ON THE SEMANTICS OF TRANSACTION TIME AND VALID TIME IN BITEMPORAL DATABASES

by

James Clifford
Information Systems Department
Stern School of Business
New York University

and

Tomas Isakowitz
Information Systems Department
Stern School of Business
New York University

December 23, 1992

Center for Research on Information Systems
Information Systems Department
Stern School of Business
New York University

Working Paper Series

STERN IS-92-42

Center for Digital Economy Research
Stern School of Business
Working Paper IS-92-42

On The Semantics of Transaction Time and Valid Time in Bitemporal Databases

James Clifford and Tomas Isakowitz
Information Systems Department
Stern School of Business
New York University

December 23, 1992

Abstract

Numerous proposals for extending the relational data model to incorporate the temporal dimension of data have appeared in the past several years. While most of these have been *historical databases*, incorporating in some fashion a *valid time* dimension to the data model and the query languages, others have been *rollback databases*, incorporating a *transaction time* dimension, or *bitemporal databases*, incorporating both of these temporal dimensions. In this paper we address an issue that has been lacking in many of these papers, namely, a formal specification of the precise semantics of these temporal dimensions of data. We introduce the notion of *reference time* – the time that any operation is applied to the database state - and provide a logical analysis of the interrelationships among these three temporal dimensions. We also provide an analysis of the meaning of various variables such as *now* and ∞ which have been used in many of these models without a complete specification of their semantics.

Contents

List of Figures

1 Introduction

In this paper we explore the issues involved in providing a coherent semantics for the temporal dimension of bitemporal databases. A bitemporal database is defined in [JCG+92] as a database with exactly one system supported valid time and exactly one system-supported transaction time. What are these two temporal dimensions to data? [JCG+92] defines the *valid time* of a fact as the time when the fact is true in the modeled reality, and the *transaction time* of a database fact as the time when the fact is stored in the database. In this paper we analyze the interrelationships among these two dimensions and a third dimension which we call the *reference time*. The reference time provides a means to refer to the state of the database as it would appear to an observer looking at the database at that time.

There have been numerous temporal database models proposed in the literature. Most of these have incorporated one or the other of these two temporal dimensions, but not both. The majority of these models have incorporated only the valid time dimension. These so-called *historical* data models include [JM80, BZ82, CW83, Ari86, Tan86, CC87, Lor87, NA89, Sno87, Gad88, Sar90], among others. A far fewer number, the so-called *rollback* data models, have incorporated only the transaction time dimension, such as [JMR89, LS92]). A slightly larger number than this, the *bitemporal* data models, have incorporated both dimensions, such as [BZ82, Sno87, MS91, Gad92].

In examining all of these temporal database proposals, we discern two patterns to the level at which they have been presented. Some of the models have been presented on what we will call the *logical level*, while others have been presented on what we call the *operational level*. On the logical level, the models have essentially been presented in such a way that the *model* (in the logical sense of the term) was clear. These papers essentially accord with the view expressed in [Rei84] that a relational database can be seen as a set of ground first-order formulae, for which there is a minimal model. However, many of the proposed models have not been specified in this tradition. Rather than being presented as a set of *ground atomic formulae*, these models have been presented a set of formulae some of which are ground, but others of which have included one or more *free variables*. Chief among these variables have

been "*now*" (used, for example in [CC87, Gad88, CT85]) and "∞" (used in [Sno87]), but "*uc*" (in [WJL93]) and "*forever*" have also been used or proposed. Nowhere have we found a clear exposition of these symbols, i.e., nowhere have the semantics of this type of database – a database with free variables – been formally specified so that the logical model that they represent was clear. Because the symbols chosen already suggest an *intuitive* meaning to many readers, their informal use has led many readers to suppose that they understood their semantics. However, this reliance on the reader's intuition has led to much confusion. The lack of a formal specification of the semantics of these databases with variables is the chief motivation for this paper.

Consider, for example. one of the most well-known bitemporal models in the literature, TQuel ([Sno87]). (Our focus on TQuel is not at all meant to disparage this important work, nor to imply that it is the only model to suffer from these problems. It is simply that, as the most completely specified bitemporal model in the literature, it has the distinction of being able to serve as the best example of the problems remaining to be solved.) An example relation in TQuel is shown in Figure 1. Note that it makes of the special variable symbol "∞" in both the transaction time and valid time dimensions. Unfortunately, the semantics of this symbol are not clearly defined. Instead, the operations of *append*, $modify$, and *delete* are defined as mappings from one database state with variables to another, and the semantics of the *retrieve* operation are defined. It is in this sense – that any semantics given for these databases with variables is given with respect to the operations of the model – that we call these operational level models.

In exploring the semantics of TQuel queries and operations as specified in the paper, we discovered several problems, chiefly related to the fact that the symbol "∞" is confusing and its semantics appears to be overloaded. First of all, the symbol "∞" does not mean what ∞ usually means. For example, in the tuple $< Jane, Full, 11\text{-}80, \infty, 10\text{-}80, \infty >$ the symbol "∞" in the $VALID\text{-}TIME(to)$ column does not mean that Jane will be a Full Professor throughout eternity, nor does the same symbol in the $TRANS\text{-}TIME(stop)$ column mean that the database should be seen as recording this fact for all eternity. Clearly something different from the intuitive meaning of *infinity* is intended by this symbol.

FACULTY					
		VALID-TIME		TRANS-TIME	
NAME	RANK	(from)	(to)	(start)	(stop)
Jane	*Assistant*	*9-71*	*12-76*	*9-71*	∞
Jane	*Associate*	*12-76*	*11-80*	*12-76*	∞
Jane	*Full*	*11-80*	∞	*10-80*	∞
Merrie	*Assistant*	*9-77*	*12-82*	*8-77*	∞
Merrie	*Associate*	*12-82*	∞	*12-82*	∞
Tom	*Associate*	*9-75*	∞	*8-75*	*10-75*
Tom	*Assistant*	*9-75*	*12-80*	*10-75*	∞
Tom	*Associate*	*12-80*	∞	*11-80*	∞

Figure 1: The Faculty Relation

Secondly, the symbol "∞" appears to have two different meanings, depending on whether it is used as a value for a transaction time or for a valid time. If indeed that is the case, it should really be two different variables, say "∞_{TT}" and "∞_{VT}". Finally, and most seriously, the precise interpretation of these symbols is not given.

Consider the pair of tuples for Merrie in Figure 1:

$< Merrie, Assistant, 9\text{-}77, 12\text{-}82, 8\text{-}77, \infty >$ and $< Merrie, Associate, 12\text{-}82, \infty, 12\text{-}82, \infty >$

It is not at all clear how these tuples came to be in the relation. It appears that the intent is to model the following sequence of events:

1. At time 8-77 (which we will refer to as the *reference time* of the operation) a transaction (which received the same time stamp) was executed which asserted that Merrie was an Assistant Professor, and would keep this rank *indefinitely.* Thus the actual tuple added at this time would have been: $< Merrie, Assistant, 9\text{-}77, \infty, 8\text{-}77, \infty >$

2. At reference time 12-82 a transaction (which received the same time stamp) was executed asserting that Merrie had the rank of Associate Professor beginning immediately and would keep this new rank indefinitely.

Figure 1, reproduced from [Sno87], incorrectly shows that this resulted in the following

sequence of changes to the relation:

1. The *Assistant* tuple for *Merrie* is shown as having its $VALID\text{-}TIME$ field updated from ∞ to 12-82, thus becoming: $< Merrie, Assistant, 9\text{-}77, 12\text{-}82, 8\text{-}77, \infty >$, and
2. The new tuple $< Merrie, Associate, 12\text{-}82, \infty, 12\text{-}82, \infty >$ was (correctly) added.

The database in Figure 1 reflects this incorrect treatment of the update of Merrie's rank. With this treatment, the database has *lost information*, namely that from 9-77 through 12-82 the database was predicting that Merrie's rank would be Assistant indefinitely into the future.

In fact, according to the description of the semantics of these symbols and the operational semantics provided for the replace and insert commands (independently confirmed ([Sno]), the following sequence of steps would actually have been performed:

1. The $TRANS\text{-}TIME(stop)$ field of Merrie's Assistant tuple would have been changed to 12-82, the tuple becoming: $< Merrie, Assistant, 9\text{-}77, \infty, 8\text{-}77, 12\text{-}82 >$. This tuple reflects that during the period 8-77 through 12-82 (exclusively), the database recorded that Merrie's rank from 9-77 onwards was Assistant.
2. A new tuple would have been added, $< Merrie, Assistant, 9\text{-}77, 12\text{-}82, 12\text{-}82, \infty >$, reflecting the fact that during the period 12-82 and onward the database recorded that Merrie's rank from 9-77 through 12-82 was Assistant.
3. Finally, a new tuple would have been added, $< Merrie, Associate, 12\text{-}82, \infty, 12\text{-}82, \infty >$, reflecting the fact that during the period 12-82 and onward the database recorded that Merrie's rank from 12-82 onward was Associate.

