The TEMPIS Project

Proposal for a Data Model for the Temporal Structured Query Language

Christian S. Jensen and Richard Snodgrass

July, 1992

TEMPIS Technical Report No. 37

Department of Computer Science
University of Arizona
Tucson, AZ 85721

The TEMPIS Project

Proposal for a Data Model for the Temporal Structured Query Language

Christian S. Jensen and Richard Snodgrass

July, 1992

Abstract

Adding time to the relational model has been a daunting task. More than two dozen time-extended relational data models have been proposed over the last fifteen years. We feel that the reason why so many temporal data models have been proposed is that these models attempt to simultaneously retain the simplicity of the relational model, present all the information concerning an object in one tuple, and ensure ease of implementation and query evaluation efficiency. We advocate instead a separation of concerns. We propose a new, conceptual temporal data model that better captures the time-dependent semantics of the data while permitting multiple data models at the representation level. This conceptual model effectively moves the distinction between the various existing data models from a semantic basis to a physical, performance-relevant basis. We compare our model with the previously proposed temporal data models.

TEMPIS Technical Report No. 37

Department of Computer Science
University of Arizona
Tucson, AZ 85721

Contents

List of Figures

List of Tables

1 Introduction

Adding time to the relational model has been a daunting task [Bolour et al. 1982, McKenzie 1986, Soo 1991, Stam & Snodgrass 1988]. More than two dozen time-extended relational data models have been proposed over the last fifteen years [Snodgrass 1992]. Most of these are *valid-time* models. Each fact in a valid-time relation has associated the time when it is true in the modeled reality. Other models support *transaction-time* relations where each fact has associated the time when it is current in the database. A few support both valid and transaction time [Ben-Zvi 1982, Bhargava & Gadia 1989A, Snodgrass 1987, Snodgrass 1993, Thompson 1991]; such models are termed *bitemporal*. As a whole, these data models are referred to as *temporal* data models [Jensen et al. 1992A].

We propose a new data model as a basis for the Temporal Structured Query Language (TSQL) extension to SQL. A data model can be said to consist of a query language, objects manipulated by the query language, an update language for updating the objects, and a mechanism for specifying integrity constraints. In this proposal, we concentrate on the objects, temporal relations. Subsequent proposals will address historical selection and projection, aggregates, and the other aspects necessary to define a comprehensive extension to SQL incorporating time.

While existing data models differ on many dimensions, perhaps the most frequently stated distinction between approaches is the one between tuple timestamping and first normal form (1NF), on one hand, and attribute-value timestamping and non-1NF, on the other. Each of the two approaches has associated difficulties. Remaining within 1NF (an example being the timestamping of tuples with valid and transaction start and end times [Snodgrass 1987]) may introduce redundancy because attribute values that change at different times are repeated in multiple tuples. The non-1NF models, one being timestamping attribute values with sets of intervals [Gadia 1988], may not be capable of directly using existing relational storage structures or query evaluation techniques that depend on atomic attribute values.

Today there exists a plethora of incompatible data models and query languages, with a corresponding surfeit of model- and language-specific database design and implementation strategies. It is our contention that the simultaneous focus on data *presentation* (how temporal data is displayed to the user), on data *storage*, with its requisite demands of regular structure, and on efficient *query evaluation* is a major reason why such a large number of very diverse data models exists. Further, we find that these simultaneous foci have complicated existing data models and made them less suited for the central data model task of capturing the time semantics of data.

Consequently, we advocate a very simple *conceptual,* unifying data model that captures the essential semantics of time-varying relations, but has no illusions of being suitable for presentation, storage, or query evaluation. For the other tasks, we are able to use the existing data models. Specifically, we have utilized the notion of *snapshot equivalence* to demonstrate equivalence mappings between the conceptual model and several *representational* models [Jensen et al. 1992B]. Snapshot equivalence formalizes the notion of having the same information contents and is a natural means of comparing rather disparate representations. Two relation instances are snapshot equivalent if all their snapshots, taken at all times (valid and transaction), are identical.

Facts in temporal relations (valid-time, transaction-time, or bitemporal) have associated times. Thus, we start out by examining the time domain itself. In Section 3, we then review, in turn, how times have previously been associated with facts of valid-time, transaction-time, and bitemporal relations. This review of 23 models and a subseqent comparison sets the scene for the presentation of the new proposal. The basic notion of a conceptual bitemporal relation is first presented. Then the semantics of the bitemporal relations is illuminated by defining insertion, deletion, and modification operators. Finally, the data model is evaluated and compared with other previously developed data models.

2 The Time Domain

In this section we focus on time itself. The topic of the next section is then how time may be combined with facts to model time-varying information.

We initially assume that there is one dimension of time. The distinctions addressed here will apply to each of the several dimensions considered later in this section.

Early work on *temporal logic* centered around two structural models of time, *linear* and *branching* [Van Benthem 1982]. In the linear model, time advances from the past to the future in a totally ordered fashion. In the branching model, also termed the *possible futures* model, time is linear from the past to now, where it then divides into several time lines, each representing a possible sequence of events [Worboys 1990]. Along any future path, additional branches may exist. The structure of branching time is a tree rooted at now. The most general model of time in a temporal logic represents time as an arbitrary set with a partial order imposed on it. Additional axioms introduce other, more refined models of time. For example, linear time can be specified by adding an axiom imposing a total order on this set. Recurrent processes may be associated with a *cyclic* model of time [Chomicki & Imelinski 1989, Lorentzos & Johnson 1988, Lorentzos 1988].

Axioms may also be added to temporal logics to characterize the *density* of the time line [Van Benthem 1982]. Combined with the linear model, *discrete* models of time are isomorphic to the natural numbers, implying that each point in time has a single successor [Clifford & Tansel 1985]. *Dense* models of time are isomorphic to either the rationals or the reals: between any two moments of time another moment exists. *Continuous* models of time are isomorphic to the reals, i.e., they are both dense and, unlike the rationals, contain no "gaps."

In the continuous model, each real number corresponds to a "point" in time; in the discrete model, each natural number corresponds to a nondecomposable unit of time with some fixed, arbitrary duration. Such a nondecomposable unit of time is referred to as a *chronon* [Ariav 1986, Clifford & Rao 1987] (other, perhaps less desirable, terms include "time quantum" [Anderson 1982], "moment" [Allen & Hayes 1985], "instant" [Gadia 1986] and "time unit" [Navathe & Ahmed 1987, Tansel & Arkun 1986A]). A chronon is the smallest duration of time that can be represented in this model. It is not a point, but a line segment on the time line.

Although time itself is generally perceived to be continuous, most proposals for adding a temporal dimension to the relational data model are based on the discrete time model. Several practical arguments are given in the literature for this preference for the discrete model over the continuous model. First, measures of time are inherently imprecise [Anderson 1982, Clifford & Tansel 1985]. Clocking instruments invariably report the occurrence of events in terms of chronons, not time "points." Hence, events, even so-called "instantaneous" events, can at best be measured as having occurred during a chronon. Secondly, most natural language references to time are compatible with the discrete time model. For example, when we say that an event occurred at 4:30 p.m., we usually don't mean that the event occurred at the "point" in time associated with 4:30 p.m., but at some time in the chronon (perhaps minute) associated with 4:30 p.m. [Anderson 1982, Clifford & Rao 1987, Dyreson & Snodgrass 1992A]. Thirdly, the concepts of chronon and interval allow us to naturally model events that are not instantaneous, but have duration [Anderson 1982]. Finally, any implementation of a data model with a temporal dimension will of necessity have to have some discrete encoding for time.

Axioms can also be placed on the *boundedness* of time. Time can be bounded orthogonally in the past and in the future.

Models of time may include the concept of *distance* (most temporal logics do not do so, however). Time is a *metric*, in that it has a distance function satisfying four properties: (1) the distance is nonnegative, (2) the distance between any two non-identical elements is non-zero, (3) the distance from

time α to time β is identical to the distance from β to α, and (4) the distance from α to γ is equal to or greater than the distance from α to β plus the distance from β to γ (the triangle inequality).

With distance and boundedness, restrictions on range can be applied. The scientific cosmology of the "Big Bang" posits that time begins with the Big Bang, 14 ± 4 billion years ago. There is much debate on when it will end, depending on whether the universe is *open* or *closed* (Hawking provides a readable introduction to this controversy [Hawking 1988]). If the universe is closed then time will have an end when the universe collapses back onto itself in what is called the "Big Crunch." If it is open then time will go on forever.

Finally, one can differentiate *relative* time from *absolute* time (more precise terms are *unanchored* and *anchored*). For example, "9 a.m., January 1, 1992" is an absolute time, whereas "9 hours" is a relative time. This distinction, though, is not as crisp as one would hope because absolute time is with respect to another time (in this example, midnight, January 1, A.D. 1). Relative time differs from distance in that the former has a direction, e.g., one could envision a relative time of -9 hours, whereas a distance is unsigned.

Time is multi-dimensional [Snodgrass & Ahn 1986]. *Valid time* concerns the time when a fact is true in reality. The valid time of an event is the wall clock time at which the event occurred in the modeled reality, independent of the recording of that event in some database. Valid times can be in the future, if it is known that some fact will become true at a specified time in the future. *Transaction time* concerns the time the fact was present in the database as stored data. The transaction time (a set of intervals) of an event identifies the transactions that inserted the information about the event into the database and removed this information from the database. Note that these two time dimensions are orthogonal. A data model supporting neither is termed *snapshot*, as it has no built-in support for any of these notions of time. A data model supporting only valid time is termed *valid-time*; one that supports only transaction time is termed *transaction-time*; and one that supports both valid and transaction time is termed *bitemporal* (*temporal* is a generic term implying some kind of time support [Jensen et al. 1992A]).

While valid time may be bounded or unbounded (as we saw, cosmologists feel that it is at least bounded in the past), transaction time is always bounded on both ends. Specifically, transaction time starts when the database is created (before which time, nothing was stored), and does not extend past now (no facts are known to have been stored in the future). Changes to the database state are required to be stamped with the current transaction time. As the database state evolves, transaction times grow monotonically, and successive transactions have successive transaction times associated. In contrast, successive transactions may mention widely varying valid times.

