University of Florida

Computer and Information Sciences

Resolution of Time Concepts in Temporal Databases

Sharma Chakravarthy
Seung-Kyum Kim

email: sharma@snapper.cis.ufl.edu

Tech. Report UF-CIS-TR-93-004

January 1993

(Submitted for publication)

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

Contents

1 Introduction **2**

2 Analysis of previous work **3**

3 Temporal validity **5**

3.1 Equi-distant discrete time 5

3.2 A simple data model 6

3.3 Definition of temporal validity 8

3.4 Sort of validity 10

4 Preservation of multiple past states **12**

4.1 Event time 13

4.1.1 Time difference between an event and its accompanying facts 13

4.1.2 Classification of events 13

4.1.3 Multiple pasts generated by the time difference 15

4.2 Transaction time 16

4.2.1 Error corrections 16

4.2.2 Delayed updates 16

4.2.3 Consolidation of error correction and delayed update 18

4.3 Orthogonality of event time and transaction time 19

4.4 Classification and maximal set of time concepts 21

5 One-dimensional interpretation **23**

5.1 History of an object 23

5.1.1 Special time points in history 24

5.1.2 Lifespan 25

5.1.3 Snapshot 26

5.2 Effect of data manipulation on special time points 26

5.3 Construction of history 28

6 Language and Implementation issues **29**

6.1 Need for a restriction on history 29

6.2 Attribute time-stamped relations 29

6.3 Temporal query languages 31

7 Conclusions **32**

Resolution of Time Concepts in Temporal Databases*

Sharma Chakravarthy **Seung-Kyum Kim**

Database Systems Research and Development Center
Department of Computer and Information Sciences
University of Florida, Gainesville, FL 32611
e-mail: {sharma, skk}@cis.ufl.edu

January 14, 1993

Abstract

In this paper, we resolve the confusion among different time concepts proposed for temporal databases and demonstrate the adequacy of three well-defined time concepts required for preserving multiple past states of a database.

We start with a formal definition of temporal validity which we believe is the first and the most comprehensive quantification of the notion of temporal validity. The notion of temporal validity is extended to sort of validity with which the confusion among various time concepts introduced earlier for temporal databases can be dispelled. Then, 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 the need for multidimensional representation. 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 only the valid and transaction times as thought earlier. We also show the adequacy of three time concepts (*event time, along with valid and transaction time*) for completely preserving different past states generated by retroactive and proactive updates, error corrections, and delayed updates. In addition, we define the evolution of an object (i.e., object's *history*) along one-dimensional time (valid time), by using the concepts of temporal validity and sort of validity. We conclude by sketching the implementation of history for the relational data model.

Index Terms: Temporal databases, valid, event, and transaction times, multiple past states, retro- and pro-active updates, error correction, delayed update, history of an object.

*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).

1 Introduction

The temporal database which incorporates time concept into the conventional database has been investigated by many researchers over the past decade [27]. 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 in this area includes [4, 2, 25, 12, 19, 20, 21]. Besides the emphasis on the relational model, temporal extensions to the ER and semantic models have begun to emerge in the literature [17, 9, 21, 29]. The formalization of temporal data model [22, 5, 30, 11] and the physical data organization for temporal databases [18, 23, 1, 10, 8] 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 [26] proposed three time concepts, valid time, transaction time, and user-defined time, based on the classification of 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 the first problem. 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. Using the temporal validity and sort of validity, we define the notion of *history*, being a representation of evolution of an object along one-dimensional time.

As another aspect of time concept, we show that valid and transaction times are not sufficient to model retroactive and proactive updates (the second problem), 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 the valid time is needed for a single sort of validity (one interpretation) and that 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, the problem of preserving multiple past states is discussed. In section 5 and 6, we show one-dimensional interpretation of temporal databases based upon the notion of history. Finally, in section 7, we make concluding remarks about our work.

2 Analysis of previous work

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

Copeland and Maier [7] 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 professor property 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 [18] 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 [7], 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 [18] is in essence the same with the effective validity of [7], provided that the event time and the logical time are for modeling of the same real-world validity.

The difference is that in [18] 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 [26]. 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 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 [7] 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 [18] in the sense that both times are used to model retroactive and proactive updates.

Yet another time notion, the *user-defined time* [26, 15] seems to be a misnomer. Its definition is that it is an attribute whose domain consists of date and time values [15] which is supplied by the user. Unlike previous time concepts, the definition of user-defined time designates it as data (attribute value), which is not defined in terms of time-stamps. If the time is treated as a data value, then there are at least two possible semantics: *interval* time such as 1 hour and *chronological* time (calendar time) such as Jan. 1, 1992. The definition of user-defined time does not indicate which time it refers to. Indeed, the interval time can be treated as an ordinary data type, and can be time-stamped with the chronological time in order to represent its validity. In this paper, the interval time is excluded from the discussion; all time concepts we deal with are chronological times.

[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 database.

Finally, another confusion arises in the work of Gadia and Yeung [13], in which the valid and transaction times of [26] were adopted for the demonstration of their n-dimensional, symmetric time concept. However, the valid and transaction times of [13] were in fact not those of [26], but the event and transaction times of [7], 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 vs. database-world) [26]. That dichotomy, together with the absence of a proper quantification of temporal validity, presumably misled us to the point that there is one genuine real-world validity and other possible database-world validity, 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 [18, 26], 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 discussed in the previous section can be clearly understood.

3.1 Equi-distant discrete time

Throughout this paper we assume an equi-distant discrete time domain T, termed the integer time domain, having "0" as the origin of time, which is used to represent the temporal validity. The integer time domain T is isomorphic to the Gregorian calendar time domain. Therefore, without the loss of generality, we use the integer time and the calendar time interchangeably in this paper.

***Now* and ∞** We adopt two *indefinite* time constants. One is *Now*, by which we mean a moving time point that exactly keeps track of the *current time*. We use the convention that all *Now*'s in a transaction are substituted with the same current time. This seems to be reasonable unless the transaction is a long-lived one.

The other indefinite time constant is ∞ (infinity). The indefinite constant ∞ denotes a sufficiently remote future time point.

Time interval A time interval is defined to be consecutive time points in T. To be more specific, a time interval, ι is a non-empty subset of the time domain T, i.e., $\iota \subseteq T$ and $\iota \neq \emptyset$, such that all time points in ι are consecutive. If ι consists of only one time point, it is regarded as being trivially consecutive.