In addition to this problem with the operational semantics for updates, the semantics given for the TQuel *retrieve* statement specifies a treatment for "∞" which also appears to be incorrect. For example, consider a query asking what the database records about Tom's rank at 11-75. In TQuel this would be expressed as:

```
range of t is faculty
retrieve (t.rank)
where t.name = ''Tom''
as-of 11-75
```

The correct answer should obviously be derived from the third tuple for Tom: $< Tom$, *Associate*, 12-80, ∞, 11-80, $\infty >$, since, intuitively, the value of the `as-of` clause, 11-75, lies in the apparent range of the tuple's transaction time validity, namely [11-80, ∞). However, according to the semantics suggested for the symbol "∞" ([Sno87, p.272]), "∞ is replaced with a distinguished integer, say 0". In this case, however, the semantics of the `as-of` clause of this tuple reduces to the Boolean expression $Before(\text{11-75}, 0) \wedge Before(\text{10-75}, \text{11-75})$ which evaluates to $false$. In fact with this semantics no tuples will satisfy the query, since no time is before 0, and so the database will respond with the empty relation, indicating (incorrectly) that it knows nothing about Tom's rank as of 11-75. The other seemingly likely choice for "∞", namely "the largest possible" transaction time, will also not function properly, for in this case the database would appear to be storing information about, and hence predicting, *future* transactions that will occur, i.e., the database could be "rolled forward" to some future state. Clearly a precise semantics for this variable symbol, and the others in the literature, is needed.

We will have frequent occasion to refer to the example database in Figure 1. Since it is notationally simpler to work with integer times (1, 2, 3, etc.) instead of dates, we will use Table 1 to map between integers and dates throughout the rest of the paper. Figure 2 represents the same information as Figure 1, but with the dates replaced by an integer.

In the remainder of this paper we will address the issues that we have touched upon in his introduction. In Section 2 we introduce our notation and definitions for the concepts to follow. In Section 3 we introduce a two-dimensional, graphical notation for bitemporal "objects" which we have found useful in illustrating their semantics. Section 4 discusses the notion of a lifespan, contrasts the lifespan of an object in its valid- and transaction-time dimensions, and discusses the timeslice operation for the transaction- and valid-time

TQuel Dates	Integer Time
9-71	1
8-75	2
9-75	3
10-75	4
12-76	5
8-77	6
9-77	7
10-80	8
11-80	9
12-80	10
12-82	11
1-83	12

Table 1: Converting from TQuel Dates to Integer Times

FACULTY					
		VALID-TIME		TRANS-TIME	
NAME	RANK	(from)	(to)	(start)	(stop)
Jane	*Assistant*	*1*	*5*	*1*	∞
Jane	*Associate*	*5*	*9*	*5*	∞
Jane	*Full*	*9*	∞	*8*	∞
Merrie	*Assistant*	*7*	*11*	*6*	∞
Merrie	*Associate*	*11*	∞	*11*	∞
Tom	*Associate*	*3*	∞	*2*	*4*
Tom	*Assistant*	*3*	*10*	*4*	∞
Tom	*Associate*	*10*	∞	*9*	∞

Figure 2: The Faculty Relation Recast

dimensions. In Section 5 we illustrate various alternatives semantics that can be given to the variables in historical, rollback, and bitemporal databases. We conclude in Section 6 with a summary of this work and suggestions for future research.

2 Definitions and Notation

In order to discuss the issues involved in the various models of valid time and transaction time and the relationships between them, we need to have one, underlying notion of time to serve as our temporal universe. The granularity of the transaction time and the valid time may be different, but it would not make sense to have completely different models of time for these two dimensions, for example, to view one of these as discrete and the other as continuous. For in such a case it would not be possible to express queries about the interaction between these two temporal dimensions – for example, *Which faculty members received a retroactive change in rank?*, i.e., for whom the the valid time of their change in rank was earlier than the transaction time which recorded it. Such queries are clearly desirable in a bitemporal database. Since most database researchers have adopted the view that valid time in a database is best viewed as discrete, and every database transaction model that we are aware of has this property, we will adopt a discrete model of time here.

There is one additional time that needs to be introduced, a time we shall call the *reference time*, a term analogous to the *indices* or "points of reference" in intensional logic ([Mon73]), and discussed more recently in the context of historical databases in [Fin92]. Like all databases, a temporal database exists in time. When we want to refer to the state of the database at some particular time in its history, we call this time the *reference time*, and denote it by t_*. The intent is to be able to denote what an observer looking at the database at time t_* would see. For the same reason that motivates the comparability of transaction times with valid times, we will also want the set of reference times to be drawn from our same universe of times.

Therefore, let T be our *temporal universe* and let T satisfy the following properties:

1. T is countably infinite,

2. T is unbounded at either end, and

3. T is totally ordered by a relation, which we will symbolize by $<$.

4. T contains a distinguished element, $\top$, that satisfies the property that $\forall t \in T[t \neq T \rightarrow t < \top]$.

In other words, T is isomorphic to the set of integers $I = \{\ldots, -2, -1, 0, +1, +2, \ldots\} \cup \{\top\}$.

A bitemporal database needs domains for two temporal universes, the *valid time universe* and the *transaction time universe*, and it may be desirable or convenient to restrict them to some subset of T. Therefore, let $T_{VT} \subseteq T$ denote the *valid time universe* of our database, and $T_{TT} \subseteq T$ denote its *transaction time universe.* We will shortly discuss what other kinds of constraints might be needed or desirable on these two sets.

Whenever we discuss a particular database instance, two moments in time are of particular interest. The first of these is what we may call the *birth* of the database, i.e., that time in T_{TT} when the first transaction occurs for the database. We can, if we like, think of this as the transaction that inserts the first tuple. Let us denote this time by t_0. This time is important because it constrains the reference time, i.e., $t_0 \leq t_*$. To simplify the discussion that follows, we will assume that t_0 is a *single* value for all of the relations in any given database. The second moment in time of special interest is the moment at which we issue some operation or query on the database. We denote this time by t_{now}. Clearly, we have the following constraints on these three times: $t_0 \leq t_* \leq t_{now}$. Note that t_{now} is the time at which an operation or query is performed, whereas t_* is the time at which we would like to evaluate or "observe" the database.

For a particular database, with a birthdate of t_0, the following sets of times will prove to be interesting:

(i) $T = \{\ldots, t_0, \ldots\}$

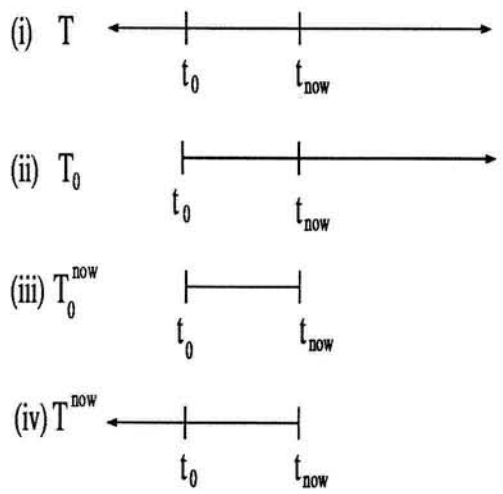

Figure 3: Temporal Domains of Interest

(ii) $T_0 = \{t_0, \ldots\}$

(iii) $T_0^{now} = \{t_0, \ldots, t_{now}\}$

(iv) $T^{now} = \{\ldots, t_0, \ldots, t_{now}\}$

The relationship between each of these times and the underlying universe of times T is shown in Figure 3.

Whenever we look at the database at a reference time t_*, we see the value of db_{t_*}, i.e., what is stored in the database at time t_*. In contrast, we will use the notation $[x]_{t_*}$ to denote the *logical interpretation* of some database object x at reference time t_*. Thus we will distinguish between r_{t_*} and $[r]_{t_*}$, $r.t_{t_*}$ and $[r.t]_{t_*}$, $r.t.A_{t_*}$ and $[r.t.A]_{t_*}$, and between db_{t_*} and $[db]_{t_*}$, for the contents or the logical interpretation, respectively, of a relation r, a tuple t in r, attribute A in t in r, or the entire database db, at a given reference time t_*. Indeed, if q is a query in some query language L for our data model, $q(db)_{t_*}$ and $[q(db)]_{t_*}$ denote the result of that query, at the operational level and its logical interpretation, respectively, when asked at reference time t_*.