Unlike the spatial dimensions, the two time dimensions are not homogeneous—transaction time has a different semantics than valid time. Valid and transaction time *are* orthogonal, though there are generally some application-dependent correlations between the two times. As a simple example, consider the situation where a fact is recorded as soon as it becomes valid in reality. In such a *specialized* bitemporal database, termed *degenerate* [Jensen & Snodgrass 1992], the valid and transaction times of a fact are identical. As another example, if temperature measurements in a chemical experiment are recorded at most two minutes after they were measured, and if it takes at least five seconds from the measurement time to record the measurement, then such a database is *delayed strongly retroactively bounded with bounds five seconds and two minutes.*

3 Previous Data Models

The previous section explored models for the time domain itself. Next, we discuss the association of facts with times. Specifically, we survey 23 existing data models that have been proposed over the last fifteen year. We consider each model in turn, starting with valid-time models, continuing with transaction-time models, and ending with bitemporal models. As a foundation for this, we initially define underlying concepts. Following the survey, we compare and categorize the data models with respect to fundamental design decisions.

3.1 Underlying Concepts

It is advantageous to examine several central concepts before each of the proposed data models are considered in turn.

3.1.1 Timestamp Types

We may distinguish between three semantically different types of time values, namely single chronons, sets of consecutive chronons, and arbitrary sets of chronons. These are termed *events*, *intervals*, and *temporal elements*, respectively [Jensen et al. 1992A].

A single event may be represented by a single, atomic, chronon-valued attribute. An interval may be represented by a pair of atomic attribute values, each of which is a chronon or a point in time. If the later representation is adopted, the interval may be defined as open, half-closed, or closed. An interval may also be encoded in a single, atomic, interval-valued attribute. An arbitrary set of chronons may be represented by a non-atomic attribute value. This value may be a set of intervals, each interval defining a set of consecutive chronons, or it may simply be a set of chronons. Finally, sets of multiple chronons, consecutive or not, may be represented via multiple tuples, one tuple per chronon or one per interval.

This discussion applies to both transaction time, valid time, and the combination of valid and transaction time. For example, a *bitemporal element* is a set of *bitemporal chronons* in the transaction-time/valid-time space, and can be represented simply as a set of bitemporal chronons, as a set of contiguous or overlapping rectangles, or via multiple tuples, one tuple per bitemporal chronon or bitemporal rectangle.

3.1.2 Attribute Variability

Attributes are commonly categorized based on how they interact with time. A *time-invariant* attribute [Navathe & Ahmed 1989] does not change over time.

The key value in a tuple of a relation instance is commonly used to identify the object, entity or relationship, in the modeled reality that the remaining attribute values of the tuple are about. If the key value changes, the tuple represents another object. Thus, the key of a relation schema is time invariant in such models. For example, attribute Name is a time-invariant key in relation schema $R =$ (Name, Course) recording the courses taken by a student population. Time invariance is not restricted to key attributes. The attribute "place of birth" is an example. Note that time invariance generally is applied to valid time. The place of birth might have been in error; in that case, the old tuple would be (logically) deleted and a new tuple with the correct place of birth inserted.

Other models identify the objects that the tuples in a relation instance are about by means of surrogates which are system-generated, unique identifiers that can be referenced and compared for equality, but not displayed to the user [Hall et al. 1976]. Surrogates are by definition time invariant.

The opposite of time invariant is *time varying*. Examples abound. In the schema R above, the courses taken by a student varies over time, and the attribute Course is time varying.

The *value* of an attribute may be drawn from a temporal domain. Such temporal domains are termed *user-defined time* [Snodgrass & Ahn 1986]; other than being able to be read in, displayed, and perhaps compared, no special semantics is associated with such domains. Interestingly, most such attributes are time-invariant. The attribute "time of birth" is an example.

3.1.3 Implicit Versus Explicit Timestamps

In some data models, the association of times with facts is implicit; in other models, this association is represented by fully explicit timestamp attributes. We shall now see how this distinction is relevant to three aspects of a data model: update language, display of data, and query language.

The transaction times of facts are supplied by the system itself. Thus, update languages of transaction-time models treat the temporal aspect of facts implicitly. In contrast, the valid times of facts are usually supplied by the user. Thus, update languages of valid-time and bitemporal data models generally must treat time explicitly and are forced to represent a choice as to how the valid times of facts should be specified by the user. At best such data models can allow the the user to choose between several formats.

If, in a data model, it is possible to display directly temporal facts, i.e., facts with associated times, then, as for update, the data model necessarily must treat time explicitly. At best, the model may allow a variety of display formats for temporal facts. Unlike for update, the possibility exists that temporal facts cannot be displayed. This option is especially feasible for the relatively simple transaction time models, and thus the display of facts in these models needs not reveal how time is associated with facts.

The query language aspect of the distinction between implicit and explicit timestamps is by far the most complex. If the temporal aspects of facts are represented by attributes, and it is possible in the query language to directly access these attributes then the temporal attributes are just like other attributes—they are explicit. On the other hand, if the timestamp attributes used for associating times with facts are not accessible directly through the query language, but are instead processed internally by queries, then the particular scheme for associating timestamps with facts is invisible to the user of the query language.

3.1.4 Temporal Homogeneity

When several temporal facts pertain to the same object (usually the object is a tuple), the concept of temporal homogeneity surfaces. A tuple is *temporally homogeneous* if each of its facts are defined over the same temporal element [Gadia 1988]. A temporal relation is said to be temporally homogeneous if its tuples are temporally homogeneous [Jensen et al. 1992A]. Further, a temporally homogeneous relation schema is restricted to have only temporally homogeneous relation instances. In addition to being specific to a type of object, homogeneity may be applied to both the valid and the transaction time dimension.

The motivation for homogeneity arises from the fact that the process of deriving a snapshots from of a homogeneous relation does not produce null values.

Certain data models assume temporal homogeneity. Models that employ tuple timestamping rather than attribute value timestamping are necessarily temporally homogeneous—only temporally homogeneous relations are possible.

3.1.5 Value Equivalence and Coalescing

Two tuples are termed *value equivalent* if, when disregarding special timestamp attributes, they are identical. A relation instance is *coalesced* if overlapping or consecutive, value-equivalent tuples are disallowed. Here "overlapping" and "consecutive" are with respect to the timestamp attribute value(s) of the tuples, which must specify a single chronon or a set of consecutive chronons.

When timestamps of tuples have temporal elements as values, the requirement of coalescing is identical to the requirement that there be no value-equivalent tuples present.

3.2 Overview

Over two dozen extensions to the relational model to incorporate time have been proposed over the last 15 years. With a focus on the types of relations they provide, we now review 23 of these temporal data models.

Table 1 lists most of the temporal data models that have been proposed to date. If the model is not given a name, we appropriate the name given the associated query language, where available. Many models are described in several papers; the one referenced is the initial journal paper in which the model was defined. Some models are defined only over valid time or transaction time; others are defined over both. The last column indicates a short identifier which denotes the model; the table is sorted on this column.

We omit a few intermediate data models, specifically Gadia's multihomogeneous model [Gadia 1986], which was a precursor to his heterogeneous model (Gadia-2), and Gadia's two-dimensional temporal relational database model [Bhargava & Gadia 1989B], which is a precursor to Gadia-3. We also do not include the data model used as the basis for defining temporal relational completeness [Tuzhilin and Clifford 1990] because it is a generic data model purposefully designed not to force decisions on most of the aspects to be discussed here.

We first examine the valid-time models that timestamp tuples, then discuss those that timestamp attribute values. We'll proceed chronologically (of course!) We then examine the transaction-time models, and finish up with the bitemporal models that support both valid and transaction time.

3.3 Valid-time Models

Approximately half the proposed temporal data models support only valid time.

Brooks The first academic treatment of time in databases was the dissertation of Frederick Brooks, Jr., which proposes a three-dimensional view of a valid-time database [Brooks 1956]. Subsequent proposals, notably Ahn, Ariav, Clifford-1 and McKenzie, have emphasized this fruitful "cubic" analogy.

Wiederhold The data model associated with the Time Oriented Data Base (TOD) was developed specifically to support medical applications. In this pioneering model, relations were sets of entity-attribute-time-value quadruples [Wiederhold et al. 1975] or, for each attribute, sequences of events represented as pairs of visit number and value or intervals represented as sequences of pairs of visit numbers and sequences of values [Blum 1981]. Timestamping is indirect through the visit number; a separate array associates each visit with a particular date. This was probably done because many measurements are taken each visit. This structure was further elaborated as *time sequences* in Segev's model.

Data Model	*Citation*	*Time Dimension(s)*	*Identifier*
—	[Snodgrass & Ahn 1986]	both	Ahn
Temporally Oriented Data Model	[Ariav 1986]	valid	Ariav
Time Relational Model	[Ben-Zvi 1982]	both	Ben-Zvi
—	[Brooks 1956]	valid	Brooks
Historical Data Model	[Clifford & Warren 1983]	valid	Clifford-1
Historical Relational Data Model	[Clifford & Croker 1987]	valid	Clifford-2
Homogeneous Relational Model	[Gadia 1988]	valid	Gadia-1
Heterogeneous Relational Model	[Gadia & Yeung 1988]	valid	Gadia-2
TempSQL	[Gadia 1992]	both	Gadia-3
DM/T	[Jensen et al. 1991]	transaction	Jensen
LEGOL 2.0	[Jones et al. 1979]	valid	Jones
DATA	[Kimball 1978]	transaction	Kimball
Temporal Relational Model	[Lorentzos & Johnson 1988]	valid	Lorentzos
—	[McKenzie & Snodgrass 1991]	both	McKenzie
Temporal Relational Model	[Navathe & Ahmed 1989]	valid	Navathe
HQL	[Sadeghi 1987]	valid	Sadeghi
HSQL	[Sarda 1990A]	valid	Sarda
Temporal Data Model	[Segev & Shoshani 1987]	valid	Segev
TQuel	[Snodgrass 1987]	both	Snodgrass
Postgres	[Stonebraker 1987]	transaction	Stonebraker
HQuel	[Tansel 1986]	valid	Tansel
Accounting Data Model	[Thompson 1991]	both	Thompson
Time Oriented Data Base Model	[Wiederhold et al. 1975]	valid	Wiederhold