Two time intervals, ι_i and ι_j, are said to be equal if any t in ι_i is also in ι_j and vice versa.

Also, ι_i and ι_j are said to *overlap* if there exists any time point t such that $t \in \iota_i$ and $t \in \iota_j$. A set of time intervals is said *disjoint* if and only if no interval overlaps one another.

A time interval is alternatively denoted by $(t_s \sim t_e)$, where t_s and t_e are the starting point and the ending point of the interval, respectively. For example, $(5 \sim 9)$ and $(1/1/88 \sim 8/31/91)$ are time intervals of the integer time system and the Gregorian calendar time system, respectively.

Indefinite time interval The indefinite time interval is defined to be a time interval whose ending point is the indefinite time constant ∞, denoted by $(t_s \sim \infty)$. The meaning of an indefinite time interval is that the ending point of an interval is not determined (or not known) as of now. Thus, the ending point, in fact, can be any time point greater than or equal to t_s. In other words, an indefinite time interval $(t_s \sim \infty)$ shall be equal to one of definite intervals, $\{t_s\}$, $\{t_s, t_s + 1\}$, $\{t_s, t_s + 1, t_s + 2\}$, and so on.

This interpretation, however, makes problems unnecessarily complicated even though it might lead to a more precise modeling. We, instead, just follow the mathematical meaning of ∞. That is, an indefinite time interval $(t_s \sim \infty)$ is simply interpreted as a set of time points, $\{t_s, t_s + 1, t_s + 2, \cdots, \infty\}$. Note that $(t_s \sim \infty)$ is more inclusive than another indefinite time interval $(t_s \sim Now)$.

3.2 A simple data model

Before presenting a formal definition of temporal validity, we define a simple data model from which some necessary concepts are drawn. The data model presented here is not intended to be complete. Its sole objective is to provide a set-theoretic definition of data value.

Object An object is an entity, a relationship among entities, or an attribute of an entity (or a relationship).

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 building, 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 the empty set) of A, i.e., $P \subseteq A$.[2]
- An object e, $e \in A$, has a property P if $e \in P$.

We also define the *empty property*, denoted by ϵ, to be non-existent property. In other words, if an object has the empty property ϵ, the object is regarded as having no property.

Incidentally, a property should not be confused with an attribute. An attribute in our work, is simply viewed as a subordinate object which along with other attributes (or subordinate objects) describes the semantics of an object. Defining various relations among objects, such as coordinate relations (entity vs. entity) and subordinate relations (entity vs. attributes), is beyond the scope and objectives of this paper. For details on data/knowledge modeling refer to [6, 3, 24, 14, 28].

Property set A *property set*, $\mathcal{P}$ is a meaningful set of properties that are 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. When every property in a property set is guaranteed to have exactly one member, the property set is called an *individual* property set. The individual property set is a special case of the disjoint property set. With the individual property set, each individual object in the object set can be identified.

The empty property ϵ is included in every property set and does not overlap any other property in the 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.

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

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

$$\text{Instructor} = \emptyset,\ \text{Assistant} = \{e_1\},\ \text{Associate} = \{e_2, e_3\},\ \text{Full} = \{e_4\},\ \text{Tenured} = \{e_3, e_4\}.$$
$$\text{Jane} = \{e_1\},\ \text{Smith} = \{e_2\},\ \text{John} = \{e_3\},\ \text{Miller} = \{e_4\}.$$

[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 (individual) property set, whereas $\mathcal{P}_{F_Rank}$ is an overlapping one. Note that all property sets include the empty property ϵ tacitly. The object e_7 might have empty properties included in property sets, $\mathcal{P}_{F_Name}$ and $\mathcal{P}_{F_Rank}$, implying that the object is no longer a faculty member.

As a matter fact, a property set corresponds to a domain of attribute in terms of the traditional definition.

3.3 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 professor property for Smith's rank is a state of whether or not Smith possesses the associate professor property with time.

Temporal validity The *temporal validity* of property P of 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 exists 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 customizing the definition of temporal validity. Note that the temporal validity of a property P of an object e is derived from the interpretation of the characteristic function. Later we will show how this concept of characteristic functions can be used to define validity for various time concepts. 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 professor 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.

In addition, the domain of valid periods (including recurring ones) can be denoted by 2^T, the power set of the time domain T.

With the valid period, the temporal validity can be extended as follows.

$$\mathcal{V}'_P : (A \times 2^T) \rightarrow \{0,1\}$$

given by

$$\mathcal{V}'_P(e,\tau) = \begin{cases} 1 & \text{if } \tau = \{t \mid \mathcal{V}_P(e,t) = 1\} \\ 0 & \text{otherwise} \end{cases}$$

where $e \in A$, a valid period $\tau \in 2^T$, and 2^T is the domain of valid periods.

We call functions such as $\mathcal{V}'_P$ *extended characteristic functions*. Note that existence and definition of an extended characteristic function, $\mathcal{V}'_P$ depends on $\mathcal{V}_P$.

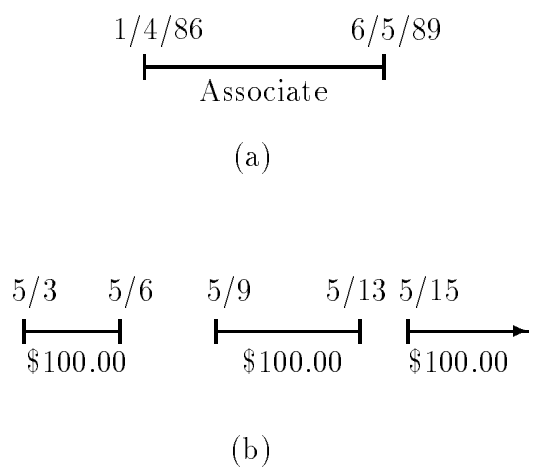

Figure 1: Valid periods

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 professor*.

$\underline{\mathcal{V}_1 P_1(e,t)}$:

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.

$\underline{\mathcal{V}_2 P_1(e,t)}$:

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 have different valid periods of Smith's associate professor property as shown in Figure 2.