Given these preliminary notions, our discussion of reference time, above, suggests a useful framework for viewing our database. A temporal database – and indeed each relation in it – can be seen as "indexed" by the reference time, i.e., the database can be viewed as a trajectory

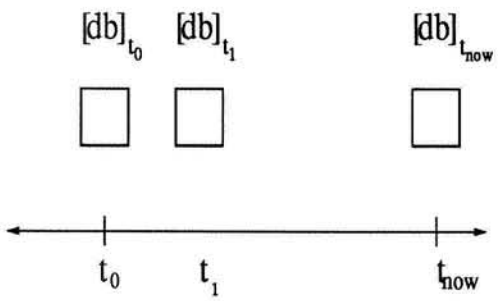

Figure 4: Database Evolution in Reference Time

of database states (or instances) through time: $[db]_{t_0} \rightarrow [db]_{t_0+1} \rightarrow \ldots \rightarrow [db]_{t_{now}}$ Moreover, each relation r can likewise be viewed as: $[r]_{t_0} \rightarrow [r]_{t_0+1} \rightarrow \ldots \rightarrow [r]_{t_{now}A}$ If the model is not completely extensional, instead of storing these relations it stores $r_{t_0} \rightarrow r_{t_0+1} \rightarrow \ldots \rightarrow r_{t_{now}A}$. Figure 4 depicts this view of the database trajectory.

Most of the time the database $[db_{i+1}]$ will be the same as the database $[db_i]$ (this is why we don't want to store the entire trajectory), except in the case where a transaction is COMMITted at time $i+1$. Note that it is possible (indeed, typical), to have $db_{i+1} = db_i$ while $[db]_{i+1} \neq [db]_i$.

2.1 Extensional Representation of Bitemporal Values

A simple and convenient extensional representation for bitemporal attribute values can be obtained by using sets of *ground* tuples of the form $<$ `tt,vt,a` $>$[1]. In order to do this, however, we need to understand the meaning of any variable symbols that may appear in our database. The set of triplets in Table 2 represents the value of the rank attribute for all of Tom's tuples in the faculty relation of Figure 2. We will call the set of all of these tuples for a given "object" in a relation (e.g., "Tom") an *attribute history*[2]. Although the representation does not make it explicit, the value `r` of `rank` is a function of vt and tt, i.e., for each pair of values $< vt, tt >$ there is exactly one entry `(vt,tt,r)` in the table. This

[1]Note that the the VALID-TIME start and stop appear here after the TRANS-TIME, whereas in Snodgrass' notation, they appear before the TRANS-TIME.

[2]We are oversimplifying a bit here, and ignoring the distinction made in [CCT93] between grouped and ungrouped temporal data models. For simplicity we are assuming here, as the TQuel paper does, that the key attribute **NAME** does not change over time, and therefore can serve to locate all of the tuples for a given "object" such as , here, a faculty member.

(2,3,Associate)
(2,4,Associate)
(2,5,Associate)
...
(3,3,Associate)
(3,4,Associate)
(3,5,Associate)
...
(4,3,Assistant)
(4,4,Assistant)
...
(4,9,Assistant)
...

Table 2: Some of the triplets in the extensional representation of Tom's rank.

representation provides a homogeneous treatment of the transaction time and valid time dimensions. In fact, it can be easily seen that swapping the order of these in the triplets produces an essentially equivalent representation.

Taken literally, the use of "∞" in the definition of the `faculty` relation results in an infinite set of triplets for `Tom`'s `rank` attribute. As is apparent form this discussion, the use of symbols such as "∞" complicates the ground model. It is not clear at all that the representation of Table 2 is correct, i.e., that it accurately represents the information implied in the database by the variable symbol "∞". Moreover, from an implementation point of view, the ground model can be very space inefficient.

We will call the extensional representation of the database its *logical view*, in contrast to the *operational view* given by the entries present in Figures 1 and 2. Thus, at the logical level, the database contains sets of ground tuples. This is consistent with current semantic models for databases [Rei84] and is useful in defining the behavior of the database. When we refer, for example, to $r.t.A_{t_*}$, we are referring to the value of attribute A in tuple t in the relation r as stored in the database at time t_*, i.e., at the operational level. By contrast, when we refer to $[r.t.A]_{t_*}$, we are referring to the the logical level of this attribute value, i.e. to this extensional representation. We point out that, in the literature, many models, such as the one in [CCT93], have been presented completely extensionally, i.e. on the logical

level, and have not made use of any variable symbols requiring further interpretation. In these models, $r.t.A_{t_\star} = [r.t.A]_{t_\star}$.

Although symbols such as ∞ have been widely used in temporal databases at the operational level, it is not clear how these are to be translated into ground tuples at the logical level. That is to say, in these models, $r.t.A_{t_\star}$ in general *cannot* equal $[r.t.A]_{t_\star}$, since the latter is a purely extensional set of *ground* tuples. Moreover, given $r.t.A_{t_\star}$, it is not entirely clear what $[r.t.A]_{t_\star}$ is. In this paper we explore these issues and propose well defined translation mappings from $r.t.A_{t_\star}$ to $[r.t.A]_{t_\star}$.

3 Bitemporal Graphs

In our investigations on bitemporal databases, we have found the graphical notation that we are about to explain quite useful for various reasons. Firstly, its visual appeal is intuitive, and this makes bitemporal concepts easier to grasp. Each time dimension corresponds to an axis: transaction time is the X-axis, valid time is the Y-axis. This allows us to represent passage of time as spatial displacement, and provides a visual representation for interesting phenomena such as history changes and predictions about the future. Secondly, the graphs provide a uniform representation of both time dimensions. This makes explicit the homogeneous nature of these entities and enables a uniform treatment of both time dimensions. Thirdly, the graphical representation makes explicit the differences in various treatments of time in temporal databases. As we will see in Section 5, the graphical representation clearly shows that the assumptions that values hold until *now*, or until *infinity* or *until-changed* are quite different. Lastly, although it may be hard to visualize graphs with more than two dimensions, the uniform treatment we present here can be nonetheless easily extended to more dimensions. Thus, our framework can be extended to encompass multi-dimensional temporal databases, for example indexical databases ([Cli92]).

3.1 Graphical Representation of Bitemporal Relations

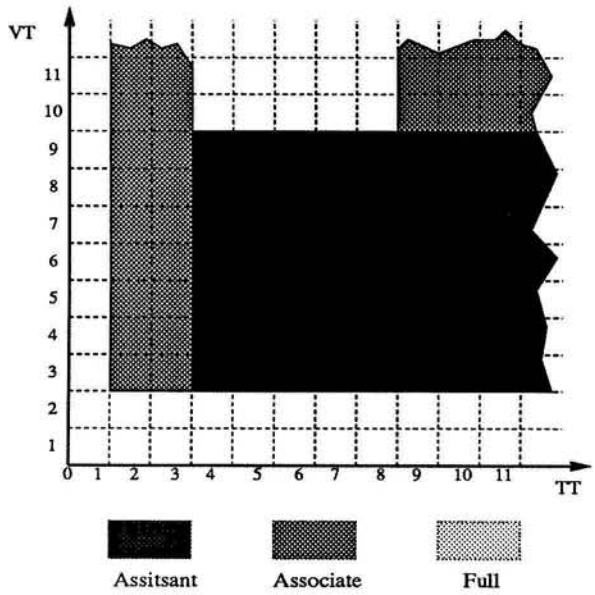

Figure 5: A graphical representation of the bitemporal **rank** attribute for `Tom` in the `Faculty` relation.

Figure 5 represents the *rank* attribute history of `Tom` in the `Faculty` relation. The ragged edges in a particular dimension indicate that the area extends in that dimension to *"infinity"*. In the graphical representation, each attribute history is assigned a graph. Thus the information about an object such as faculty member `Tom` is viewed with two graphs: one for the **Name** attribute and another for the **Rank**. Alternatively, different colors can be used for the various attributes, thereby plotting all attributes onto the same graph[3]. For the purposes of this paper, it will suffice to concentrate on one attribute at a time.

Each cell in the graph stands for a particular transaction time, tt, and a particular valid time, vt. As indicated in the "key" to Figure 5, the cell's hue of grey represents the value of the rank corresponding to the $< tt, vt >$ combination. For example, at $tt = 2, vt = 3$, Tom's rank is `Associate`; and for $tt = 8, vt = 9$ it is `Assistant`. In fact, although the graph is bi-dimensional, it represents tri-dimensional information since it plots points of the form $< tt, vt, rank >$, where the domain of $rank$ is $\{Assistant, Associate, Full\}$.

The vertical dimension, valid time, represents historical data, namely how the value of an attribute changed over time in the real world. A vertical column, for example, the column

[3]This would only be possible in a *homogeneous* model which assumes that all attributes in a tuple have the same lifespan.

for $tt = 11$, shows what the database recorded at that time about the attribute's history. In this case, at $tt = 11$, the database recorded that Tom was an Assistant Professor from time 3 through 9, and an Associate from time 10 onwards.

The horizontal dimension, transaction time, captures the changing database perceptions about a particular time in the real world. For example, a horizontal row for $vt = 3$ in the graph shows that for transaction times 2 through 3, the database recorded Tom's rank as `Associate`, but subsequently recorded his rank as `Assistant`, through *"infinity"*. The change in hue along the horizontal row for $vt = 3$ captures this change in perception.