Table 1: Temporal Data Models

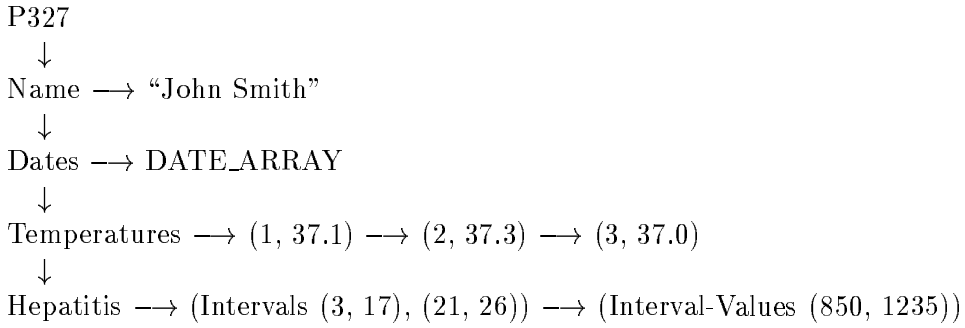

Figure 1: A Time-Oriented Record for a Hypothetical Patient

EXAMPLE: For the patient whose record is shown in Figure 1 [Blum 1981], John Smith's temperature was recorded during visit 1 (July 24, 1970, as recorded in the DATE_ARRAY) as 37.1°. He experienced two episodes of hepatitis, the first from visits 3 to 17, with a maximum of 850 International Units of SGOT during that interval of time. □

Jones LEGOL 2.0 [Jones et al. 1979] is a language designed to be used in database applications such as legislative rules writing and high-level system specification in which the temporal ordering of events and the valid times for objects are important. It was the first time-oriented algebra defined; it introduced many of the features found in later algebras.

Objects in the LEGOL 2.0 data model are relations as in the relational data model, with one distinction. Tuples in LEGOL 2.0 are assigned two implicit time attributes, Start and Stop. The values of these two attributes are the chronons corresponding to the (inclusive) end-points of the interval of existence (i.e., valid time) of the entity or relationship in the modeled reality represented by a tuple; these values are specified during data entry by the user.

EXAMPLE: Let R be a relation schema in LEGOL 2.0 that records the courses taken by a student population. The schema has the two explicit attributes, Name and Course. An instance of R is shown in Figure 2. We use 1 to represent the Fall semester 1980, 2 to represent the Spring semester 1981, and so on. Later examples will show the semantically equivalent representation of this instance in other data models. Because the data models all define relations differently and, in some cases, require implicit attributes, we show all relation examples in tabular form for both clarity and consistency of notation. This relation shows that Bill was a student in the English course for the Fall 1980 semester and for the Fall 1981 and Fall 1982 semesters. □

Clifford-1 In the *Historical Database Model*, an additional, chronon-valued attribute, STATE, is part of each relation schema. A boolean attribute, EXISTS, is also added to indicate whether the particular tuple exists for that state [Clifford 1982, Clifford & Warren 1983].

Ariav In the Temporally Oriented Data Model, a valid-time relation is a sequence of snapshot relation states, indexed by valid time, termed the *data cube* [Ariav 1986]. Associated with this data model is a calculus-based query language, TOSQL.

Name	Course	Start	Stop
Bill	English	1	1
Bill	English	3	4
George	English	1	2
George	Math	5	6

Figure 2: An Example Relation with Time

Navathe The Temporal Relational Model [Navathe & Ahmed 1987] and its associated algebra were defined primarily to support TSQL [Navathe & Ahmed 1989], a temporal extension to SQL defined in the same paper. This valid-time model allows both non-time-varying and time-varying attributes, but all of a relation's attributes must be of the same type. Objects are classified as: snapshot relations, whose attributes are all non-time-varying, and valid-time relations, whose non-key attributes are all time-varying. Each tuple has associated an interval of validity which is recorded in two mandatory time attributes, Time-start and Time-end. The structure of a valid-time relation in the Temporal Relational Model is the same as that of a valid-time relation in LEGOL 2.0 (Figure 2), with one additional restriction: Value-equivalent tuples, although allowed, are required to be coalesced.

Sadeghi Sadeghi's data model [Sadeghi 1987] is similar in many ways to Navathe's. It was designed to support the calculus-based valid-time query language HQL [Sadeghi et al. 1987], which in turn is based on DEAL [Deen 1985]. In Sadeghi's data model, all objects are valid-time relations. Two implicit attributes, Start and Stop, record the end-points of each tuple's interval of validity. Hence, the structure of a valid-time relation in Sadeghi's model is also the same as that of the valid-time relation in LEGOL 2.0 (Figure 2). Sadeghi's data model requires coalescing.

Sarda Sarda's data model and associated algebra [Sarda 1990B] were designed to support the calculus-based query language HSQL [Sarda 1990A]. This model associates valid time with tuples. Objects can be either snapshot or valid-time relations. Unlike the data models mentioned previously, Sarda's model represents valid time in a valid-time relation as a single, non-atomic, implicit attribute named Period. Also unlike the previous models, a tuple in Sarda's model is not considered valid at its right-most boundary point, i.e., the interval is closed on the left and open on the right.

EXAMPLE: The relation in Figure 3 is a valid-time relation instance in Sarda's model. The first two tuples signify that Bill was enrolled in English during the Fall semester 1980 and the Fall semesters 1981 and 1982, but not during the Spring semester 1981. □

The remaining data models employ distinct non-first-normal form data models, with attribute value timestamping and perhaps with multiple values per attribute. The non-atomicity of attribute values is due to their time-varying nature; any timeslice will usually be in first normal form. Hence, the data models are an extension of the conventional (1NF) relational model; the representation, viewed as a normal relation, is certainly not in 1NF, but then the operators included in the models do not operate on conventional relations—they operate on valid-time relations, which are extensions of conventional relations.

Name	Course	Period
Bill	English	1...2
Bill	English	3...5
George	English	1...3
George	Math	5...7

Figure 3: The Example Relation in Sarda's Data Model

Segev The principal structure of the Temporal Data Model is the *time sequence*, which is a so-called surrogate value identifying the object along with a sequence of time-value pairs [Segev & Shoshani 1987]. There are a variety of time sequences, depending on the assumptions made about the values at points of time intermediate to the points explicitly represented. For a bank account balance, step-wise constant behavior would be assumed; for a time sequence recording the number of copies sold on a day for a particular book, discrete behavior would be assumed; and for measurement of a magnetic field taking at regular intervals, continuous behavior would be assumed. A *time sequence collection* (TSC) is then a set of time sequences.

Clifford-2 The *Historical Relational Data Model* [Clifford & Croker 1987], a refinement of the model associated with a valid-time algebra [Clifford & Tansel 1985], is unique in that it associates timestamps with both individual tuples and with individual attribute values of the tuples. The data model allows two types of objects: a set of chronons, termed a *lifespan*, and a valid-time relation, where each attribute in the relation schema and each tuple in the relation is assigned a lifespan. A relation schema in the Historical Relational Data Model is an ordered four-tuple containing a set of attributes, a set of key attributes, a function that maps attributes to their lifespans, and a function that maps attributes to their value domains. A tuple is an ordered pair containing the tuple's value and its lifespan. Attributes are not atomic; rather, an attribute's value in a given tuple is a partial function from a domain of chronons onto the attribute's value domain. The domain of chronons is defined as the the intersection of the lifespan for the particular attribute and tuple. Relations have key attributes and no two tuples in a relation are allowed to match on the values of the key attributes at the same chronon.

EXAMPLE: Figure 4 illustrates the valid-time relation instance in the Historical Relational Data Model, where {Name $\rightarrow$ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, Course $\rightarrow$ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}} is the function assigning lifespans to attributes, and the attribute Name is the key.

Because tuple lifespans are sets and because both Bill and George were never enrolled in more than one course at the same time, we are able to record each of their enrollment histories in a single tuple. If one had been enrolled in two or more courses at the same time, however, his total enrollment history could not have been recorded in a single tuple as attribute values are functions from a lifespan onto a value domain. Note also that we have chosen the most straightforward representation for an attribute whose value is a function. Because attribute values in both Clifford's model and Gadia's models, which we describe later, are functions, they have many physical representations. □

Tansel Tansel's model [Clifford & Tansel 1985, Tansel 1986] was designed to support the calculus-based query language HQuel [Tansel & Arkun 1986B] and, later, the Time-by-Example language [Tansel

Tuple Value		*Tuple Lifespan*
Name	Course	
$1 \rightarrow$ Bill $3 \rightarrow$ Bill $4 \rightarrow$ Bill	$1 \rightarrow$ English $3 \rightarrow$ English $4 \rightarrow$ English	$\{1, 3, 4\}$
$1 \rightarrow$ George $2 \rightarrow$ George $5 \rightarrow$ George $6 \rightarrow$ George	$1 \rightarrow$ English $2 \rightarrow$ English $5 \rightarrow$ Math $6 \rightarrow$ Math	$\{1, 2, 5, 6\}$

Figure 4: The Example Relation in the Clifford-2 Data Model

Name	Course
Bill	{ ([1, 2), English), ([3, 5), English) }
George	{ ([1, 3), English), ([5, 7), Math) }

Figure 5: The Example Relation in Tansel's Data Model

et al. 1989]. The model allows only one type of object: the valid-time relation. However, four types of attributes are supported: Attributes may be either non-time-varying or time-varying, and they may be either atomic-valued or set-valued. The attributes of a relation need not be the same type, and attribute values in a given tuple need not be homogeneous. The value of a time-varying, atomic-valued attribute is represented as a triplet containing an element from the attribute's value domain and the boundary points of its interval of existence while the value of a time-varying, set-valued attribute is simply a set of such triplets.

EXAMPLE: Figure 5 shows the valid-time relation instance in Tansel's data model, where Name is a non-time-varying, atomic-valued attribute and Course is a time-varying, set-valued attribute.

