89 in transaction time, only the history of Figure 9(c) will be effective. Such a transaction time period is shown on the right side of each history. Thus, if two time points, one in the valid time and the other in the transaction time are given, a single data value can be retrieved without losing past database states generated by error corrections and delayed updates.

4.3 Orthogonality of event time and transaction time

It is interesting to see in previous work [SA85, Sno87] that the retroactive update and the delayed update have been treated in the same way; in other words, the two were not differentiated at all. In their work, the time a fact is generated and the time it is recorded are indistinguishable and represented by the transaction time. In this section, we have established the difference between the two operations and shown why the notion of event time is required to model retroactive and proactive updates.

To comprehend the inadequacy of transaction time, let's compare the examples shown in Figure 5 and Figure 8. In the first case, Smith got a promotion on 11/89, which was retroactively effective since 8/89. In the second case, he got an on-time promotion in 8/89, but recording of the fact was procrastinated until 11/89. It is obvious that these two cases, a retroactive event and an on-time event but record-delayed, are not the same. Note that Figure 5(b) and Figure 8(c) are seemingly identical although they have different semantics.

More dramatic example can be shown as follows. Consider the proactive event of Figure 6 in which a research grant is awarded in 10/91 and the grant is effective from 7/92. And assume that we are trying to model the event with valid time and transaction time. If the proactive event happens to be recorded on 10/91, it might be regarded as a correct recording for the proactive event. However, if the recording time happens to be 12/91, 7/92, or 10/92, then the recorded event will be wrongly interpreted as another proactive event, on-time event, or even retroactive event, respectively. Figure 10 illustrates that situation.

A more general situation is illustrated in Figure 11. We have four events, E_i, E_j, E_l, and E_k, which are, respectively, an on-time event, a retroactive event, another on-time event, and a proactive event. Facts generated by the events are P_i, P_j, P_l, and P_k, respectively. The events

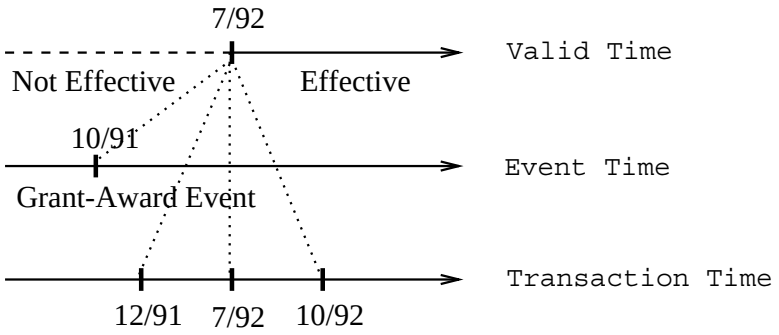


Figure 10: Misinterpretation of a proactive event

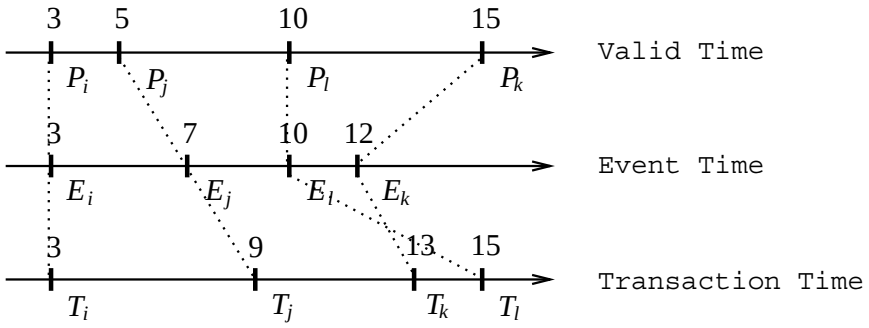


Figure 11: A general situation of event-occurring and its recording

are recorded by transactions, T_i, T_j, T_l, and T_k, respectively. Integers represent time points. Note that T_l is an error correction that adds a missing event E_l, which happened prior to E_k, to the database.[8]

4.4 Classification and maximal set of time concepts

Based on the discussion so far, we can classify the previously proposed time concepts using the notion of sort of validity and the supplemental time concepts needed to preserve multiple past states.

As shown in Figure 12, the event time of [CM84], the logical, valid, and user-defined times are in essence the same time in the sense that they all represent an arbitrary single sort of validity; the sort of validity is determined by characteristic functions, or more practically, by users' needs.

The transaction time of [CM84] and that of [SA85] used in rollback databases represent the

[8] As mentioned before, we regard all the events as being the same kind, such as events of promotion. Especially, E_l should not be misunderstood as an event of error-discovering. Such an event, bearing a different semantics, should have occurred somewhere between time points, 10 and 15.