The diagonal "line" in the graph corresponding to (tt, vt) where $tt = vt = x$ captures information about what the database records at time x about time x in the real world. (This is what is captured by the so called *snapshot* databases.) Any cells above the diagonal represent predictions, because the tt is earlier than the vt. Any cells below the diagonal record database perceptions about the past, because the vt is earlier than the tt.

Not all graphs for bitemporal relations need to correspond to contiguous rectangular shapes. Figure 6 presents the hypothetical case of a professor who was an Associate until time 4, then left the department, and came back later at time 9 with the rank of Full Professor. There is no information about her rank during the time she was not in the department.

4 Constraints and Operations on Transaction and Valid Times

In this section we will explore the issue of how a particular data model or application might want to constrain the set of allowable transaction or valid times in a database. We will discuss this issue in terms of the notion of a lifespan, and define precisely what a lifespan is in a bitemporal database. We will then use this notion to look at two particular operations of special interest in a bitemporal database, namely a valid-time timeslice and a transaction time timeslice, and we will relate the second of these to the notion of a rollback.

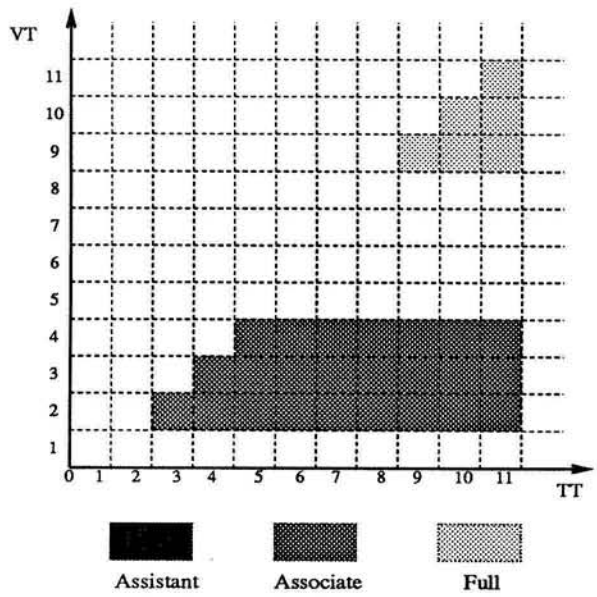

Figure 6: The discontinuity in available information is represented by discontiguous areas in the graph.

4.1 Lifespans

We have not spoken yet of how precisely we are to incorporate our two temporal dimensions into the relations of our model. We have done this because we wish to speak about bitemporal database models in general, and not about any particular model.

The idea of a *lifespan* is central in temporal databases. In [JCG+92] it is defined as follows:

> "The *lifespan* of a database object is the time over which it is defined. The valid-time lifespan of a database object refers to the time when the corresponding object exists in the modeled reality, whereas the transaction-time lifespan refers to the time when the database object is current in the database.
>
> If the object (attribute, tuple, relation) has an associated timestamp then the lifespan of that object is the value of the timestamp. If components of an object are timestamped, then the lifespan of the object is determined by the particular data model being employed."

Notice that the definition implies that in a bitemporal database, an object has a lifespan in each temporal dimension. In fact, if we consider the combination of these two dimensions, as in a bitemporal database, there is a third lifespan that we need to consider. Let us therefore make the following definitions:

- By l_{VT} shall be meant the lifespan of an object in its valid time dimension.
- By l_{TT} shall be meant the lifespan of an object in its transaction time dimension.
- By $l_{TT\text{-}VT}$ shall be meant the bitemporal lifespan of an object.

The question then arises as to what precisely are these lifespans, and secondly, is it reasonable to constrain them in any way, and for what purpose.

4.2 Valid Time Lifespan

The valid time lifespan of a database object is simply the set of all valid times that are associated with that object. Let us frame this discussion in terms of the value of an attribute A in a tuple t in a relation r, i.e., in terms of $r.t.A$.

Definition. The *valid time lifespan* of $r.t.A$ at reference time t_* is given by:

$$l_{VT}([r.t.A]_{t_*}) = \{vt \mid \exists a \; \exists tt \; [(tt, vt, a) \in [r.t.A]_{t_*}]\}$$

This definition is extensional. It says that the valid time lifespan of the tuple is simply the set of all valid times in its extensional representation. Let us now consider four possible restrictions that a database might choose to enforce on this lifespan:

1. VT_P: $\forall vt \; \forall t_*[vt \leq t_*]$, i.e., all valid times must be prior to the reference time. Such a database could only allow the recording of *past information.*

2. VT^F: $\forall vt\ \forall t_*\ [t_* \leq vt]$, i.e., all valid times must be later than the reference time. Such a database could only allow the recording of *future information.*

3. VT_P^F: no restrictions. Such a database could allow the recording of *past or future information.*

4. VT_ϕ : $\forall vt\ \forall t_*\ [t_* = vt]$, i.e., the valid time must always equal the reference time. Such a database could only allow the recording of *current information.*

4.3 Transaction Time Lifespan

Definition. The *transaction time lifespan* of $r.t.A$ at reference time t_* is given by:

$$l_{TT}([r.t.A]_{t_*}) = \{tt \mid \exists a\ \exists vt\ [(tt, vt, a) \in [r.t.A]_{t_*}]\}$$

Before considering four possible restrictions that a database might choose to enforce on this lifespan, we must remember that the intrinsic meaning of transaction time enforces a constraint of its own on the possible values of tt, above, viz.: $t_0 \leq tt \leq t_*$. In other words, at any given moment (t_*) the universe of transaction times for all objects in the database is restricted to just those moments from t_0 through t_*, inclusive. That is, it should not be possible either to *predict* future transactions or *predate* a transaction with an earlier timestamp. This is consistent with the discussion of transaction time in [JCG+92]:

> Transaction times are consistent with the serialization order of the transactions. Transaction time values cannot be after the current time. Also, as it is impossible to change the past, transaction times cannot be changed. Transaction times may be implemented using transaction commit times.

This constraint distinguishes our two universes, i.e., there is no *necessary* analog to this constraint for T_{VT}. The *transaction times* are system-generated, not user-generated, and they are monotonically increasing. In other words, the appropriate model for transaction

	TT_ϕ	TT_P
VT_ϕ	static	rollback
VT_P	(past) historical	(past) Bitemporal
VT^F	(future) historical	(future) Bitemporal
VT_P^F	(full) historical	(full) Bitemporal

Table 3: Classification of Databases By Temporal Dimensions Offered

times is that they are given out by a constantly ticking system clock, which always functions properly.

This intrinsic constraint eliminates the possibilities TT^F and TT_P^F. We are left with the following two:

1. TT_P: $\forall tt\ \forall t_*\ [t_0 \leq tt \leq t_*]$, i.e., all transaction times must be prior to the reference time. Such a database could only record *past transactions.*

2. TT_ϕ : $\forall vt\ \forall t_*\ [t_* = vt]$, i.e., the transaction time must always equal the reference time. Such a database could only record the *current transaction.*

These considerations lead to the classification in Table 3 of databases with respect to what constraints they impose with respect to the two temporal dimensions.

Finally, we can define the notion of a *bitemporal lifespan* of an object as the set of pairs $< tt, vt >$ associated with it in the database, as follows:

Definition. The *bitemporal lifespan* of $r.t.A$ at reference time t_* is given by:

$$l_{TT\text{-}VT}([r.t.A]_{t_*}) = \{< tt, vt > \ | \ \exists a\ [(tt, vt, a) \in [r.t.A]_{t_*}]\}$$

4.4 Timeslice Operators

A frequent operation in temporal databases is the *timeslice* operation, which is analogous to a *projection* in the temporal dimension. The valid-time timeslice, which we will denote

by Π^{VT}, projects the database onto some subset of its valid times. The transaction-time timeslice, denoted Π^{TT}, projects the database onto some subset of its transaction times.

Recall that at any time t_{now}, if we look at particular attribute history in the database, the information in it is given by $[r.t.A]_{t_{now}}$, consisting of a set of triples of the form (tt, vt, a).

Definition. The *valid time timeslice* of $r.t.A$ at valid time vt_i is given by:

$$\Pi^{VT}_{vt_i}([r.t.A]_{t_{now}}) = \{(tt, vt_i, a) \mid (tt, vt_i, a) \in [r.t.A]_{t_{now}}\}$$

Definition. The *valid time timeslice* of $r.t.A$ at a set of valid times $S = \{vt_{i_1}, \ldots, vt_{i_n}\}$ is given by:

$$\Pi^{VT}_{S}([r.t.A]_{t_{now}}) = \{(tt, vt, a) \mid (tt, vt, a) \in [r.t.A]_{t_{now}} \wedge vt \in S\}$$

Note that a frequent type of set S of interest is an interval, such as $[t_i, t_j)$.