The enrollment history of a student can be recorded in a single tuple, even if the student was enrolled in two or more courses at some time. Note, however, that each interval of enrollment, even for the same course, must be recorded as a separate element of a time-varying, set-valued attribute. □

Gadia-1 Gadia's homogeneous model [Gadia 1988] allows two types of objects: valid-time elements [Gadia & Vaishnav 1985] and valid-time relations. Valid-time elements are closed under union, difference, and complementation, unlike intervals. The model requires that all attribute values in a given tuple be functions on the same valid-time element, i.e., homogeneity.

Name	Course
$[1, 2) \cup [3, 5) \rightarrow$ Bill	$[1, 2) \cup [3, 5) \rightarrow$ English
$[1, 3) \cup [5, 7) \rightarrow$ George	$[1, 3) \rightarrow$ English $[5, 7) \rightarrow$ Math

Figure 6: The Example Relation in the Gadia-1 Data Model

EXAMPLE: Figure 6 depicts the relation instance in Gadia's homogeneous model. Here the interval $[t_1, t_2)$ is the set of chronons $\{t_1, \cdots, t_2 - 1\}$. Again, we are able to record the enrollment histories of Bill and George in single tuples only because they were never enrolled in more than one course at the same time (otherwise multiple tuples are required). □

Bhargava's 2-dimensional model [Bhargava & Gadia 1990, Bhargava & Gadia 1991] is an extension of Gadia's homogeneous model; it supports both valid and transaction time. Many of the criteria concerning transaction time that are satisfied by the data model discussed below are also satisfied by Bhargava's data model.

Gadia-2 Gadia's multihomogeneous model [Gadia 1986] and Yeung's heterogeneous models [Gadia & Yeung 1988, Yeung 1986] are all extensions of the homogeneous model. They lift the restriction that all attribute values in a tuple be functions on the same temporal element, in part to be able to perform Cartesian product without loss of temporal information caused by merging two timestamps into one. We consider here only the latest [Gadia & Yeung 1988] of these extensions. In this data model (termed Gadia-2), temporal elements may be multi-dimensional to model different aspects of time (e.g., valid time and transaction time). Attribute values are still functions from temporal elements onto attribute value domains, but attribute values need not be functions on the same temporal element. As a result of the lack of temporal homogeneity, some timeslices may produce nulls. Relations are assumed to have key attributes, with the restriction that such attributes be single-valued over their interval of validity. Also, no two tuples may match on the ranges of the functions assigned to the key attributes. Hence, in the previous example, the attribute Name would qualify as a key attribute in the heterogeneous model.

Lorentzos The *Temporal Relational Model* [Lorentzos & Johnson 1988, Lorentzos 1988] was the first to support nested specification of timestamps using values of different granularity and to support periodic events. As with the data models discussed above, this model associates timestamps with individual attribute values rather than with tuples. Although a timestamp is normally associated with each of the attribute values in a tuple, a timestamp may be associated with any non-empty subset of attribute values in a tuple. Furthermore, no implicit or mandatory timestamp attributes are assumed. Timestamps are simply explicit, numeric-valued attributes, to be viewed and updated directly by the user. They represent either the chronon during which one or more attribute values are valid or a *boundary point* of the interval of validity for one or more attribute values. A timestamp in the Temporal Relational Model, like one in Sarda's model, does not include its right-most boundary point. Several timestamp attributes of nested granularity may also be used together in a specification of a chronon.

EXAMPLE: Let R be a valid-time relation schema in the Temporal Relational Model defined by R = (Name, Course, Semester-start, Semester-stop, Week-start, Week-stop) where all four timestamp

Name	Course	Semester-start	Semester-stop	Week-start	Week-stop
Bill	English	1	2	1	9
Bill	English	3	5	1	17
George	English	1	3	1	9
George	Math	5	7	9	17

Figure 7: The Example Relation in Lorentzos' Data Model

attributes are associated with both Name and Course. Assume that the granularity for the timestamp attributes Week-start and Week-stop is a week relative to the first week of a semester. Figure 7 shows the an instance of this relation schema. In this example, we specify the weeks during a semester when a student was enrolled in a course. For example, Bill was enrolled in English during the Fall semester 1980 for only the first 8 weeks of the semester. Note that the meaning of the Week-start and Week-stop attributes is relative to the Semester-start and Semester-stop attributes. □

The data model thus differs from the normal relational model only in that certain columns are given a specific interpretation as representing the period of validity of other column(s) in the relation.

3.4 Transaction-time Models

Transaction-time data models have the valuable property that the objects are *append-only*.

Kimball In the data model termed DATA [Kimball 1978], the association of facts with times is fully implicit. Being a transaction-time model, update operations avoid the explicit mention of time, and do not reveal how times and facts are associated. Next, transaction-time relations cannot be displayed—only snapshot extracted from the transaction-time relations can be displayed. Thus, display does not reveal the particular association of facts and time, either. Finally, the association of facts and times is implicit in the query language—the notion of an explicit timestamp attribute is absent. The consequence is that a user has no way of knowing whether, e.g., timestamps are assigned on the attribute-value level or on the tuple level. Similarly, there is no way to see whether transaction-time event, interval, or element stamping is used.

The DATA data model is implemented using a combination of event-stamped tuples and pointers to predecessor tuples.

Stonebraker The Postgres Data Model [Rowe & Stonebraker 1987] supports transaction time. As for the previous model, the association of facts with time is implicit with respect to the update language, the query language, and the display of facts. Unlike the previous model, display is not restricted to snapshot states as a relation containing all tuples is a sequence of states may be displayed as well. Such a relation is still a conventional snapshot relation.

In the Postgres system, transaction-time relations are implemented using two timestamp attributes specifying the time when the particular tuple is current in the relation, i.e., when it will appear in a snapshot.

Name	Course	Time	Op
Bill	English	423	Ins
Bill	English	427	Mod
George	English	438	Ins
Bill	English	452	Ins
George	Math	487	Ins
George	Math	495	Del

Figure 8: The Example Relation in Jensen's Data Model

Jensen As in the previous two models, the association of facts with time is invisible in the data model DM/T [Jensen et al. 1991].

As a compensation for the inability to display and directly access timestamped facts, DM/T contains a special system-generated and maintained transaction-time relation, termed a backlog, for each user-defined transaction-time relation. This log-like backlog contains the full, timestamped change history of the associated user-defined relation. Backlog tuples, change requests, are stamped with a single time value and an attribute with values that indicate whether an insertion, deletion, or modification is requested. The timeslice of a backlog is a selection of the portion that existed at the time of the time argument. Thus, the timestamps are present as explicit attributes even after timeslice and may be accessed like any other attribute.

EXAMPLE: Figure 8 illustrates a backlog, timesliced at transaction time 510, for a user-defined transaction-time relation. At transaction time 423, it was recorded that Bill took the Math course. This entry was then "modified," without changing any values at time, 427. □

3.5 Bitemporal Data Models

Bitemporal data models support both valid time and transaction time.

Ben-Zvi The *Time Relational Model* [Ben-Zvi 1982] was the first bitemporal data model. Two types of objects are defined: snapshot relations, as defined in the snapshot model, and bitemporal relations. Bitemporal relations are sets of tuples, with each tuple having five implicit attribute values. The attributes Effective-time-start and Effective-time-stop are the end-points of the interval of validity of the real-world phenomenon being modeled; Registration-time-start is the transaction time of the transaction that stored the Effective-time-start value; Registration-time-stop is the transaction time that stored the Effective-time-stop value; and Deletion-time records the time when erroneously entered tuples are logically deleted. An erroneous attribute value may be corrected by deleting that tuple and inserting a corrected one.

EXAMPLE: The relation instance in Figure 9 is a bitemporal relation in the Time Relational Model over a relation schema with explicit attributes Name and Course. Note that Georeg's enrollment in the Math course has been (logically) deleted. □

Name	Course	Effective time-start	Effective time-stop	Registration time-start	Registration time-stop	Deletion time
Bill	English	1	1	423	427	—
George	English	1	2	438	438	—
Bill	English	3	4	452	452	—
George	Math	5	6	487	487	495

Figure 9: The Example Relation in Ben-Zvi's Data Model

Name	Course	Valid		Transaction	
		Begin	End	Start	Stop
Bill	English	1	∞	423	427
Bill	English	1	1	427	∞
George	English	1	2	438	∞
Bill	English	3	4	452	∞
George	Math	5	6	487	495

Figure 10: The Example Relation in Snodgrass' Data Model

Ahn In differentiating valid and transaction time, a four-dimensional data model was used [Snodgrass & Ahn 1985, Snodgrass & Ahn 1986]. Relational instances were illustrated as a sequence, stamped with individual transaction times, of three-dimensional volumes, where one of the dimensions was valid time (tuples were stamped with intervals).

Snodgrass In the data model associated with TQuel, four implicit attributes were added to each relation: the transaction time of the transaction inserting the tuple, the transaction time of the transaction logically deleting the tuple, the time that the tuple started being valid in reality, and the time that the tuple stopped being valid in reality [Snodgrass 1987, Snodgrass 1993].

EXAMPLE: Figure 10 shows, in the TQuel data model, the bitemporal relation given in Figure 9. □

McKenzie McKenzie's bitemporal model [McKenzie 1988, McKenzie & Snodgrass 1991] timestamps attribute values but retains the requirement that attributes be single valued. This was done in an effort to achieve the benefits of attribute-value timestamping (e.g., the ability to perform a cartesian product) without the implementation complexities of set-valued attributes. The two types of objects in this model are the snapshot and valid-time relations (a transaction-time relation is a sequence of snapshot relations; a bitemporal relation is a sequence of valid-time relations, both indexed by transaction time). The value of an attribute in a valid-time relation is always an ordered pair whose components are a value from the attribute's domain and a set of chronons. There is no requirement that the timestamps of any of the attribute values in a relation be homogeneous, but relations are not allowed to have value-equivalent tuples.