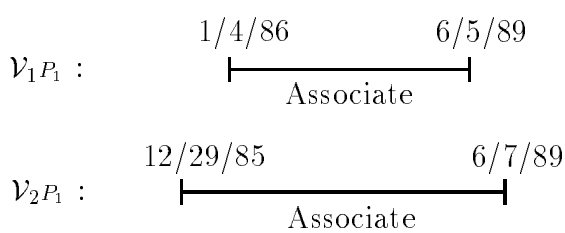

Figure 2: Two different valid periods of associate-professor property

3.4 Sort of validity

While the temporal validity deals with each individual property, the sort of validity concerns the totality of 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 $\mathcal{V}_u P_w \in \mathcal{S}_{\mathcal{P}}$, $\mathcal{V}_u P_w \neq \mathcal{V}_l P_m$, $\mathcal{V}_u P_w(e,t) = 0$.

According to the above definition, for a set of characteristic functions to be a legitimate sort of validity, the following conditions need to be satisfied.

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

The first condition 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.

The second condition 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 validity 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, $\mathcal{V}_1 P_1$, $\mathcal{V}_2 P_1$, $\mathcal{V}_1 P_2$, and $\mathcal{V}_2 P_2$ 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.

$$\mathcal{S}_{1\mathcal{P}} = \{\mathcal{V}_1 P_1, \mathcal{V}_1 P_2\}$$
$$\mathcal{S}_{2\mathcal{P}} = \{\mathcal{V}_2 P_1, \mathcal{V}_2 P_2\}$$

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* validity. For example, if we define $\mathcal{V}_1 P_2$ and $\mathcal{V}_2 P_2$ in a similar manner to $\mathcal{V}_1 P_1$ and $\mathcal{V}_2 P_1$, respectively, where P_2 is *full professor*, 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.

[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.

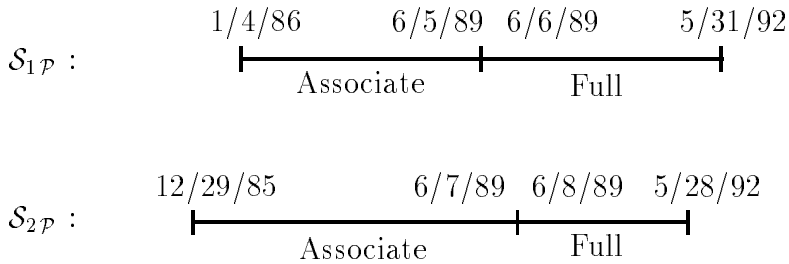

Figure 3: Two sorts of validity

On the other hand, 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.

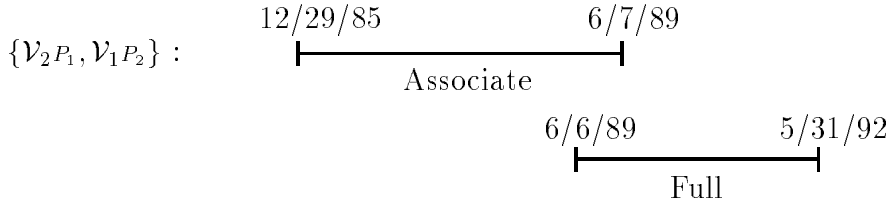

Figure 4: A disallowed combination of characteristic functions

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 one object set, in general, does not apply to other object sets. For example, the sort of validity $\mathcal{S}_{1\mathcal{P}}$ is not likely to be 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.

[25] is regarded as the first comprehensive treatment of the preservation of past states, although the aspect of retroactive update was mentioned earlier in [18]. 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 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

[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.

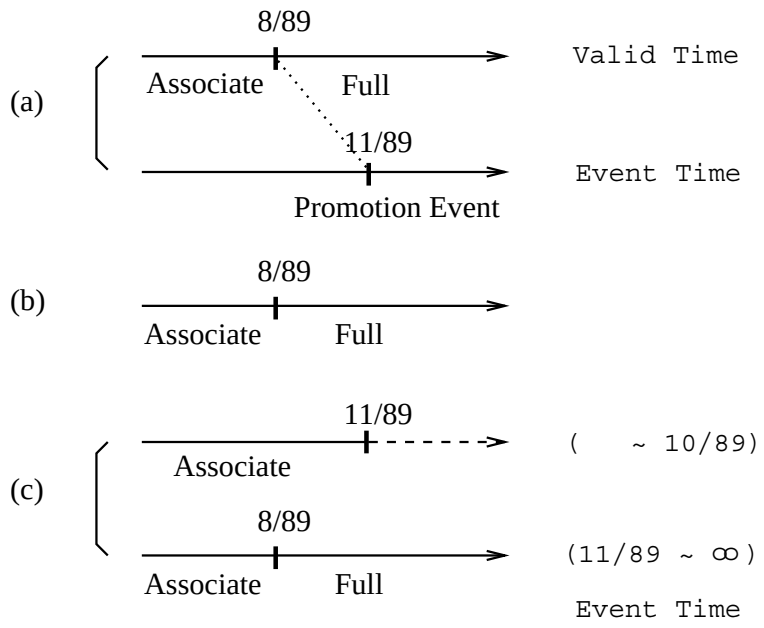

Figure 5: Retroactive event

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 has now happened), the grant usually will not be effective until a later time. The time difference

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

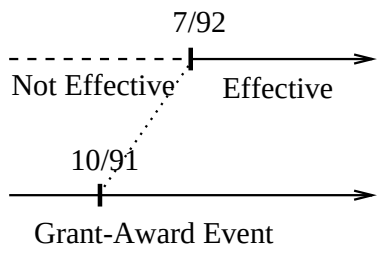

Figure 6: Proactive event

for the proactive event is shown in Figure 6. Other examples for the proactive event may be drawn from a temporal database for weather forecasting. 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 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.