Definition. The *transaction time timeslice* of $r.t.A$ at transaction time tt_i is given by:

$$\Pi^{TT}_{t_i}([r.t.A]_{t_{now}}) = \{(tt, vt_i, a) \mid (tt_i, vt, a) \in [r.t.A]_{t_{now}}\}$$

Definition. The *valid time timeslice* of $r.t.A$ at a set of transaction times $S = \{tt_{i_1}, \ldots, tt_{i_n}\}$ is given by:

$$\Pi^{TT}_{S}([r.t.A]_{t_{now}}) = \{(tt, vt, a) \mid (tt, vt, a) \in [r.t.A]_{t_{now}} \wedge tt \in S\}$$

The graphical notation for attribute histories provides a useful way to visualize these timeslice operators. A transaction-time timeslice at a time t_i would correspond to viewing *only* the column in the graph corresponding to $tt = t_1$. A transaction-time timeslice at

a set of times $S = \{tt_{i_1}, \ldots, tt_{i_n}\}$ would correspond to viewing only those columns in the graph corresponding to each tt_j in S. A transaction-time timeslice at a set of times $S = \{tt_{i_1}, \ldots, tt_{i_n}\}$ which corresponds to a contiguous interval $[tt_{i_1}, tt_{i_n}]$ would correspond to viewing only those columns corresponding to a rectangle with lower and upper bounds of tt_{i_1} and tt_{i_n}, respectively. The same visualization of valid-time timeslicing can be made, with *"horizontal rows"* replacing *"vertical columns"* in the description, above.

4.5 Rollbacks

When users are interested in what the database looked like at a particular reference time, an operation, typically called *rollback*, is invoked. Given a particular reference time rt_i, the user is interested in $[db]_{rt_i}$. The purpose of the rollback operation is to enable the user to ask queries about $[db]_{rt_i}$ using the current database $[db]_{t_{now}}$.

How does the rollback operation differ from our transaction timeslice operator? A request to rollback the database to time rt_i is a request to see the database as it was at a time rt_i. We can think of this rollback as a two-place operator defined as follows,

$$rollback([db]_{t_{now}}, rt_i) = [db]_{rt_i}.$$

Note that this is a semantic definition, i.e. *rollback* does not specify how $[db]_{rt_i}$ is to be obtained from $[db]_{t_{now}}$. It is our contention that in a sound bitemporal database model, transaction time timeslice should provide rollbacks. Hence, we expect the following equation to be satisfied:

$$rollback([db]_{t_{now}}, rt_i)) = \Pi_S^{TT}([db]_{t_{now}})$$

where $S = [t_0, rt_i]$.

Let us return to the discussion in Section 1 about the problems with the treatment of the *update* operation in TQuel. Consider the evolution of Tom's rank extracted from Figure 2 and shown in Figure 7. Two predictions and one correction can be observed in the figure. The first prediction occurs at *rt=2*, when a prediction is stored that his rank will be that

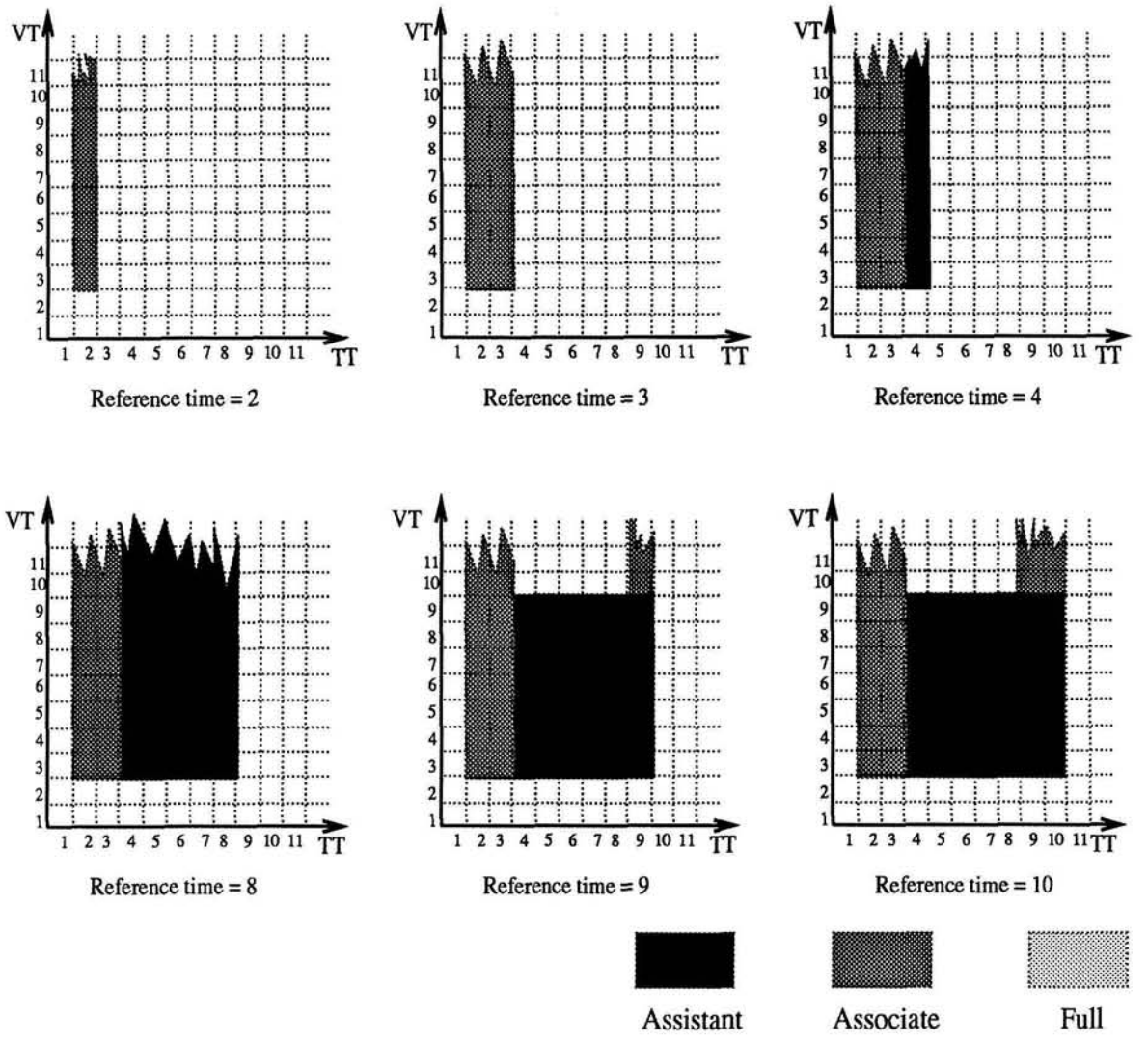

Figure 7: The evolution of Tom's rank.

of Associate from time *vt=3* onwards (until ∞). The graph for *rt=3* shows how temporal relations do change even when no transactions are committed. A whole new column for *tt=3* is now available as a result of time passage. A correction can be observed at *rt=4* in the discontinuity of the horizontal line for *vt=3* that is below the diagonal– normally values below the diagonal suffer no changes. The relation grows naturally until *rt=9* when a new prediction is recorded that Tom will be promoted to Associate at *vt=10.* The last picture– *rt=10* –shows once more, normal growth.

Let us consider Tom's rank in Figure 7, and assume that $t_{now} = 10$. The graph itself indicates that for this history $t_0 = 2$. Consider two rollback operations on a database with

this information, namely $rollback([db]_{10}, 3)$ and $rollback([db]_{10}, 8)$.

the first of these rollbacks would result in the graph for Reference time = 3. By our discussion above concerning the visualization of timeslice operators on these graphs, we see that this is exactly the same as that portion of the graph labeled Reference time = 10 consisting of the vertical columns 2 and 3. Thus we have

$$rollback([db]_{10}, 3) = \Pi^{TT}_{[2,3]}([db]_{10}).$$

A database that correctly handles transaction time should have this property. However, the second of these rollbacks would result in the graph for Reference time = 4. This is not the same as that portion of the graph labeled Reference time = 10 consisting of the vertical columns 2 through 8. Hence, in this case $rollback([db]_{10}, 4) \neq \Pi^{TT}_{[2,4]}([db]_{10})$. As we pointed out in Section 1, this is due to an error in the way updates were handled, resulting in the `Faculty` relation shown in Figure 1.

5 On the semantics of ∞, *now*, *forever* and *until changed*

For at least two reasons, temporal database models presented in the literature have tended to utilize *variables* such as ∞ in their representation schemes. In the valid time dimension, these variables have been used because the values of many attributes frequently do not change over long periods of time. In the transaction time dimension, they have been used because as the reference time moves inexorably forward – as the clock on the wall "ticks" – for most of these "ticks" the database does not change state, and so it is inefficient to store multiple copies of the same, unchanged information. While there have been various schemes proposed in the literature for this, it is unfortunate that the semantics of these schemes has seldom been clearly presented. It has therefore been unclear whether and how these schemes differ, or if in fact some of them are the same. Figure 8 presents a number of these schemes, namely: interval representations, *now*, ∞, and *uc*.