Name	Course
$\langle$ Bill, $\{1, 3, 4\}$ $\rangle$	$\langle$ English, $\{1, 3, 4\}$ $\rangle$
$\langle$ George, $\{1, 2\}$ $\rangle$	$\langle$ English, $\{1, 2\}$ $\rangle$
$\langle$ George, $\{5, 6\}$ $\rangle$	$\langle$ Math, $\{5, 6\}$ $\rangle$

Figure 11: The Example Relation in McKenzie's Data Model

Name		Course	
$[1, \infty] \times [423, 427]$	Bill	$[1, \infty] \times [423, 427]$	English
$[1, 1] \times [423, \text{NOW}]$	Bill	$[1, 1] \times [423, \text{NOW}]$	English
$[3, 4] \times [452, \text{NOW}]$	Bill	$[3, 4] \times [452, \text{NOW}]$	English
$[1, 2] \times [438, \text{NOW}]$	George	$[1, 2] \times [438, \text{NOW}]$	English
$[5, 6] \times [487, 495]$	George	$[5, 6] \times [487, 495]$	Math

Figure 12: The Example Relation in the Gadia-3 Data Model

EXAMPLE: A valid-time relation instance in McKenzie's data model is shown in Figure 11. In this model, Bill's enrollment in English must be recorded in a single tuple, otherwise the value-equivalence requirement is violated. George's enrollment history, however, cannot be recorded in a single tuple; an attribute may be assigned only one value from its value domain. □

Transaction time was supported by indexing a sequence of valid-time states with transaction time [McKenzie & Snodgrass 1990]. This data model also allowed the schema, and even the class of the relation (i.e., snapshot, valid-time, transaction-time, or bitemporal) to vary.

Gadia-3 In the data model associated with the calculus-based query language TempSQL [Gadia 1992], attributes are timestamped with finite unions of rectangles in valid-time/transaction-time space [Bhargava & Gadia 1989B], i.e., effectively bitemporal elements.

EXAMPLE: Figure 12 shows the bitemporal relation given earlier, now as an instance of a relation in the TempSQL data model. □

Thompson In the Accounting Data Model, tuples have, in addition to the natural key, the static attributes, and the time-varying attributes, four timestamp attributes: accounting start time, accounting finish time, engineering start time, engineering finish time, as well as a boolean timewarp attribute [Thompson 1991]. The accounting time roughly corresponds to valid time, and the engineering time corresponds to transaction time (a more detailed comparison may be found elsewhere [Jensen & Snodgrass 1992]). The time warp attribute enables attribute values to change historically.

3.6 Summary

The following brief summary oversimplifies the data models in an effort to differentiate them.

- Brooks was the first to consider time in the database (long before the relational model was proposed!).
- Wiederhold was the first temporal model to be implemented.
- Jones was the first to define a time-oriented algebra.
- Clifford-1 attempted to model the semantics of natural language.
- Ariav exploited the three-dimensional analogue, where the third dimension is valid time.
- Navathe defined his data model primarily to support his extension to SQL called TSQL.
- Sadeghi's data model was defined primarily to support his extension to DEAL called HQL.
- Sarda, Lorentzos and Tansel all incorporated operators to switch between an interval representation and a single chronon representation. Lorentzos' data model, closest to the conventional relational data model, supports nested granularity timestamps and periodic time.
- Segev focussed on scientific data, collected generally at regular intervals by multiple sensors.
- Clifford-2, Gadia-1, Gadia-2, Gadia-3, and Tansel all employ non-1NF data models. Clifford-1 emphasizes associating timestamps with both the attribue value and with the tuple; Clifford-2 associates timestamps with both attributes and with tuples; Gadia-1 emphasizes the homogeneity property; Gadia-2 emphasizes the multi-homogeneous property; and Tansel includes four types of attribute values.
- Kimball was the first implemented transaction-time model.
- Stonebraker has the most impressive implementation to date of a temporal data model.
- Jensen used backlog relations to encode the changes made to transaction-time relations.
- Ben-Zvi was the first to incorporate both transaction time and valid time.
- Ahn demonstrated that transaction time and valid time are entirely orthogonal.
- Snodgrass used a particularly simple bitemporal model to support TQuel.
- McKenzie timestamped attribute values but retains the requirement that attributes have only a single value within a tuple.
- Gadia-3 effectively used bitemporal elements.
- Thompson focused on the use of temporal databases in accounting.

3.7 Comparison

The temporal data models just summarized may be compared by asking four basic questions: how is valid time represented, how is transaction time represented, how are attribute values represented, is the model *homogeneous* and is the model *coalesced*.

	Event	Interval	Valid-time Element
timestamped attribute values		Gadia-2 Lorentzos McKenzie Thompson Tansel	Brooks Clifford-2 Gadia-1 Gadia-3
timestamped tuples	Ariav Clifford-2 Segev	Ahn Ben-Zvi Jones Navathe Sadeghi Sarda Snodgrass Wiederhold	

Table 2: Representation of Valid Time

3.7.1 Valid Time

Two fairly orthogonal aspects are involved in representing valid time. First, is valid time represented with single chronon identifiers (i.e., event timestamps), with intervals (i.e., as interval timestamps), or as valid-time elements (i.e., as a set of chronon identifiers, or equivalently as a finite set of intervals)? Second, is valid time associated with entire tuples or with individual attribute values? A third alternative, associating valid time with sets of tuples, i.e., relations, has not been incorporated into any of the proposed data models, primarily because it lends itself to high data redundancy. The data models are evaluated on these two aspects in Table 2. Interestingly, only one quadrant, timestamping tuples with an valid-time element, has not been considered.

3.7.2 Transaction Time

The same general issues are involved in transaction time, but there are about twice as many alternatives. Transaction time may be associated with

- a single chronon. When stamping a tuple identifying a change to a relation state, the insertion of the tuple signifies the termination (logical deletion) of the most recent tuple (if any) with an identical key value. An additional attribute is required to indicate whether the newly inserted tuple only terminates the previous tuple or also becomes part of the new state (e.g., the attribute Op in Jensen). When an entire evolving states is stamped, no such attribute is necessary. One state is current from its chronon and until it is superceeded by a state with a higher chronon. Note that this alternative results in very high redundancy when compared with the first alternative.

- an interval. A newly inserted tuple would be associated with the interval starting at now and ending at the special value U.C., *until-changed*.

- three chronons. Ben-Zvi's model records (1) the transaction time when the valid start time was recorded, (2) the transaction time when the valid stop time was recorded, and (3) the transaction time when the tuple was logically deleted.

- a transaction-time element, which is a set of not-necessarily-contiguous chronons.

	Single chronon	Interval (pair of chronons)	Three Chronons	Transaction-time element (set of chronons)
timestamped attribute values				Gadia-3
timestamped tuples	Jensen Kimball	Snodgrass Stonebraker	Ben-Zvi	
timestamped sets of tuples	Ahn Thompson	McKenzie		

Table 3: Representation of Transaction Time

Another issue concerns whether transaction time is associated with individual attribute values, with tuples, or with sets of tuples.

The choices made in the various data models are characterized in Table 3. Gadia-3 is the only data model to timestamp attribute values; it is difficult to efficiently implement this alternative directly. Gadia-3 also is the only data model that uses transaction-time elements. Ben-Zvi is the only one to use three transaction-time chronons. All of the rows and columns are represented by at least one data model.

3.7.3 Homogeneity and Coalescing

Table 4 compares the models on the last two aspects. The name of the data model is given in the first column. Whether the model is homogeneous in valid time is indicated in the next column (c.f., Section 3.1.4). All the models are homogeneous in transaction time. Tuple-timestamped data models, to be identified shortly, are necessarily temporally homogeneous. All data models that use single chronons as timestamps turn out to be temporally homogeneous as well. For data models that only support transaction time, this aspect is not relevant.

The next column specifies whether the data model requires that tuples be coalesced in valid time (c.f., Section 3.1.5). No model is coalesced on transaction time. Event-stamped data models are by necessity not valid-time coalesced.

3.7.4 Attribute Value Structure

The final major decision to be made in designing a temporal data model is how to represent attribute values. Six basic alternatives are present in the data models. In some models, the timestamp appears as an explicit attribute; we do not consider such attributes in this analysis.

- *Atomic valued*—values do not have any internal structure.
- *Set valued*—values are sets of atomic values.
- *Functional, atomic valued*—values are functions from the (generally valid) time domain to the attribute domain.
- *Ordered pairs*—values are an ordered pair of a value and a (valid-time element) timestamp.
- *Triplet valued*—values are a triple of attribute values, valid-from time, and valid-to time. This is similar to the ordered pairs representation, except that only one interval may be represented.

Data Model	*Valid-time Homogeneous*	*Valid-time Coalesced*	*Attribute Values*
Ahn	yes	yes	atomic
Ariav	yes	no	atomic
Ben-Zvi	yes	no	atomic
Brooks	no	?	atomic
Clifford-1	yes	no	atomic
Clifford-2	no	no	functional
Gadia-1	yes	no	functional
Gadia-2	no	yes	functional
Gadia-3	yes	no	functional
Jensen	N/A	N/A	atomic
Jones	yes	no	atomic
Kimball	N/A	N/A	atomic
Lorentzos	no	no	atomic
McKenzie	no	yes	ordered pairs
Navathe	yes	yes	atomic
Sadeghi	yes	yes	atomic
Sarda	yes	no	atomic
Segev	yes	no	atomic
Snodgrass	yes	yes	atomic
Stonebraker	N/A	N/A	atomic
Tansel	no	no	atomic, set-valued, triplet, set-triplet
Thompson	yes	no	atomic
Wiederhold	yes	no	atomic, ordered pairs

Table 4: Comparison of Temporal Data Models

- *Set-triplet valued*—values are a set of triplets. This is more general than ordered pairs, in that more than one value can be represented, and more general than functional valued, since more than one attribute value can exist at a single valid time [Tansel 1986].

The last column of Table 4 specifies the attribute value structure associated with each temporal data model.

In the conventional relational model, if attributes are atomic-valued, they are considered to be in *first normal form* [Codd 1972]. Hence, only the data models placed in the first category may be considered to be strictly in first normal form. However, in several of the other models, the non-atomicity of attribute values comes about because time is added.