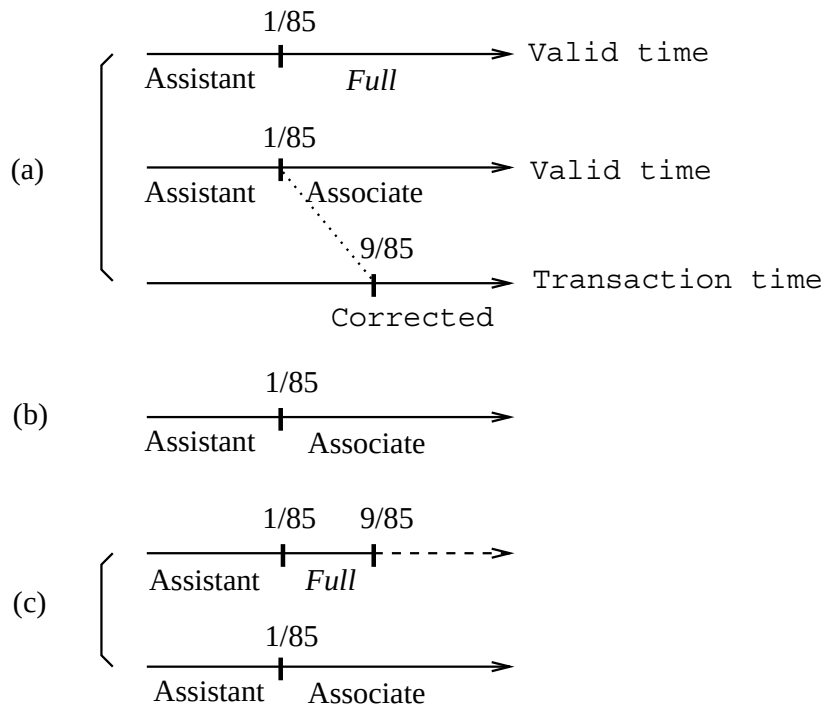

Figure 7: Error correction

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 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

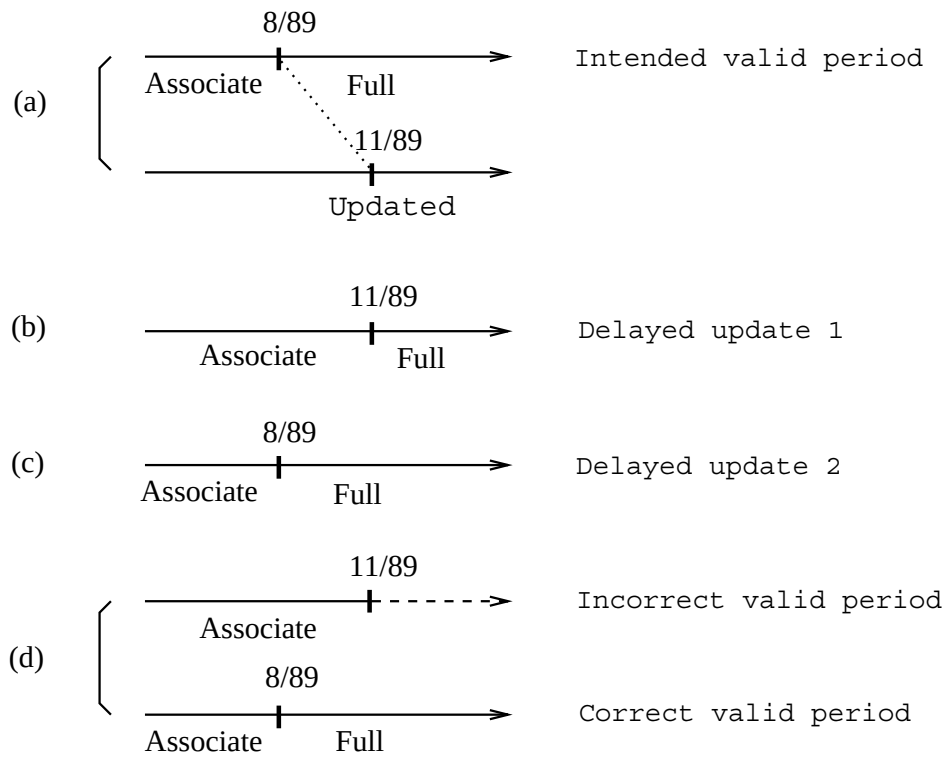

Figure 8: Delayed update

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 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)

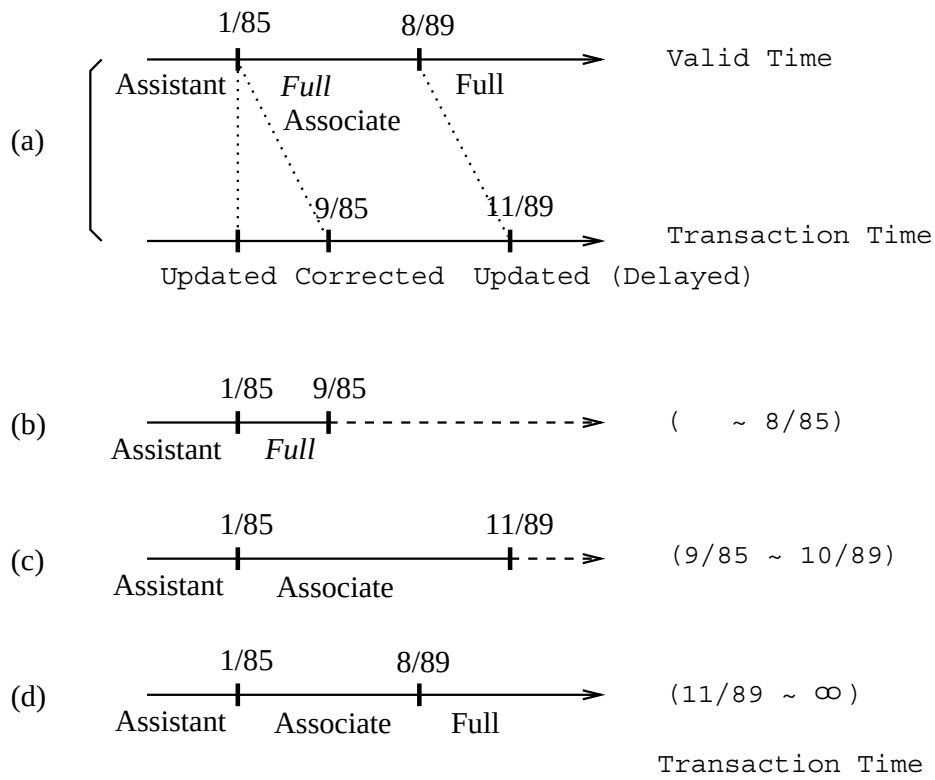

Figure 9: Consolidation of error correction and delayed update

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 [26, 25] 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