It is logic that can provide an answer to the question of what, precisely, these various notations mean. Specifically, we want to look at the *logical model* that is intended by each of

NAME	*RANK*	*VALID-TIME*	*TRANS-TIME*
Tom	Assistant	$[3, 10)$	$[4, 10)$

(a) Intervals

NAME	*RANK*	*VALID-TIME*	*TRANS-TIME*
Tom	Associate	$[3, \infty)$	$[2, 4)$
Tom	Assistant	$[3, 10)$	$[4, \infty)$
Tom	Associate	$[10, \infty)$	$[9, \infty)$

(b) Intervals and ∞

NAME	*RANK*	*VALID-TIME*	*TRANS-TIME*
Tom	Associate	$[3, now)$	$[2, 4)$
Tom	Assistant	$[3, 10)$	$[4, now)$
Tom	Associate	$[10, now)$	$[9, now)$

(c) Intervals and *now*

NAME	*RANK*	*VALID-TIME*	*TRANS-TIME*
Tom	Associate	$[3, uc)$	$[2, 4)$
Tom	Assistant	$[3, 10)$	$[4, uc)$
Tom	Associate	$[10, uc)$	$[9, uc)$

(d) Intervals and *uc*

Figure 8: Shorthand Representation Schemes

these representation shorthands, for the logical model is concerned with questions of truth and validity. In order to do this we need to understand the role of the variables in these representations.

A logical model specifies the value of every predicate, *for every moment* in the universe of times T. If a shorthand notation is used, with special symbols such as *now*, the model is *incompletely specified* unless the meaning of these symbols is given, i.e., unless it is possible to uniquely determine what *logical model* is being defined.

Given our universe of times T, linearly ordered and discrete, it is clear that the tuple in Figure 8 (a), which uses no variables, corresponds to the logical model given in Figure 9. The interpretation of the other tuples, which *do* utilize variable symbols, is more problematic. Earlier, in Figure 2, we showed what we assumed to be the logical model for Figure 8 (b). However, other interpretations of this symbol are possible. In the next section we will explore this topic in detail.

```
(4,3,Assistant)
(4,4,Assistant)
(5,3,Assistant)
(5,4,Assistant)
(5,5,Assistant)
(6,3,Assistant)
...
(9,9,Assistant)
```

Figure 9: Logical Model of Intervals

We perceive the special symbols "∞", "*now*", "*forever*" and *until changed* ("*uc*") to be notational aids at the operational level, that provide means of expressing an evolving database without explicitly enumerating all of the values concerned. Although these symbols have been extensively used in the literature on temporal databases, their exact meaning has not been made precise. In this section, a number of possible mappings are proposed and discussed to make the meaning of these symbols precise. These mappings translate the special symbols into the logical model whose extensional representation was introduced in Section 3.1.

We view entries at the database level of the form $< [t_1, t_2) \mapsto value >$ as shorthand for the set of tuples: $\{(t_1, a), (t_1 + 1, a), \ldots, (t_2 - 1, a)\}$. Hence $[t_1, t_2)$ stands for an open-closed interval (and $[t_1, t_2]$ for a closed-closed one). In addition, the special symbols ∞, *now*, *forever* and *uc* denote interval endpoints as follows.

- $[t_1, \infty)$: From time t_1 (including t_1) onwards.
- $[t_1, now]$: From time t_1 (including t_1) up until (and including) the current time.
- $[t_1, forever)$: From time t_1 (including t_1) onwards (this the same as ∞.)
- $[t_1, uc)$: From time t_1 (including t_1) onwards until there is a change.

It appears to us that ∞ and *forever*, used in different models, have the same meaning. Since, as was pointed out in Section 1, the symbol ∞ has been used in the literature with various meanings, we will use the symbol *forever* here. The semantics of *uc* will be discussed

at the end of this section. However, let us note at this point, that in the case where only one time dimension is considered, there does not appear to be a difference between *uc* and *forever*. This follows from the fact that until the symbol *uc* is changed, it does indeed have the meaning as same intention as *forever*. Once a transaction changes the symbol, it no longer appears in the database where it used to, and hence the issue of its semantics vanishes.

Based on this analysis, we conclude that there are only two symbols to consider in the case of rollback and historical databases: *forever* and *now*. In the case of bitemporal databases, as we shall see, there are additional consideration.

It is useful to think of these special symbols as variables whose values are to be determined by a *variable assignment* $g : Var \mapsto T$, where Var is the set of variables in our language and T is the universe of times. Thus, if $g(now) = 55$, then the interval $[t_1, now]$ is mapped by g to $[t_1, 55]$. As we shall see, certain restrictions apply to the functions g. Since these restrictions depend upon the reference time t_*, at which we wish to observe the database we will denote these variable assignments by g_{t_*}.

The underlying semantics of these variables might differ, for rollback and historical databases. In order to distinguish among them, we will use subscripts as follows: now_{tt} and $forever_{tt}$, for rollback databases, and now_{vt} and $forever_{vt}$, for historical databases.

5.1 Rollback Databases

When only the transaction time dimension of the data is incorporated into the database, the symbols now_{tt} and $forever_{tt}$ are used in a tuple to allow it to anticipate answers to queries to be made at a future time. As explained in Section 4, the semantics of transaction time do not allow future transaction times to be recorded in the database. This forces t_2 in **r1** of Table table:rollback to be no later than t_*, and also forces the meaning of *forever* to denote intervals that end at the current time.

Table 4 explains how the special symbols at the database level are to be interpreted at the

	Database View	Logical View
r1	$[t_1, t_2) \mapsto a$	$[A]_{t_*} = \{(tt, a) \mid t_1 \leq tt < t_2\}$
r2	$[t_1, now_{tt}] \mapsto a$	$[A]_{t_*} = \{(tt, a) \mid t_1 \leq tt \leq t_*\}$
r3	$[t_1, forever_{tt}) \mapsto a$	$[A]_{t_*} = \{(tt, a) \mid t_1 \leq tt \leq t_*\}$

Table 4: The Meaning of the Special Symbols in Rollback Databases.

logical level. In terms of variable assignments, we see that the assignment g_{t_*} is constrained so that:

$$\begin{aligned} g_{t_*}(now_{tt}) &= t_* \\ g_{t_*}(forever_{tt}) &= t_* \end{aligned}$$

Hence, $g_{t_*}(forever_{tt}) = g_{t_*}(forever_{tt})$. Therefore, *now* and *forever* have the same meaning in rollback databases.

Recall that even when the database itself does not change at time t_*, its logical interpretation might be different. As an example, consider the following entry for the value of a rank attribute history:

$$[5, now] \mapsto \texttt{Assistant}$$

The entry is added to the database at time $rt = 5$, and so the tt value of 5 is stored in the database. The logical view at $rt = 5$ is derived from $g_5(now_{tt}) = 5$, hence $[rank]_5 = \{(5, \texttt{Assistant})\}$. Assuming that no transaction is committed at time $tt = 6$, the logical view as observed at time $rt = 6$ will be derived from $g_6(now_{tt}) = 6$, yielding $[rank]_6 = \{(5, \texttt{Assistant}), (6, \texttt{Assistant})\}$. Hence, even though no transaction has taken effect between times 5 and 6, the logical views differ.

5.2 Historical Databases

When only the valid-time dimension of the data is incorporated into the database, the symbols *now* and *forever* are used in a tuple to allow it to express the values of attributes over some period of time, perhaps encompassing the past or even the future.

	Database View	Logical View
h1	$[t_1, t_2) \mapsto a$	$[A]_{t_*} = \{(vt, a) \mid t_1 \leq vt < t_2\}$
h2	$[t_1, now_{vt}] \mapsto a$	$[A]_{t_*} = \{(vt, a) \mid t_1 \leq vt \leq t_*\}$
h3	$[t_1, forever_{vt}) \mapsto a$	$[A]_{t_*} = \{(vt, a) \mid t_1 \leq vt\}$

Table 5: The Meaning of the Special Symbols in Historical Databases.

Table 5 provides the meaning for the special symbols, as sets of tuples. In contrast to rollback databases, future times are allowed because it is possible to make predictions in historical databases. Hence, *now* and *forever* have different meanings.