4 Context

The previously proposed data models arose from several considerations. They were all extensions of the conventional relational model that attempted to capture the time-varying semantics of both the enterprise being modeled and the state of the database. They attempted to retain the simplicity of the relational model; the tuple timestamping models were perhaps most successful in this regard. They attempted to present all the information concerning an object in one tuple; the attribute value timestamped models were perhaps best at that. And they attempted to ensure ease of implementation and query evaluation efficiency; the backlog representation may be advantageous here.

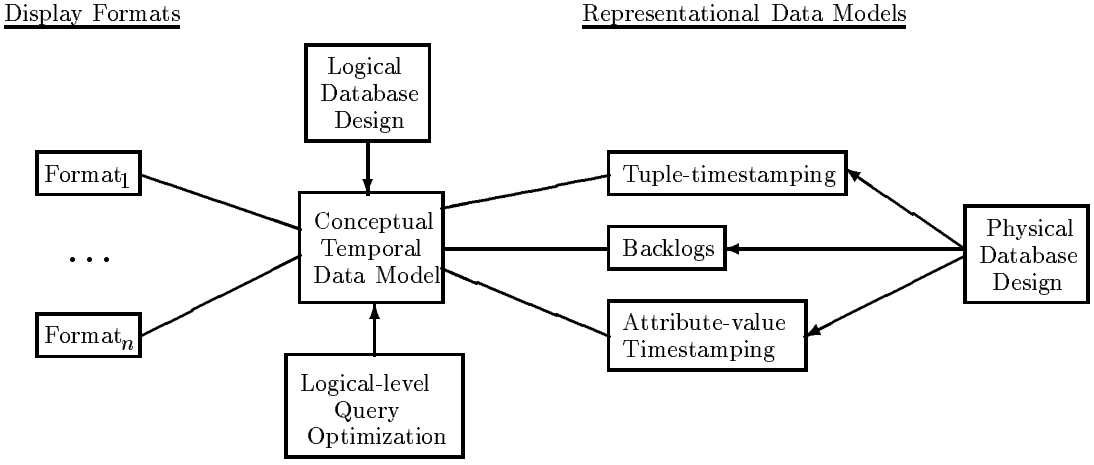

Figure 13: Interaction of Conceptual and Representational Data Models

It is clear from the number of proposed representations that meeting all of these goals simultaneously is a difficult, if not impossible task. It is our contention that focusing on data *presentation* (how temporal data is displayed to the user), on data *storage*, with its requisite demands of regular structure, and on efficient *query evaluation* has complicated the central task of capturing the time-varying semantics of data. The result has been, as we have seen, a plethora of incompatible data models, with many query languages, and a corresponding surfeit of database design and implementation strategies that may be employed across these models.

We therefore advocate a separation of concerns. The time-varying semantics is obscured in the representation schemes by other considerations of presentation and implementation. We feel that the *conceptual* data model to be proposed shortly is the most appropriate basis for expressing this semantics. This data model is generally not the most appropriate way to present the stored data to users, nor is

it the best way to physically store the data. However, there are mappings to several *representational* data models that, in many situations, may be more amenable to presentation and storage, those representations can be employed for those purposes, while retaining the semantics of the conceptual data model. Figure 13 shows the placement of the proposed data model with respect to the tasks of logical and physical database design, storage representation, query optimization, and display. As the figure shows, logical database design produces the conceptual relation schemas, which are then refined into relation schemas in some representational data model(s). Query optimization may be performed on the logical algebra, parameterized by the cost models of the representation(s) chosen for the stored data. Finally, display presentation should be decoupled from the storage representation.

Note that this arrangement hinges on the semantic equivalence of the various data models. It must be possible to map between the conceptual model and the various representational models, as will be discussed in Section 5.4.

5 A New Proposal

We now present a new model, termed the *bitemporal conceptual data model*, or BCDM. This data model supports both valid and transaction time. It is designed to be a conceptual, as opposed to representational data model, in the sense just described.

In the following, we specify the structural aspects of the time domain assumed by the data model, describe its objects (temporal relations), consider how these objects may be updated, and review the relationship with representational models.

5.1 The Time Domain

For both valid and transaction time domains, we assume the linear, discrete, bounded structural model of time. We utilize chronons, as discussed in detail in a separate proposal on timestamp representation [Dyreson & Snodgrass 1992B]. We assume that chronons have length (some multiple or fraction of a "second"). We assume that valid and transaction time are absolute. Relative times may be stored in relations as values of attributes (termed *spans* [Soo & Snodgrass 1992]); such user-defined times are not discussed further here. As we can number the chronons, the domains are isomorphic to the domain of natural numbers.

5.2 Objects in the Model

Tuples in a conceptual bitemporal relation instance are associated with time values from two orthogonal time domains, namely valid time and transaction time. Valid time is used for capturing the time-varying nature of the part of reality being modeled, and transaction time models the update activity of the relation.

In general, the schema of a conceptual bitemporal relation, $\mathcal{R}$, consists of an arbitrary number of explicit attributes, $A_1, A_2, \ldots, A_n$, encoding some fact (possibly composite) and an implicit timestamp attribute, T. Thus, a tuple, $x = (a_1, a_2, \ldots, a_n|t)$, in a conceptual bitemporal relation instance, $r(\mathcal{R})$, consists of a number of attribute values associated with a timestamp value.

An arbitrary subset of the domain of valid times is associated with each tuple, meaning that the fact recorded by the tuple is *true in the modeled reality* during each valid time chronon in the subset. Each individual valid time chronon of a single tuple has associated an arbitrary subset of the domain of transaction times, meaning that the fact, valid during the particular chronon, is *current in the relation* during each of the transaction time chronons in the subset.

Associated with a tuple is a set of so-called *bitemporal chronons* (tiny rectangles) in the two-dimensional space spanned by valid time and transaction time. Such a set is termed a *bitemporal element*[1]. Because no two tuples with mutually identical explicit attribute values (value-equivalent) are allowed in a bitemporal relation instance, the full time history of a fact is contained in a single tuple.

EXAMPLE: Consider a relation recording employee/department information, such as "Jake works for the shipping department." We assume that the granularity of chronons is one day for both valid time and transaction time, and the period of interest is the month of June 1992.

Figure 14 shows how the bitemporal element in an employee's department tuple changes. The x-axis denotes transaction time, and the y-axis denotes valid time. Employee Jake was hired by the company as temporary help in the shipping department for the interval from June 10th to June 15th, and this fact is recorded in the database proactively on June 5th. This is shown in Figure 14(a). The arrows pointing to the right signify that the tuple has not been logically deleted; it continues through to the transaction time NOW. On June 10th, the personnel department discovers an error. Jake had really been hired for the valid time interval from June 5th to June 20th. The database is corrected on June 10th, and the updated bitemporal element is shown in Figure 14(b). On June 15th, the personnel department is informed that the correction was itself incorrect; Jake really was hired for the original time interval, June 10th to June 15th, and the database is corrected the same day. This is shown in Figure 14(c). Lastly, Figure 14(d) shows the result of three updates to the relation, all of which take place on June 20th. While the the period of validity was correct, it was discovered that Jake was not in the shipping department, but in the loading department. Consequently, the fact (Jake, Ship) is removed from the current state and the fact (Jake, Load) is inserted. A new employee, Kate, is hired for the shipping department for the interval from June 25th to June 30th.

We note that the number of bitemporal chronons in a given bitemporal element is the area enclosed by the bitemporal element. The bitemporal element for (Jake, Ship) contains 140 bitemporal chronons.

The example illustrates how transaction time and valid time are handled. As time passes, i.e., as the computer's internal clock advances, the bitemporal elements associated with current facts are updated. For example, when (Jake, Ship) was first inserted, the six valid time chronons from 10 to 15 had associated the transaction time chronon NOW. At time 5, the six new bitemporal chronons, $(5,10),\ldots,(5,15)$, were appended. This continued until time 9, after which the valid time was updated. Thus, starting at time 10, 16 bitemporal chronons are added at every clock tick.

The actual bitemporal relation corresponding to the graphical representation in Figure 14(d) is shown below. This relation contains three facts. The timestamp attribute T shows each transaction time chronon associated with each valid time chronon as a set of ordered pairs.

Emp	Dept	T
Jake	Ship	$\{(5,10),\ldots,(5,15),\ldots,(9,10),\ldots,(9,15),$
		$(10,5),\ldots,(10,20),\ldots,(14,5),\ldots,(14,20),$
		$(15,10),\ldots,(15,15)\ldots,(19,10),\ldots,(19,15)\}$
Jake	Load	$\{(NOW,10),\ldots,(NOW,15)\}$
Kate	Ship	$\{(NOW,25),\ldots,(NOW,30)\}$

□

5.3 Update

We consider the three forms of update, insertion, deletion, and modification, in turn.

[1]This term is a generalization of *temporal element*, used to denotes a set of single dimensional chronons [Gadia 1988].

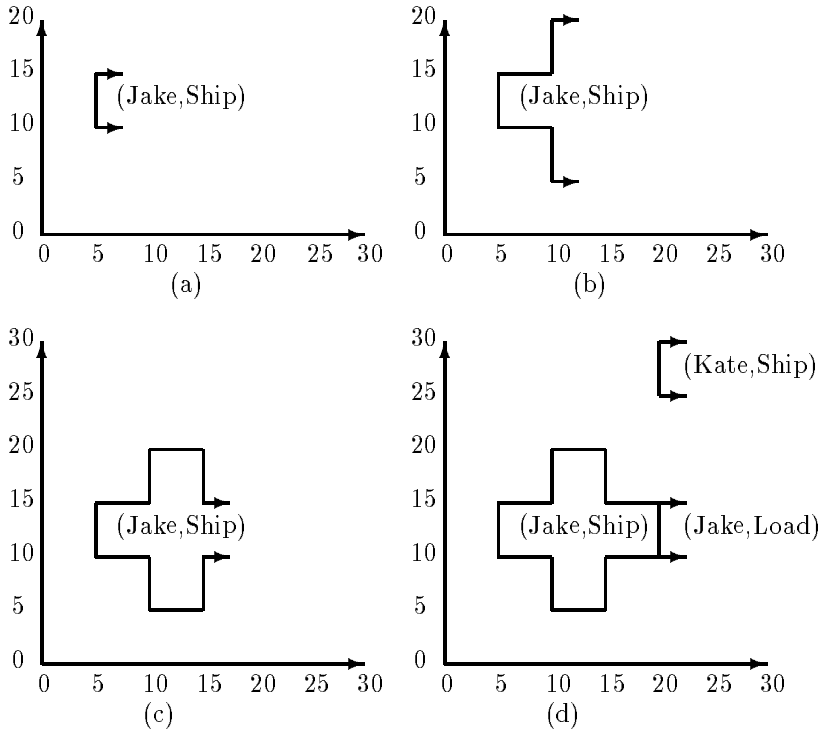

Figure 14: Bitemporal Elements

An insertion is issued when we want to record in a bitemporal relation instance that a currently unrecorded fact $(a_1, \ldots, a_n)$ is true for some period(s) of time. These periods of time are represented by a valid-time element, i.e., a set of valid-time chronons, t_v. When the fact is stored, its valid-time element stamp is transformed into a bitemporal element stamp to capture that, from now on, the fact is current in the relation. We indicate this with a special value in the domain of transaction chronon identifiers, NOW.