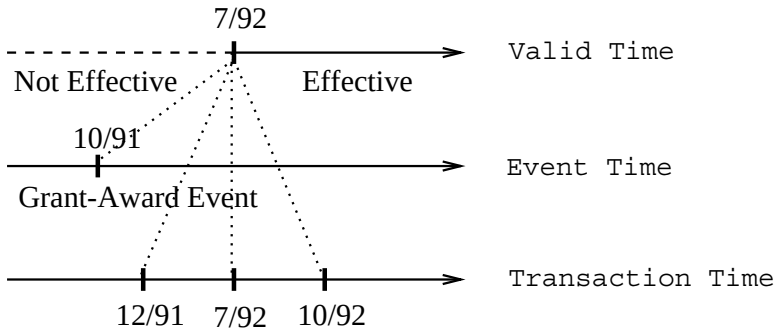

Figure 10: Misinterpretation of a proactive event

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 from 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 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]

[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-discovery. Such an event, bearing a different semantics, should have occurred somewhere between time points, 10 and 15.

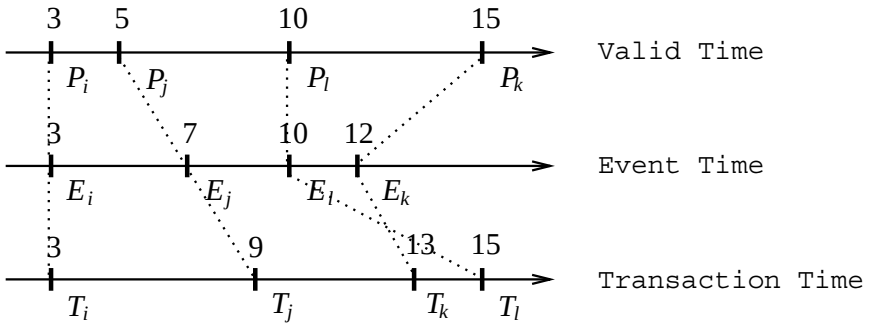

Figure 11: A general situation of event-occurring and its recording

Reference	Time concept	Representing sort of validity
[7]	Event	any sort of validity
	Transaction	registration-based validity
[18]	Logical	any sort of validity
	Physical	none (reference time for logical time)
[26]	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

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 [7], 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.

Here, the meaning of user-defined time in question needs to be bound to a chronological time, which is used as time-stamps to represent the temporal validity of certain data objects. Using time-stamps as data (attribute) values may be unavoidable due to the limitations of underlying temporal database models. In the temporal relational database, for example, birth-date of a faculty member need be included as an attribute of a relation TEACHERS, because the birth-date is not directly related to the validity of a faculty member but to that of a person, which is in a more generic object set (i.e., entity superclass) of TEACHERS. It should be noted that, however, the user-defined time has the same nature with the valid time in terms of representing validity.

The transaction time of [7] and that of [26] used in rollback databases represent the 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 this terminology) is enough to represent the temporal validity of data objects unless the preservation of multiple past states is 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 [26] 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 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

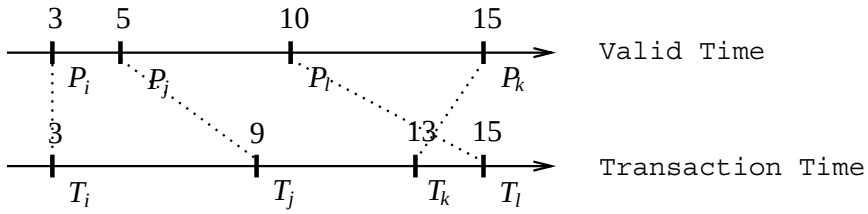

Figure 13: Combination of valid and transaction times

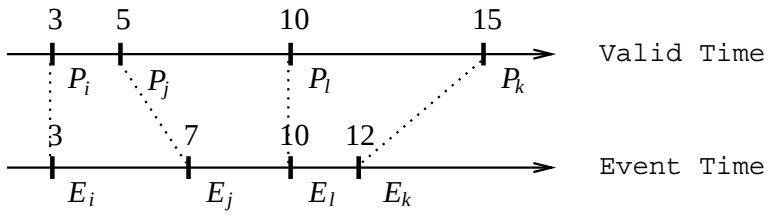

Figure 14: Combination of valid and event times

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 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 its generality 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, which are needed and can be implemented in temporal databases.

5 One-dimensional interpretation

The significance of temporal and sort of validity defined earlier is that they lead to a formal representation of the evolution of data values, which we call the *history* of an object. Before we can define validity and preservation of multiple pasts using two or three time concepts (leading to two- or three-dimensional representations, respectively), we need to establish the history of an object in terms of its evolution. This section formally defines the history of an object and relevant concepts, and shows how the history is constructed from data manipulation operations.

5.1 History of an object

A *temporal data value* possessed by an object is represented by a pair, (P, τ), where P is a property and τ is a valid period of P for the object. Usually, an object may have several properties during its

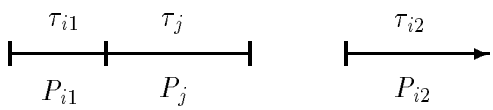

Figure 15: History of an object

lifetime. From the definition of sort of validity, it is guaranteed that valid period of each property does not overlap one another. Then, temporal data values which an object has had during its lifetime can be grouped together. We call such a group the *history* of an object. The history of an object reflects its evolution along the time dimension. A formal definition of the history is given below.

Definition: Given an object set A and its property set $\mathcal{P}$ and given a sort of validity $\mathcal{S}_{\mathcal{P}}$, a *history* H of object e in A is a set of temporal data values,

$$H = \{(P_i, \tau_i) \mid \mathcal{V}'_{jP_i}(e, \tau_i) = 1,\ \mathcal{V}_{jP_i} \in \mathcal{S}_{\mathcal{P}}\}$$

where property $P_i \in \mathcal{P}$, τ_i is the valid period of P_i, and $\mathcal{V}'_{jP_i}$ is the extended characteristic function of $\mathcal{V}_{jP_i}$.

A history can be depicted as shown in Figure 15. In the figure, the valid period of the property P_i (indicated in the figure as P_{i1} and P_{i2}) is the union of τ_{i1} and τ_{i2}.