Reference	Time concept	Representing sort of validity
[CM84]	Event	any sort of validity
	Transaction	registration-based validity
[LDE+84]	Logical	any sort of validity
	Physical	none (reference time for logical time)
[SA85]	Valid	any sort of validity
	Transaction in rollback DB	registration-based validity
	Transaction in temporal DB	none (reference time for valid time)
	User-defined	any sort of validity

Figure 12: Classification of time concepts

registration-based validity discussed in section 2. The definition of characteristic functions for that validity is perhaps the simplest. For any object e and any property P_i, $\mathcal{V}_{P_i}(e,t)$ will be 1 if t is equal to the time the recording is carried out, and will keep the value until recording of a different property P_j is performed. In a real situation, this characteristic function is easily implemented by automatically setting the start point of valid period of a data value to the system clock. Note that the registration-based validity is a special case of a single sort validity represented by a valid time.

From the foregoing arguments we can conclude that the valid time alone (we have chosen the terminology) is enough to represent the temporal validity of data objects unless the preservation of multiple past states is not a concern. If an application requires a multisort validity, that requirement can be easily fulfilled by providing multiple valid times in a temporal database system.

On the other hand, the physical time and the transaction time of [SA85] used in their temporal databases do not represent a sort of validity by themselves. They act as a reference time for the valid (or logical) time to measure the time difference we discussed in this section. Employing the two-dimensional time, consisting of valid and transaction times, we are able to preserve multiple past states produced by error corrections and delayed updates (not by retroactive updates). Nevertheless, it should be noted that only one sort of validity is represented by the two-dimensional time. For example, Figure 9 illustrates a sort of validity. In the figure, (d) depicts that validity most precisely, while (b) and (c) show stale ones. As a consequence, supporting multisort validity and preserving all database states by multidimensional time are two independent things. Accordingly, it is possible that we have an one-dimensional time and a two-dimensional time to support two

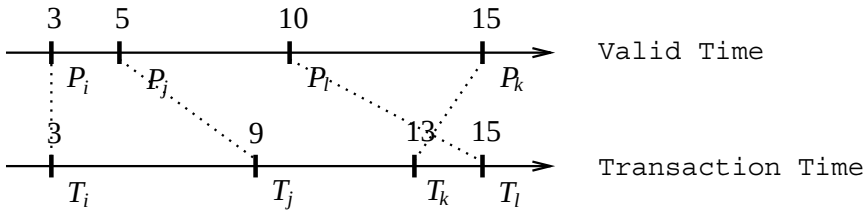


Figure 13: Combination of valid and transaction times

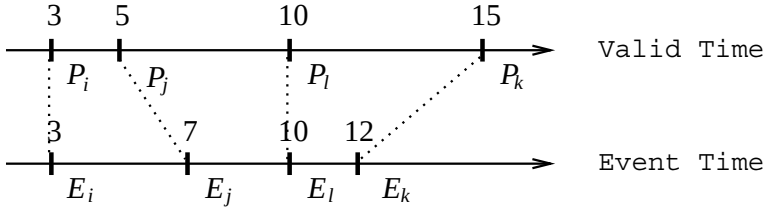


Figure 14: Combination of valid and event times

sorts of validity, if necessary.

Using the time notions introduced in this paper, multidimensional time may be configured in three ways:

1. Combination of valid and transaction times.
2. Combination of valid and event times.
3. Combination of valid, event, and transaction times.

Using (1), we can preserve multiple histories generated by error corrections and delayed updates. But we lose incomplete pasts generated by retroactive updates. Of course, the database does not keep any information concerning when events happened, much less the types of events. Figure 13 shows the roles of valid time and transaction time, which is a simplified view of Figure 11. As we can see, no information about events exists in Figure 13. However, from the transaction time it is possible to infer that transaction T_l was an error correction.

Using (2), we can preserve multiple histories generated by retroactive updates. But erroneous pasts generated by error corrections and delayed updates cannot be preserved. Figure 14 illustrates the situation simplified from Figure 11 by employing only the valid time and the event time. In the figure we don't have any information regarding transactions. As a result, it is not possible to infer that recoding of E_l was done by an error correction T_l.

Lastly, using all the three times (case (3)), all multiple pasts generated by those operations can

be preserved. This approach will give us the most general interpretation of temporal databases. However, the generality needs to be weighed against the increased complexity in interpretation from the users' viewpoint. While the implementation of three-dimensional time is possible, we do not believe that four- (or higher) dimensional time is manageable and that benefits from the generality of it would offset the far increasing intricacy of interpretation. From this practical standpoint, we claim that the valid, event, and transaction times are a maximal set of times needed and can be implemented in temporal databases.