The arguments to the `insert` routine are the relation into which a fact is to be inserted, the explicit values of the fact, and the set of valid-time chronons, t_v, during which the fact was true in reality. Insert returns the new, updated version of the relation. There are three cases to consider. First, if $(a_1, \ldots, a_n)$ was never recorded in the relation, a completely new tuple is appended. Second, if $(a_1, \ldots, a_n)$ was part of some previously current state, the tuple recording this is updated with the new valid time information. Third, if $(a_1, \ldots, a_n)$ is already current in the relation, a modification is required, and the insertion is rejected. (In the following, we denote valid-time chronons with c_v and transaction-time chronons with c_t.)

$$\texttt{insert}(r, (a_1, \ldots, a_n), t_v) =$$
$$\begin{cases} r \cup \{(a_1, \ldots, a_n | \{NOW\} \times t_v)\} & \text{if } \neg\exists\, t\ ((a_1, \ldots, a_n | t) \in r) \\ r - \{(a_1, \ldots, a_n | t_b)\} & \\ \quad \cup \{(a_1, \ldots, a_n | t_b \cup \{NOW\} \times t_v)\} & \text{if } \exists\, t_b\ ((a_1, \ldots, a_n | t_b) \in r \ \wedge\ \neg\exists\, (NOW, c_v) \in t_b) \\ r & \text{otherwise} \end{cases}$$

The `insert` routine adds bitemporal chronons with a transaction time of NOW.

As time passes, new chronons must be added. We assume that a special routine `ts_update` is applied to all bitemporal relations at each clock tick. We also assume that the transaction-time granularity

is sufficiently small that only one transaction can execute within a transaction-time chronon. This function simply updates the timestamps to include the new transaction time value. The timestamp of each tuple is examined in turn. When a bitemporal chronon of the type (NOW, c_v) is encountered in the timestamp, a new bitemporal chronon (c_t, c_v), where time c_t is the new transaction-time value, is made part of the timestamp.

```
ts_update(r, c_t) :
    for each x ∈ r
        for each (NOW, c_v) ∈ x[T]
            x[T] ← x[T] ∪ {(c_t, c_v)};
```

Deletion concerns the (logical) removal of a complete tuple from the current valid-time state of the bitemporal relation. We distinguish between the case when there is a tuple to delete and the case when no tuple matching the one to be deleted exists.

$$\texttt{delete}(r, (a_1, \ldots, a_n)) = \begin{cases} r - \{(a_1, \ldots, a_n | t_b)\} \cup \{(a_1, \ldots, a_n, t_b - now\_ts(t_b))\} & \text{if } \exists\, t\, ((a_1, \ldots, a_n | t_b) \in r) \\ r & \text{otherwise} \end{cases}$$

where now_ts is defined as follows.

$$now\_ts(t_b) = \{(NOW, c_v) \mid (NOW, c_v) \in t_b\}$$

Finally, a modification of an existing tuple may be defined by a deletion followed by an insertion as follows.

$$\texttt{modify}(r, (a_1, \ldots, a_n), t_v) = \texttt{delete}(r, (a_1, \ldots | a_n)) \cup \texttt{insert}(r, (a_1, \ldots, a_n), t_v)$$

EXAMPLE: The sequence of bitemporal elements shown in Figure 14 is created by the following sequence of commands, invoked at the indicated transaction time.

Command	*Transaction Time*
`insert(dept,("Jake","Ship"),[6/10,6/15])`	6/5
`modify(dept,("Jake","Ship"),[6/5,6/20])`	6/10
`modify(dept,("Jake","Ship"),[6/10,6/15])`	6/15
`delete(dept,("Jake","Ship"))`	6/20
`insert(dept,("Jake","Load"),[6/10,6/15])`	6/20
`insert(dept,("Kate","Ship"),[6/25,6/30])`	6/20

□

Valid time relations and transaction time relations are special cases of bitemporal relations that support only valid time and transaction time, respectively. Thus an valid-time tuple has associated a set of valid time chronons (termed a *valid-time element* and denoted t_v), and a transaction-time tuple has associated a set of transaction time chronons (termed a *transaction-time element* and denoted t_t). For clarity, we use the term snapshot relation for a conventional relation. Snapshot relations support neither valid time nor transaction time.

5.4 Relationship with Representational Models

It is possible to demonstrate equivalence mappings between the conceptual model and several *representational* models [Jensen et al. 1992B]. Mappings have already been demonstrated for three data models: Gadia-3 (attribute timestamping), Jensen (tuple timestamping with a single transaction chronon), and Snodgrass (tuple timestamping, with interval valid and transaction times). This equivalence is based on *snapshot equivalence*, which says that two relation instances are equivalent if all their snapshots, taken at all times (valid and transaction), are identical. Snapshot equivalence provides a natural means of comparing rather disparate representations. An extension to the conventional relational algebraic operators may be defined in the conceptual data model, and can be mapped to analogous operators in the representational models.

5.5 Logical Design

A confusing array of normal forms for temporal relations, including *First Temporal Normal Form* [Segev and Shoshani 1988], *Time Normal Form* [Navathe & Ahmed 1989], and *P* and *Q Normal Forms* [Lorentzos & Kollias 1989], have been proposed. None of these definitions is truly an extension of conventional normal forms, for a variety of reasons that we detail elsewhere [Jensen et al. 1992C]. Also, each definition is restricted to a specific data model, and inherits the peculiarities inherent in that model. It is not satisfactory to have to define all the normal forms anew for each of the two dozen existing temporal data models.

Elsewhere we present a consistent framework of temporal equivalents of all the important conventional database design concepts: functional and multivalued dependencies, primary keys, and third, Boyce-Codd, and fourth normal forms [Jensen et al. 1992C]. This framework is enabled by making a clear distinction between the logical concept of a temporal relation and its physical representation. As a result, the role played by temporal normal forms during temporal database design in the BCDM closely parallels that of normal forms during conventional database design.

5.6 Evaluation

Here we evaluate the bitemporal conceptual data model along the same aspects by which existing temporal data models were compared in Section 3.7.

The BCDM timestamps tuples, as does Ben-Zvi, Clifford-1, Jones, Navathe, Sadeghi, Sarda, Segev and Snodgrass. The timestamps are temporal elements, as Clifford-2, Gadia-1 and Gadia-3. In Table 2, the BCDM occupies the unfilled entry corresponding to timestamping tuples with valid-time elements. In Table 3, the BCDM occupies the unfilled entry corresponding to timestamping tuples with transaction-time elements. Hence, the timestamping tuples with bitemporal elements is unlike any other existing data model. The BCDM is inherently valid-time homogeneous—about half of the temporal data models are homogeneous. The BCDM is also inherently valid-time coalesced; Ahn, Gadia-2, McKenzie, Navathe, Sadeghi and Snodgrass are coalesced. Attributes are atomic in the BCDM, as in most of the temporal data models proposed to date.

6 Conclusion

This paper has compared the many existing temporal data models and has proposed a new one, the bitemporal conceptual data model (BCDM), as the basis for the Temporal Structured Query Language (TSQL).

The notion of temporal relation proposed in this paper is a conceptual one. It is based on the observation that different data models are appropriate for data presentation, storage representation, and the modeling of the time semantics of data. These separate tasks pose very different requirements for a data model, and they should be considered in isolation, utilizing different data models. It is our contention that the large number of data models existing today is a consequence of trying to do each of the tasks using the same data model. As a result of trying to accommodate presentation and representation, the central task of modeling time semantics of data has been obscured by data models.

The type of temporal relation proposed in this paper is built on the experiences from older data models and is a very simple one. A temporal relation consists simply of a set of ordinary tuples. For each tuple, an implicit attribute value specifies when the (composite) fact represented by the tuple is true in the modeled reality and is current in the stored relation. The implicit attribute has temporal elements (i.e., sets of chronons) as its values. Temporal elements have been chosen because they allow relations to contain one complete fact per tuple—a relation is a *set* of tuples, and value-equivalence and coalescing are easily ensured. Also temporal elements are generalizations of events and intervals and are closed under union, difference, and complementation.

An important property of the conceptual model—shared with the conventional relational model, but not held by the representational models—is that relation instances are semantically unique; distinct instances model different realities and thus have distinct semantics.

That two temporal relations have the same information content was formalized as the notion of the temporal relations being snapshot equivalent. Briefly, two relations are snapshot equivalent if all derived snapshots are mutually identical. As all data models provide means of producing snapshots, snapshot equivalence provides a natural way to compare temporal relations in diverse data models. The semantic uniqueness of the relations in the BCDM implies that two snapshot equivalent temporal relations are identical (and reversely).

Finally, we feel that the BCDM is the appropriate location for database design and logical-level query optimization.

This paper has addressed one aspect of the design of the Temporal Structured Query Language. The closely related tasks of designing an appropriate query language and algebra are under way.