5.1.1 Special time points in history

Suppose that property P_i of object e has valid period τ_i which is a set of disjoint time intervals, $\{\{t_i, t_i+1, \cdots, t_j-1\}, \{t_k, t_k+1, \cdots, t_l-1\}, \cdots, \{t_m, t_m+1, \cdots, t_n-1\}\}$.

Acquisition and release points Now we are interested in two kinds of special time points over the valid period τ_i: the initial points $t_i, t_k, \cdots, t_m$ of the time intervals, in which an object (say e) acquires a property (say P_i), and the time points $t_j, t_l, \cdots, t_n$, immediately after the final points of the time intervals, where the object e loses that property. We call these *acquisition* and *release* points, respectively. The valid period τ_i of the property P_i can also be denoted as $\{(t_i \sim t_j - 1), (t_k \sim t_l - 1), \cdots, (t_m \sim t_n - 1)\}$.

A valid period may contain an indefinite time interval, being the latest one in the valid period. An indefinite time interval, as a part of a valid period, is produced if there is no matching release point to an acquisition point. When a valid period contains an indefinite time interval, the valid period is called *indefinite valid period*.

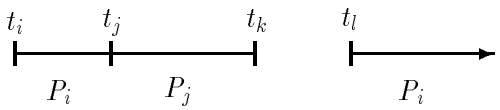

Figure 16: Special time points in history

Figure 16 illustrates acquisition and release points. Property P_i has two acquisition points t_i and t_l and a release point t_j, while property P_j has an acquisition point t_j and a release point t_k. The valid period of property P_i is an indefinite one.

Birth and death points Among the release points of a history, the one which is not an acquisition point of another property is termed a *death point*. In Figure 16, the release point t_k is a death point. Similarly, among the acquisition points, the first one in a history or the first one after a death point is termed a *birth point*. In Figure 16, the acquisition points t_i and t_l are birth points.

Note that a history may have multiple birth and death points. If the history for an object has more than one birth point (which implies at least one death point), as the one shown in Figure 16, we say that the object has been *reincarnated*.

5.1.2 Lifespan

A *lifespan* of an object is defined to be a time interval between its birth point and its closest death point. The death point itself is not included in the lifespan. To be more specific, let's assume that t_b is a birth point and t_d is its closest death point. Then, the relevant lifespan is a time interval, $\{t_b, t_b + 1, \cdots, t_d - 1\}$. If there is no matching death point for a birth point, then the relevant lifespan becomes *indefinite*. An indefinite lifespan is denoted by $(t_b \sim \infty)$[9], in the same manner an indefinite time interval or valid period is denoted. In Figure 16, the relevant object has two lifespans. One is $(t_i \sim t_k - 1)$, and the other is an indefinite lifespan, $(t_l \sim \infty)$. The reincarnation of an object can be stated in another way: if an object has multiple lifespans, the object is said to be reincarnated.

For the sake of simplicity, we assume that if an object has subordinate objects, all objects including the main object have the same lifespan. This assumption seems to be reasonable as insert and delete operations determining lifespan of an object (by specifying the birth and death points, respectively) usually take effect on an object as a whole[10].

[9] The denotation might have to be $(t_b \sim \infty - 1)$. However, it does not make any practical difference, although the two are not strictly equal.

[10] Sometimes we may confront with situations in which this assumption is too restrictive. For instance, as part of the schema evolution, some attributes may be dropped from (or added to) an existing relation. Then, those affected attributes may have to be treated as having different lifespan.

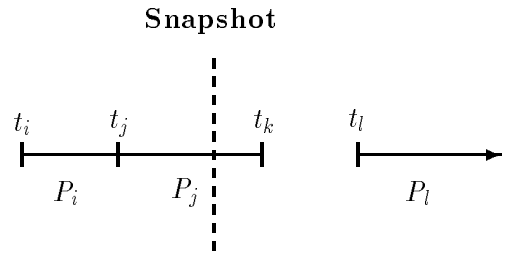

Figure 17: Snapshot

5.1.3 Snapshot

Since valid periods of properties in a history are disjoint, it is obvious that given a time point at most one property becomes valid. By moving the time point along the time axis we can get properties of an object at various moments. We, informally (somewhat imprecisely), call the property (data value) obtained at a time point a *snapshot*. Figure 17 depicts a snapshot. In the figure, the snapshot at the given time point is property P_j. A formal definition of snapshot is given below.

Definition: Given a property set $\mathcal{P}$ and a history H, the *snapshot* is a function,

$$sn : T \rightarrow \mathcal{P}$$

given by

$$sn(t) = \begin{cases} P_i & \text{if } t \in \tau_i \text{ and } (P_i, \tau_i) \in H \\ \epsilon & \text{otherwise} \end{cases}$$

where t is a time point in the (valid) time domain T, P_i is a property in $\mathcal{P}$ and ϵ is the empty property.

Analogous to the valid period, the snapshot provides us with information directly extractable from a history. While the valid period gives information regarding time period(s) an object possessed a given data value, the snapshot renders information about what data value an object possessed at a given time point.

5.2 Effect of data manipulation on special time points

In addition to the fact that the special time points are delimiters of a valid period, practically they are usually the only time points preserved in the database with respect to the validity of data values. And only data manipulation operations play the role of affecting the temporal information stored in a database. The information conveyed by data manipulation operations is defined as follows (object identification is assumed and is not explicitly shown).

$$
\begin{aligned}
I &: P\ (t_i, \{t_j\}) \\
U &: P\ (t_i, \{t_j\}) \\
D &: \epsilon\ (t_i) \\
C &: \{P\}\ (t_i, \{t_j\})
\end{aligned}
$$

where I, U, D, and C denote insert, update, delete, and error correction operations, respectively, and P is a property (data value), and t_i and t_j are acquisition and release points, respectively. In the definition, items in braces ($\{\}$) are optional.

For insert operation, t_i is the acquisition point of P and the birth point of the relevant object. Usually only t_i will be specified considering that in most situations the release point of a property is not known at the time of insertion. In that case, t_j is assumed to be ∞, resulting in an indefinite valid period.

For update operation, t_i is the acquisition point of P. In addition, t_i also should be the release point of the previous property because at most one property can be valid at a given time. t_j will not be specified unless release point of P is also known. If t_j is omitted, it is assumed to be ∞.