In terms of variable assignments, the interpretation of *forever*, is handled by the inclusion of $\top$ in the set T, such that $\top$ is greater than any other element of T. Thus, we have the following variable assignment:

$$\begin{aligned} g_{t_*}(now_{vt}) &= t_* \\ g_{t_*}(forever_{vt}) &= \top \end{aligned}$$

5.3 Bitemporal Databases

When valid time and transaction time are combined, there might be interactions between the two times. In Table 6 we provide denotations for the six combinations obtained from the analyses of the rollback and historical cases, namely one of $\{\mathbf{r1}, \mathbf{r2}\}$ with one of $\{\mathbf{h1}, \mathbf{h2}, \mathbf{h3}\}$. The table interprets the combination of a transaction time-interval and valid time-interval as a form of Cartesian product. Figure 10 provides a graphical view of the six cases. The arrows in Figures 10-b, 10-d, 10-e and 10-f stand for *growing now*, i.e. they move with t_*. The ragged edges in Figures 10-c and 10-f indicate that the areas continue to infinity, or the value $\top$.

In terms of variable assignments, the entries of Table 6 combine the assignments for the rollback and historical cases as follows,

$$g_{t_*}(now_{tt}) = t_*$$

	TT	VT	Logical View
r1× h1	$[tt_1, tt_2)$	$[vt_1, vt_2) \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt < tt_2 \wedge vt_1 \leq vt < vt_2\}$
r1× h2	$[tt_1, tt_2)$	$[vt_1, now_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt < tt_2 \wedge vt_1 \leq vt \leq t_*\}$
r1× h3	$[tt_1, tt_2)$	$[vt_1, forever_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt < tt_2 \wedge vt_1 \leq vt\}$
r2× h1	$[tt_1, now_{tt}]$	$[vt_1, vt_2) \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq t_* \wedge vt_1 \leq vt < vt_2\}$
r2× h2	$[tt_1, now_{tt}]$	$[vt_1, now_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq t_* \wedge vt_1 \leq vt \leq t_*\}$
r2× h3	$[tt_1, now_{tt}]$	$[vt_1, forever_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq t_* \wedge vt_1 \leq vt\}$

Table 6: The meaning of the special symbols in bitemporal databases.

$$g_{t_*}(now_{vt}) = t_*$$

$$g_{t_*}(forever_{vt}) = \top$$

5.4 Until Changed

A subtle point in the semantics of bitemporal intervals is that combinations of transaction-time intervals and a valid-time intervals are interpreted as Cartesian products. These result in rectangular areas as shown in Figure 10, or unions of rectangular areas. However, it is debatable whether this captures the meaning that the notation seems to convey. Consider for example, the entry for *Jane* inserted into the database at reference time 1, as:

$$< Jane \quad , [1, now_{tt})[1, forever_{vt}) \mapsto Assistant >$$

Such a tuple *predicts* that Jane's rank will be *Assistant* forever, i.e. at valid times $2, 3, 4, \ldots$. This can hardly be an accurate representation of the real world where, in fact, it is not known for sure how long she will remain in her position. Moreover, the tenure process dictates that, in any case, she will not be an Assistant beyond seven years of her hiring date. Thus the information conveyed by the database as we interpreted it is inaccurate! The intended meaning of the above entry is more likely to be: *"Jane will be an Assistant as long as she is not promoted or quits her job, i.e. until changed."* No prediction is made about her rank at times beyond reference time= 1. However, if at reference time=2, the entry still remains in the database, then it would state that at time valid time=2, Jane is still an Assistant.

How are we to represent *until changed*? Probably, the intended evolution of the informa-

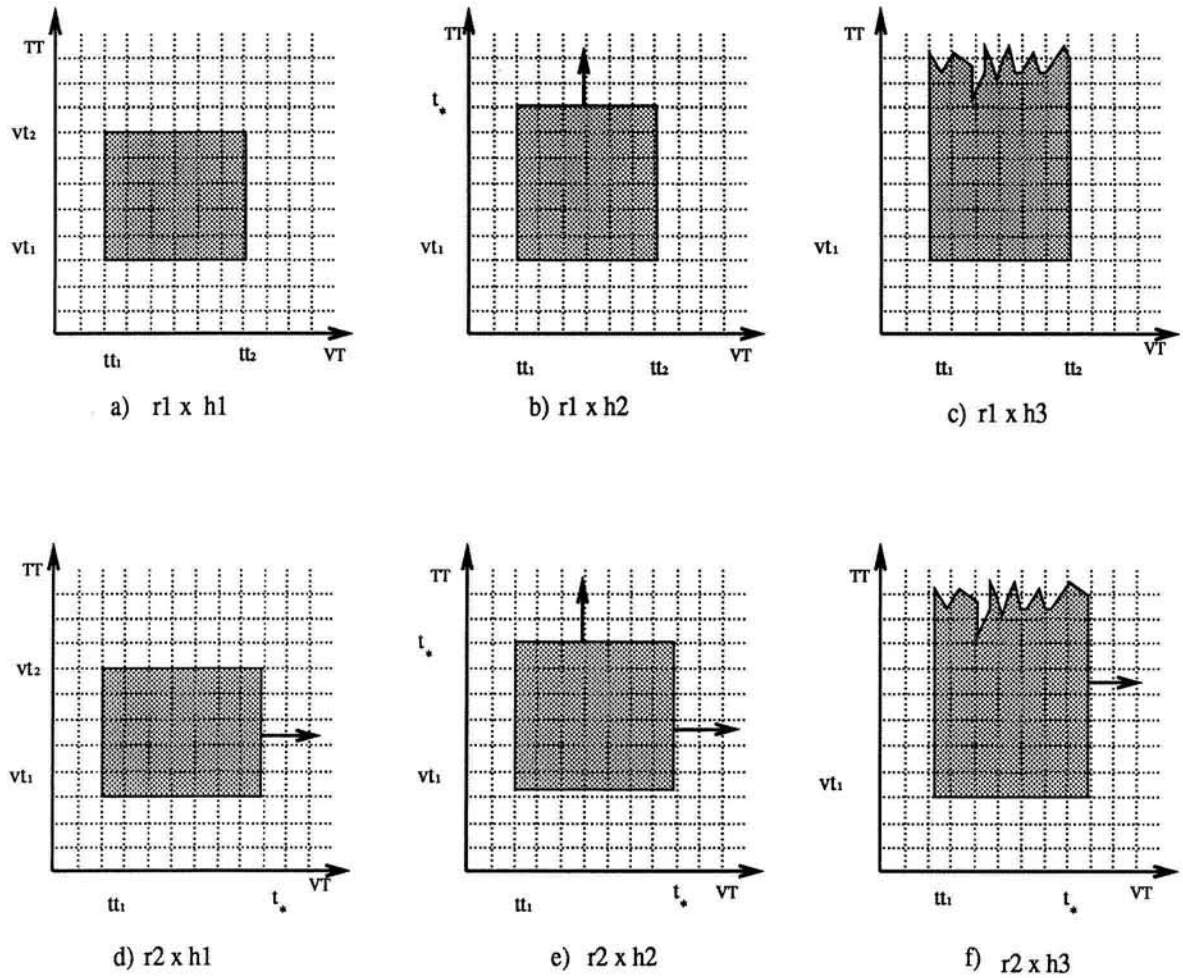

Figure 10: Interactions between transaction and valid time.

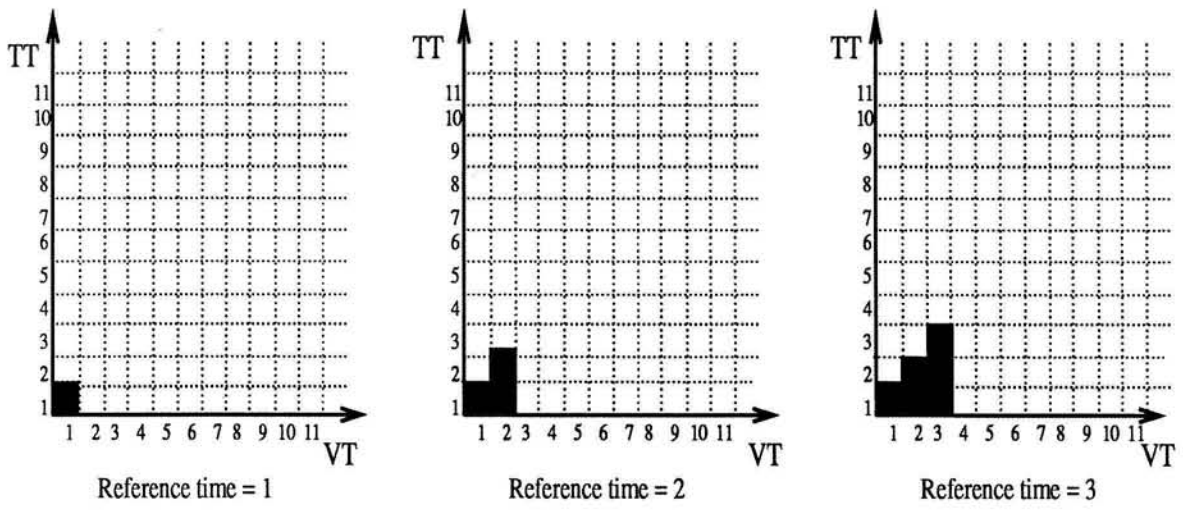

Figure 11: How Jane's rank evolved.