5 Conclusions

In this paper, we have presented the notions of temporal validity and sort of validity. With these notions, we have been able to dispel the confusion among various time concepts. In addition, we have shown that retroactive and proactive updates cannot be modeled by the valid and transaction times. In order to resolve this problem, we have proposed the event time. Also, we have clarified the combinations of multidimensional times needed for preserving multiple pasts generated by retroactive updates, error corrections, and delayed updates. Handling multidimensional times in temporal databases requires a systematic interpretation methodology. We have extended this work to two-dimensional representations of time which correspond to the combinations of valid and transaction times and valid and event times [Kim92]. The notion of valid period is extended to a valid pattern in the two-dimensional times. Currently, we are investigating the three-dimensional representation using all the three time concepts.

References

[Ahn86] I. Ahn. Towards An Implementation of Database Management Systems with Temporal Support. In *Proceedings International Conference on Data Engineering*, pages 374–381, Los Angeles, 1986.

[Ari86] G. Ariav. A Temporally Oriented Data Model. *ACM Transactions on Database Systems*, 11(4):499–527, 1986.

[CM84] G. Copeland and D. Maier. Making Smalltalk a Database System. In *Proceedings International Conference on Management of Data*, pages 316–325, Boston, MA, 1984.

[CT85] J. Clifford and A. Tansel. On An Algebra for Historical Relational Databases: Two Views. In *Proceedings International Conference on Management of Data*, pages 247–265, Austin, Texas, 1985.

[CW83] J. Clifford and D. Warren. Formal Semantics for Time in Databases. *ACM Transactions on Database Systems*, 8(2):214–254, 1983.

[EJK92] R. Elmasri, M. Jaseemuddin, and V. Kouramajian. Partitioning of Time Index for Optical Disks. In *Proceedings International Conference on Data Engineering*, pages 574–583, Tempe, Arizona, 1992.

[EW90] R. Elmasri and G. Wuu. A Temporal Model and Query Language for ER Databases. In *Proceedings International Conference on Data Engineering*, pages 76–83, Los Angeles, 1990.

[EWK90] R. Elmasri, G. Wuu, and Y.-J. Kim. The Time Index: An Access Structure for Temporal Data. In *Proceedings International Conference on Very Large Data Bases*, pages 1–12, Brisbane, Australia, 1990.

[Gad88] S. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, 1988.

[GM91] D. Gabbay and P. McBrien. Temporal Logic & Historical Databases. In *Proceedings International Conference on Very Large Data Bases*, pages 423–430, Barcelona, Spain, 1991.

[GY88] S. Gadia and C. Yeung. A Generalized Model for A Relational Temporal Database. In *Proceedings International Conference on Management of Data*, pages 251–259, Chicago, 1988.

[Kim92] Seung-Kyum Kim. A Formal Appproach to Two-Dimensional Temporal Databases. Master's thesis, Department of Computer & Information Sciences, University of Florida, Gainesville, FL 32611, December 1992.

[Klo83] M. Klopprogge. Term: An Approach to Include the Time Dimension in the Entity-Relationship Model. In P. P. Chen, editor, *Entity-Relationship Approach to Information Modeling and Analysis*, pages 473–508. Elsevier Science Publishers, Amsterdam, 1983.

[LDE+84] V. Lum, P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner, and J. Woodfill. Designing DBMS Support for the Temporal Dimension. In *Proceedings International Conference on Management of Data*, pages 115–130, Boston, MA, 1984.

[NA89] S. B. Navathe and R. Ahmed. A Temporal Relational Model and a Query Language. *Information Sciences*, 49:147–175, 1989.

[SA85] R. Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In *Proceedings International Conference on Management of Data*, pages 236–246, Austin, Texas, 1985.

[Sar90] N. Sarda. Extensions to SQL for Historical Databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):220–230, June 1990.

[Ser80] A. Sernadas. Temporal Aspects of Logical Procedure Definitions. *Information Systems*, 5:167–187, 1980.

[SK86] A. Shoshani and K. Kawagoe. Temporal Data Management. In *Proceedings International Conference on Very Large Data Bases*, pages 79–88, Kyoto, Japan, 1986.

[Sno87] R. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[Soo91] M. D. Soo. Bibliography on Temporal Databases. *ACM SIGMOD Record*, 20(1):14–23, March 1991.

[SS91] L. Schäfers and G. Schlageter. Toward Full Support of Modeling and Querying Temporal Aspects in Relational Database Systems. Technical report, University of Hagen, West Germany, 1991.

[TC90] A. Tuzhilin and J. Clifford. A Temporal Relational Algebra as a Basis for Temporal Relational Completeness. In *Proceedings International Conference on Very Large Data Bases*, pages 13–23, Brisbane, Australia, 1990.

[TL91] C. Theodoulidis and P. Loucopoulos. The Time Dimension in Conceptual Modelling. *Information Systems*, 16(3):273–300, 1991.