For delete operation, the empty property (ϵ) is recorded. That is, the relevant object will lose its property and cease to exist.[11] The acquisition point of the empty property, t_i, effectively becomes the release point of the previous property and eventually a death point of the relevant object. Note that only t_i can be given in delete operation. The release point, say t_j, of empty property can be given only by insert operation (i.e., reincarnation).

The error correction operation is quite general that it operationally encompasses all the other operations. If a corrective property P is given, it is operationally similar to insert or update operation. If P is not given, it is operationally a delete operation. However, the error correction operation needs to be semantically differentiated from the rest of the operations considering user's intention for its use. That is, insert, update, and delete operations contribute to the normal evolution of data objects, whereas correction operation is used to rectify errors occurred on the record of an evolution.

A correction operation can be easily identified by the acquisition point t_i which will be less than or equal to the last special point in a history. For example, suppose that P_k is the current (or the last) property in a history and t_k is the acquisition point of the property, being the final special point in the history. If an operation adds a property P_i with acquisition point t_i being less than t_k, then the operation is actually an error correction operation.

[11] In section 3 we assumed that if an object has no property the object does not exist.

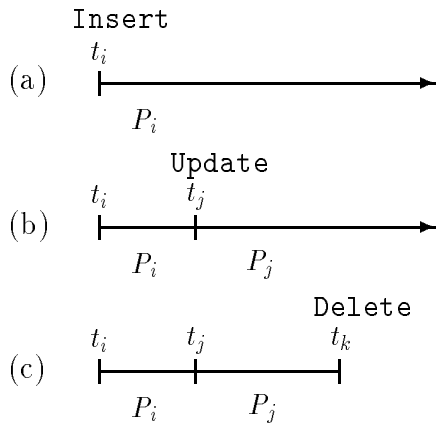

Figure 18: A developing history

5.3 Construction of history

Typically, the history of an object is created by an insert operation, developed by update operations, and eventually terminated by a delete operation. Once terminated, an object can be reincarnated by a new insert operation (thereby creating another lifespan). Often, the data manipulation operations contribute only one special point and that too aperiodically. Thus, we need a rationale to make up a complete valid period (regardless of whether it is a definite or an indefinite one) using the values given as part of the data manipulation operations. The following assumption, called *isochronal recording*, is used for that purpose.

Assumption 1 (Isochronal recording) *If and only if a property of an object changes, the change is recorded in the database.*[12]

The above assumption implies that if a change is not recorded (by users), the current property of the object is assumed to be unchanged and will continue to hold. In other words, if an object has property P_i at t_i, then the object is regarded as still retaining property P_i at $t_i + 1$, unless a new property P_j is recorded at $t_i + 1$. Based on the assumption, the validity of a property at intermediate points between an acquisition point and a release point can be derived; that is, the property of the acquisition point is retained up to its release point.[13]

Figure 18 shows a history based on the following data manipulation operations:

[12]Strictly, a change cannot be recorded simultaneously, causing temporal discrepancies between the modeled world and the recorded world. We will discuss this issue in the next section.

[13]This behavior of a property corresponds to that of "step-wise constant" time sequences described in [23]. In fact, there are properties that do not display this behavior. For example, a property describing an act, such as "shipping a product", is valid for a short period of time, not until the next act. Nonetheless, the isochronal assumption still works. That is, if a property changes soon after its occurrence, such a change also need be recorded simultaneously according to the assumption.

$I \ : \ P_i \ (t_i)$
$U \ : \ P_j \ (t_j)$
$D \ : \ \epsilon \ (t_k).$

By the insert operation, the relevant object acquires a data value P_i at t_i, and the resultant history is shown in Figure 18(a). By the update operation, the object gets P_j at t_j, and a new history is shown in Figure 18(b). Lastly, the object loses its data value at t_k by the delete operation, and Figure 18(c) shows the resultant history. Although we have multiple histories for an object as shown in Figure 18, only the last history is kept in the temporal database with one-dimensional time.

6 Language and Implementation issues

In this section, we sketch the language and implementation issues for the formalism proposed in this paper. We believe that our formalism provides a sound basis for various temporal database concepts without complicating the implementation. We use the relational data model for concreteness as well as to compare our results with previous work [4, 25, 19, 20].

6.1 Need for a restriction on history

We impose a restriction on the history to ensure that *the valid period of a property consists of a single time interval.* This restriction precludes the recurring valid periods, such as shown in Figure 1(b), to be a part of a history, and simplifies the implementation of temporal validity. If we allow the valid period to be a set of time intervals, then whenever an object gets a new temporal data value (that is, the object is updated), the existing history might have to be searched to see if the same property already exists. If so, a new time interval would be added to the existing valid period rather than create a new pair of property and valid period for the new temporal data value. A more serious problem than the search overhead, perhaps, will be that one cannot anticipate storage (memory/disk) space needed for time-stamps. This will result in waste of space as well as performance degradation. A ramification of this restriction is that whenever a data value is added by an update or an error correction operation, the data value is regarded as a new value regardless of the previous ones. In Figure 15, for example, even though P_{i1} and P_{i2} are the same properties, they are treated as different ones having their own valid periods of single time interval, τ_{i1} and τ_{i2}.

6.2 Attribute time-stamped relations

Although both attribute and tuple time-stampings have been proposed in the literature along with their advantages and disadvantages, we adopt the attribute time-stamping in our work. Our view of an attribute is that of an object that can evolve independently. A major critique against

TEACHERS

Name	Address	Rank
John	Gainesville	Assistant
Smith	Gainesville	Full
Jane	Alachua	Associate

Figure 19: A non-temporal relation: TEACHERS

TEACHERS

Name	*Valid time*	Address	*Valid time*	Rank	*Valid time*
John	$(8/88 \sim \infty)$	Gainesville	$(8/88 \sim \infty)$	Instructor Assistant	$(8/88 \sim 12/90)$ $(1/91 \sim \infty)$
Smith	$(1/81 \sim \infty)$	Gainesville Jacksonville Gainesville	$(1/81 \sim 10/85)$ $(11/85 \sim 7/86)$ $(8/86 \sim \infty)$	Assistant Associate Full	$(1/81 \sim 12/84)$ $(1/85 \sim 7/89)$ $(8/89 \sim \infty)$
Jane Jane	$(8/85 \sim 4/89)$ $(8/91 \sim \infty)$	Ocala Alachua	$(8/85 \sim 4/89)$ $(8/91 \sim \infty)$	Assistant Associate	$(8/85 \sim 4/89)$ $(8/91 \sim \infty)$