	TT	VT	Logical View
r1× h2'	$[tt_1, tt_2)$	$[vt_1, uc_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt < tt_2 \wedge vt_1 \leq vt \leq tt\}$ $\cup \{(tt, vt_1, a) \mid tt_1 \leq tt < tt_2\}$
r2× h2'	$[tt_1, now_{tt}]$	$[vt_1, uc_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq t_* \wedge vt_1 \leq vt \leq tt\}$ $\cup \{(tt, vt_1, a) \mid tt_1 \leq tt \leq tt_*\}$
r2'× h1	$[tt_1, uc_{tt}]$	$[vt_1, vt_2) \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq vt \wedge vt_1 \leq vt < vt_2\}$ $\cup \{(tt_1, vt, a) \mid vt_1 \leq vt < vt_2\}$
r2'× h2	$[tt_1, uc_{tt}]$	$[vt_1, now_{vt}] \mapsto a$	$[A]_{t_*} = \{(tt, vt, a) \mid tt_1 \leq tt \leq vt \wedge vt_1 \leq vt \leq t_*\}$ $\cup \{(tt_1, vt, a) \mid vt_1 \leq vt \leq t_*\}$

Table 7: A Possible Meaning for *until changed.*

tion about Jane's rank is depicted in the step-wise pattern of Figure 11. Since the shaded areas in the Figure are not rectangular, the intended meaning of the combination of the intervals cannot be that of a Cartesian product. We propose the use of a new symbol uc_{vt} that indicates not a new special variable, but a different way of combining the intervals. Its meaning coincides with the one given in Figure 10, and it is defined in the first two rows of Table 7. The role of the second component of the unions appearing in the table is to allow explicit predictions to be made. For example, if at reference time 2, it is predicted that Tom will start as an Associate at time 3, his record would show $< [2, now)[3, uc) \mapsto$ `Associate` $>$. Hence the triplet $(2, 3,$ `associate`$)$, a prediction, will be in the logical view. Based on these definitions of the variable symbols, we are in a position to recast the original faculty relation of [Sno87] to the relation shown in Figure 12.

The last two rows in Table 7 present the symmetric case where the transaction time is bound by the valid time. In this case, the database *forgets* about transactions that affect valid dates if these are committed after that date. We include them here for completeness. It is not clear whether or not they have any applicability to real-world problems.

FACULTY					
		VALID-TIME		TRANS-TIME	
NAME	RANK	(from)	(to)	(start)	(stop)
Jane	*Assistant*	*1*	uc_{vt}	*1*	5
Jane	*Associate*	*5*	uc_{vt}	*5*	8
Jane	*Full*	*9*	uc_{vt}	*8*	now_{tt}
Merrie	*Assistant*	*7*	uc_{vt}	*6*	11
Merrie	*Associate*	*11*	uc_{vt}	*11*	now_{tt}
Tom	*Associate*	*3*	uc_{vt}	*2*	*4*
Tom	*Assistant*	*3*	uc_{vt}	*4*	*9*
Tom	*Associate*	*10*	uc_{vt}	*9*	now_{tt}

Figure 12: A Fresh View of the Faculty Relation

6 Summary and Conclusions

The interactions between transaction time, valid time, and reference time in bitemporal databases are complex. These interactions have not been thoroughly understood or explicated in the various bitemporal data models that have appeared in the literature.

In this paper we have explored these interactions and attempted to clarify the issues involved. In particular, we have pointed out problems with the widespread use of data models which admit variables, such as "∞", "*now*", and "*uc*", without formally defined semantics. We have attempted to clarify what reasonable semantics can be given to these symbols in two ways, both through he introduction of a two-dimensional, graphical representation for temporal objects, as well as by precise specification of the possible semantics for these two dimensions both independently and in combination. This exploration led to the positing of additional semantic variations generated by possible interactions between the two dimensions. Finally, these explorations enable us to further refine the classification of temporal databases presented in [SA85]. We are presently at work on a logic-based query language based on the semantic analysis presented here.

Several additional research questions are posed by this work. While the step-wise growth semantics for *uc* in Section 5 seems reasonable, other possible combinations of these two

temporal dimensions might also prove useful. In addition, our analysis assumed that the variables such as *now* and *forever* were only used as the upper limit on an interval. Entries that are the temporal mirror-image of those we analyzed, such as $[now, t_1)$, for an entry that progressively *forgets* information as time passes, or $[past - forever, t_1)$, that records a fact as forever true *prior* to some time t_1, might also prove useful. Finally, the use of these bitemporal graphs at the user interface – for displaying the results of queries, for the assertion of temporal integrity constraints, etc. – seems to us a promising one for further research.

References

[Ari86] G. Ariav. A temporally oriented data model. *ACM Transactions on Database Systems*, 11(4):499–527, December 1986.

[BZ82] J. Ben-Zvi. *The Time Relational Model.* PhD thesis, University of California at Los Angeles, 1982.

[CC87] J. Clifford and A. Croker. The historical relational data model HRDM and algebra based on lifespans. In *Proc. Third International Conference on Data Engineering*, pages 528–537, Los Angeles, February 1987. IEEE.

[CCT93] J. Clifford, A. Croker, and A. Tuzhilin. On completeness of query languages for grouped and ungrouped historical data models. In A. Tansel, S. Gadia, S. Jajodia, A. Segev, and Snodgrass R., editors, *Temporal Databases.* Press, 1993. (to appear).

[Cli92] J. Clifford. Indexical databases. In *Proceedings of Workshop on Current Issues in Database Systems*, Newark, N.J., October 1992. Rutgers University.

[CT85] J. Clifford and A.U. Tansel. On an algebra for historical relational databases: Two views. In S. Navathe, editor, *Proceedings of ACM SIGMOD Conference*, pages 247–265, Austin, TX, May 1985. acm.

[CW83] J. Clifford and D. S. Warren. Formal semantics for time in databases. *ACM Transactions on Database Systems*, 6(2):214–254, June 1983.

[Fin92] M. Finger. Handling database updates in two-dimensional temporal logic. *Journal of Applied Non-Classical Logics*, 2(2), 1992.

[Gad88] S. K. Gadia. A homogeneous relational model and query languages for temporal databases. *TODS*, 13(4):418–448, 1988.

[Gad92] S. Gadia. A seamless generic extension of sql for querying temporal data. Technical report, Iowa State University, 1992.

[JCG+92] C.S. Jensen, J. Clifford, S.K. Gaida, A. Segev, and R.T. Snodgrass. A glossary of temporal database concepts. *ACM SIGMOD Record*, 21(3), September 1992.

[JM80] S. Jones and P.J. Mason. Handling the time dimension in a data base. In *Proc. International Conference on Data Bases*, pages 65–83, Heyden, July 1980. British Computer Society.

[JMR89] C. S. Jensen, L. Mark, and N. Roussopoulos. Incremental implementation model for relational databases with transaction time. Technical Report UMIACS-TR-8963/CS-TR-2275, University of Maryland, College Park, MD, jun 1989.

[Lor87] R.G. Lorentzos, N.A.; Johnson. TRA: A model for a temporal relational algebra. In *Proceedings of the Conference on Temporal Aspects in Information Systems*, pages 99–112, France, May 1987. AFCET.

[LS92] D. Lomet and B. Salzberg. Rollback databases. Technical Report NU-CCS-92-3, Northeastern University, 1992.

[Mon73] Richard Montague. The proper treatment of quantification in english. In K.J.J. Hintikka, editor, *Approaches to Natural Language*, pages 221–242. Dordrecht, 1973. Reprinted in [Mon74].

[Mon74] R. Montague. *Formal Philosophy: Selected Papers of Richard Montague.* Yale University Press, New Haven, 1974.

[MS91] E. McKenzie and R. Snodgrass. Supporting valid time in an historical relational algebra: Proofs and extensions. Technical Report TR–91–15, Department of Computer Science, University of Arizona, Tucson, AZ, August 1991.

[NA89] S. B. Navathe and R. Ahmed. A temporal relational model and a query language. *Information Sciences*, 49(2):147–175, 1989.

[Rei84] R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling*, pages 191–233. Springer, 1984.

[SA85] R. Snodgrass and I. Ahn. A taxonomy of time in databases. In *Proceedings of ACM SIGMOD Conference*, pages 236–246, New York, 1985. ACM.

[Sar90] N.L. Sarda. Algebra and query language for a historical data model. *The Computer Journal*, 33(1):11–18, February 1990.

[Sno] R. Snodgrass, *private communication*, 1992.

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

[Tan86] A.U. Tansel. Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4):343–355, 1986.

[WJL93] G. Wiederhold, S. Jajodia, and W. Litwin. Integrating temporal data in a heterogeneous environment. In A. Tansel, S. Gadia, S. Jajodia, A. Segev, and Snodgrass R., editors, *Temporal Databases.* Press, 1993. (to appear).