Figure 20: A temporal relation: TEACHERS

the attribute time-stamping has been its generation of non-1NF (non-first-normal-form) relations. However, unless one is building an experimental temporal database system using an existing non-temporal relational one, the tuple time-stamping has little advantage over the attribute time-stamping but causes enormous space waste and fragmentation of information pertinent to an entity. Moreover, regarding attribute time-stamped relations as non-1NF is inappropriate in that multiple data values which an attribute has during its evolution are different than a set value of non-temporal databases. In the former case, only one data value may be valid at a given time point, whereas the whole set is currently valid in the latter case. In this respect, an attribute time-stamped relation may be thought of as 1NF since multiple data values for an attribute in such a relation are still atomic.

In order to implement the attribute time-stamping, a pair of time-stamps are associated with each attribute value, representing a valid period of the value. A set of data values and their time-stamps belonging to an attribute of tuple represents a history of the attribute.

Figure 19 shows a traditional relation TEACHERS whose key attribute is "Name". The relation is extended to a temporal one by time-stamping each attribute as shown in Figure 20. Note that the key attribute value does not change while other non-key attribute values (may) change.

As a tuple is inserted, all its attributes get time-stamped by setting the birth point to a specified value and the death point to ∞. In Figure 20, Smith's rank has been evolved by the following four

operations:

I : Assistant (1/81)
U : Full* (1/85)
C : Associate (1/85)
U : Full (8/89).

The insert operation set the acquisition point of "Assistant" to 1/81. The rank was wrongly updated to "Full*" by the first update operation. The update set the release point of Assistant to 1/85 and the acquisition point of Full* to 1/85 too. The error was found later and the rank was corrected to "Associate". By the correction, the effect of the first update was removed from the relation and the value Associate got the acquisition point, 1/85. The rank was updated to "Full" by the second update operation. By the update, the release point of Associate and the acquisition point of Full were both set to 8/89; thus, the valid period of Associate became $(1/85 \sim 7/89)$.

Similarly, Jane's rank has evolved through the following operations:

I : Assistant (8/85)
D : ϵ (5/89)
I : Associate (8/91).

The operations suggest that Jane left the university in 5/89 and rejoined in 8/91.

Note that insert and delete operations take effect on a whole tuple rather than an attribute, thereby all attributes in a tuple have the same lifespan. All attributes of tuple Jane, for example, have lifespan, $(8/85 \sim 4/89)$ and $(8/91 \sim \infty)$. In other words, the object representing Jane was reincarnated.

6.3 Temporal query languages

Devising query languages over temporal databases has been a dominant part of the temporal database research. We already presented two rudimentary constructs of temporal query languages, the valid period and snapshot. Similar language constructs to the snapshot or more sophisticated forms are found in all work dealing with the temporal query language. They are usually termed "when" or "time-slice" clause. The following temporal query is borrowed from TQuel of [25].

```
range of F is TEACHERS
retrieve  F.Name
    where  F.Rank = "Associate"
    when  F.Rank overlap 1/89
```

The answer to the query is "Smith", since the when clause specifies that only attributes whose valid periods contain the time point, 1/89 be selected.[14]

The counterpart to the valid period as a language construct is also found in some researchers' work, but in more diverse and primitive forms. The syntax of the following temporal query is from TSQL of [19].

```
select  Rank.time-start, Rank.time-end
    from  TEACHERS
    where  Name = "Jane" and Rank = "Associate"
    time-slice  Now
```

The suffixes, "time-start" and "time-end" in select-clause return a valid period of an attribute value, and the "time-slice" clause is similar to the when clause. Accordingly, the answer should be (8/91, ∞).[15] However, a language construct implementing the valid period needs to be more sophisticated, because although we restricted a valid period to be a single time interval, in reality a data value may have a valid period composed of several time intervals. Smith's address, Gainesville, for example, has a valid period composed of two time intervals. For such a case, the constructs time-start and time-end do not work properly.

7 Conclusions

In this paper, we have formally defined the notions of temporal and sort of validity. Use of a characteristic function provides an interpretation based mechanism for defining the notion of validity. This is analogous to the interpretation of clauses in first-order logic which is essentially based on the domain as well as the interpretation assigned to constants and functions.

With the notions defined in this paper, we have been able to dispel the confusion among various time concepts. Also we have shown that retroactive and proactive updates cannot be modeled using only valid and transaction times. In order to resolve this problem, we have proposed the event time. Furthermore, we have clarified the combinations of multidimensional times needed for preserving multiple pasts generated by retroactive updates, error corrections, and delayed updates. In addition, the notion of history has been defined formally and its implementation using the relational data model has been sketched.

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 as well as valid and event times [16]. In

[14] The original TQuel is based on the tuple time-stamping. Thus, the query shown above and its interpretation do not precisely coincide with those of TQuel.

[15] Again, TSQL is based on the tuple time-stamping. The original TSQL will not allow such a query on attribute time-stamped tables.

the two-dimensional cases, the notion of valid period is extended to valid pattern and the notion of history to multi-history. Currently, we are investigating the three-dimensional representation using all the three time concepts.

References

[1] 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.

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

[3] P. P. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[4] 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.

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

[6] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

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

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] M. Hammer and D. McLeod. Database Description with SDM: A Semantic Database Model. *ACM Transactions on Database Systems*, 6(3):351–386, 1981.

[15] C. S. Jensen, J. Clifford, S. K. Gadia, A. Segev, and R. T. Snodgrass. A Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 21(3):35–43, Sep. 1992.

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

[17] 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.

[18] 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.

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

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

[21] 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.

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

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

[24] J. M. Smith and D. Smith. Database Abstractions: Aggregation and Generalization. *ACM Transactions on Database Systems*, 2(2):105–133, 1977.

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

[26] 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.

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

[28] S. Y. W. Su, V. Krishnamurthy, and H. Lam. An Object-Oriented Semantic Association Model (OSAM*). In *Artificial Intelligence: Manufacturing Theory and Practice*. Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, GA, 1989.

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

[30] 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.