7 Acknowledgements

This work was supported in part by NSF grant ISI-8902707. This work was done while the first author (a faculty member at the Department of Mathematics and Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark) visited the University of Arizona. James Clifford was helpful in understanding structural aspects of models of time. Curtis Dyreson, Nick Kline and Michael Soo provided useful comments on a previous draft.

Bibliography

[Allen & Hayes 1985] Allen, J.F. and P.J. Hayes. *A Common-Sense Theory of Time*, in *Proceedings of the International Joint Conference on Artificial Intelligence*. Los Angeles, CA: Aug. 1985, pp. 528–531.

[Anderson 1982] Anderson, T.L. *Modeling Time at the Conceptual Level*, in *Proceedings of the International Conference on Databases: Improving Usability and Responsiveness*. Ed. P. Scheuermann.

Jerusalem, Israel: Academic Press, June 1982, pp. 273–297.

[Ariav 1986] Ariav, G. *A Temporally Oriented Data Model. ACM Transactions on Database Systems*, 11, No. 4, Dec. 1986, pp. 499–527.

[Ben-Zvi 1982] Ben-Zvi, J. *The Time Relational Model.* PhD. Diss. Computer Science Department, UCLA, 1982.

[Bhargava & Gadia 1989A] Bhargava, G. and S. Gadia. *Achieving Zero Information Loss in a Classical Database Environment*, in *Proceedings of the Conference on Very Large Databases.* Amsterdam: Aug. 1989, pp. 217–224.

[Bhargava & Gadia 1989B] Bhargava, G. and S.K. Gadia. *A 2-dimensional temporal relational database model for querying errors and updates, and for achieving zero information-loss.* Technical Report TR#89-24. Department of Computer Science, Iowa State University. Dec. 1989.

[Bhargava & Gadia 1990] Bhargava, G. and S.K. Gadia. *The Concept of Error in a Database: An Application of Temporal Databases*, in *Proceedings of 1990 COMAD International Conference on Management of Data.* Ed. N. Prakash. New Delhi: Tata McGraw-Hill, Dec. 1990, pp. 106—121.

[Bhargava & Gadia 1991] Bhargava, G. and S.K. Gadia. *Relational database systems with zero information-loss. IEEE Transactions on Knowledge and Data Engineering*, (to appear) (1991).

[Blum 1981] Blum, R.L. *Displaying Clinical Data from a Time-Oriented Database. Comput. Biol. Med.*, 11, No. 4 (1981), pp. 197–210.

[Bolour et al. 1982] Bolour, A., T.L. Anderson, L.J. Dekeyser and H.K.T. Wong. *The Role of Time in Information Processing: A Survey. SigArt Newsletter*, 80, Apr. 1982, pp. 28–48.

[Brooks 1956] Brooks, F.P. *The Analytic Design of Automatic Data Processing Systems.* PhD. Diss. Harvard University, May 1956.

[Chomicki & Imelinski 1989] Chomicki, J. and T. Imelinski. *Relational Specifications of Infinite Query Answers*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* May 1989, pp. 174–183.

[Clifford 1982] Clifford, J. *A Model for Historical Databases*, in *Proceedings of Workshop on Logical Bases for Data Bases.* Toulouse, France: Dec. 1982.

[Clifford & Warren 1983] Clifford, J. and D.S. Warren. *Formal Semantics for Time in Databases. ACM Transactions on Database Systems*, 8, No. 2, June 1983, pp. 214–254.

[Clifford & Tansel 1985] Clifford, J. and A.U. Tansel. *On an Algebra for Historical Relational Databases: Two Views*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247–265.

[Clifford & Croker 1987] Clifford, J. and A. Croker. *The Historical Relational Data Model (HRDM) and*

Algebra Based on Lifespans, in *Proceedings of the International Conference on Data Engineering.* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1987, pp. 528–537.

[Clifford & Rao 1987] Clifford, J. and A. Rao. *A Simple, General Structure for Temporal Domains*, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* AFCET. France: May 1987, pp. 23–30.

[Codd 1972] Codd, E. F. *Further Normalization of the Data Base Relational Model*, in Data Base Systems. Vol. 6 of Courant Computer Symposia Series. Englewood Cliffs, N.J.: Prentice hall, 1972.

[Deen 1985] Deen, S.M. *DEAL: A Relational Language with Deductions, Functions and Recursions. Data and Knowledge Engineering*, 1 (1985).

[Dyreson & Snodgrass 1992A] Dyreson, C. E. and R. T. Snodgrass. *Historical Indeterminacy.* Technical Report TR 92-16a. Computer Science Department, University of Arizona. July 1992.

[Dyreson & Snodgrass 1992B] Dyreson, C. E. and R. T. Snodgrass. *Time-stamp Semantics and Representation.* TempIS Technical Report 33. Computer Science Department, University of Arizona. Feb. 1992.

[Gadia & Vaishnav 1985] Gadia, S.K. and J.H. Vaishnav. *A Query Language for a Homogeneous Temporal Database*, in *Proceedings of the ACM Symposium on Principles of Database Systems.* Mar. 1985, pp. 51–56.

[Gadia 1986] Gadia, S.K. *Toward a Multihomogeneous Model for a Temporal Database*, in *Proceedings of the International Conference on Data Engineering.* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 390–397.

[Gadia 1988] Gadia, S.K. *A Homogeneous Relational Model and Query Languages for Temporal Databases. ACM Transactions on Database Systems*, 13, No. 4, Dec. 1988, pp. 418–448.

[Gadia & Yeung 1988] Gadia, S.K. and C.S. Yeung. *A Generalized Model for a Relational Temporal Database*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery. Chicago, IL: June 1988, pp. 251–259.

[Gadia 1992] Gadia, S.K. *A Seamless Generic Extension of SQL for Querying Temporal Data.* Technical Report TR-92-02. Computer Science Department, Iowa State University. May 1992.

[Hall et al. 1976] Hall, P., J. Owlett and S. J. P. Todd. *Relations and Entities*, in *Modelling in Data Base Management Systems.* Ed. G. M. Nijssen. North-Holland, 1976. pp. 201–220.

[Hawking 1988] Hawking, S. *A Brief History of Time.* New York: Bantam Books, 1988.

[Jensen et al. 1991] Jensen, C. S., L. Mark and N. Roussopoulos. *Incremental Implementation Model for Relational Databases with Transaction Time. IEEE Transactions on Knowledge and Data Engineering*, 3, No. 4, Dec. 1991, pp. 461–473.

[Jensen & Snodgrass 1992] Jensen, C. S. and R. Snodgrass. *Temporal Specialization and Generalization, to appear. IEEE Transactions on Knowledge and Data Engineering*, (1992).

[Jensen et al. 1992B] Jensen, C. S., M. D. Soo and R. T. Snodgrass. *Unification of Temporal Relations.* Technical Report 92-15. Computer Science Department, University of Arizona. July 1992.

[Jensen et al. 1992C] Jensen, C. S., R. T. Snodgrass and M. D. Soo. *Extending Normal Forms to Temporal Relations.* TR 92-17. Computer Science Department, University of Arizona. July 1992.

[Jensen et al. 1992A] Jensen, C.S., J. Clifford, S.K. Gadia, A. Segev and R.T. Snodgrass. *A Glossary of Temporal Database Concepts. SIGMOD Record*, 21, No. 3, Sep. 1992.

[Jones et al. 1979] Jones, S., P. Mason and R. Stamper. *LEGOL 2.0: A Relational Specification Language for Complex Rules. Information Systems*, 4, No. 4, Nov. 1979, pp. 293–305.

[Kimball 1978] Kimball, K.A. *The DATA System.* Master's Thesis, University of Pennsylvania, 1978.

[Lorentzos & Johnson 1988] Lorentzos, N. and R. Johnson. *Extending Relational Algebra to Manipulate Temporal Data. Information Systems*, 13, No. 3 (1988), pp. 289–296.

[Lorentzos & Kollias 1989] Lorentzos, N. and V. Kollias. *The Handling of Depth and Time Intervals in Soil-Information Systems. Computers and Geosciences*, 15, No. 3 (1989), pp. 395–401.

[Lorentzos 1988] Lorentzos, N.A. *A formal extension of the relational model for the representation and manipulation of generic intervals.* PhD. Diss. Birkbeck College, University of London, 1988.

[McKenzie 1986] McKenzie, E. *Bibliography: Temporal Databases. ACM SIGMOD Record*, 15, No. 4, Dec. 1986, pp. 40–52.

[McKenzie 1988] McKenzie, E. *An Algebraic Language for Query and Update of Temporal Databases.* PhD. Diss. Computer Science Department, University of North Carolina at Chapel Hill, Sep. 1988.

[McKenzie & Snodgrass 1990] McKenzie, E. and R. Snodgrass. *Schema Evolution and the Relational Algebra. Information Systems*, 15, No. 2, June 1990, pp. 207–232.

[McKenzie & Snodgrass 1991] McKenzie, E. 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. Aug. 1991.

[Navathe & Ahmed 1987] Navathe, S. B. and R. Ahmed. *TSQL-A Language Interface for History Databases*, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* AFCET. France: May 1987, pp. 113–128.

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

[Rowe & Stonebraker 1987] Rowe, L. and M. Stonebraker. *The POSTGRES Papers.* Technical Report UCB/ERL M86/85. University of California. June 1987.

[Sadeghi 1987] Sadeghi, R. *A Database Query Language for Operations on Historical Data.* PhD. Diss. Dundee College of Technology, Dec. 1987.

[Sadeghi et al. 1987] Sadeghi, R., W.B. Samson and S.M. Deen. *HQL — A Historical Query Language.* Technical Report. Dundee College of Technology. Sep. 1987.

[Sarda 1990A] Sarda, N. *Extensions to SQL for Historical Databases. IEEE Transactions on Knowledge and Data Engineering*, 2, No. 2, June 1990, pp. 220–230.

[Sarda 1990B] Sarda, N. *Algebra and Query Language for a Historical Data Model. The Computer Journal*, 33, No. 1, Feb. 1990, pp. 11–18.

[Segev & Shoshani 1987] Segev, A. and A. Shoshani. *Logical Modeling of Temporal Data*, in *Proceedings of the ACM SIGMOD Annual Conference on Management of Data.* Ed. U. Dayal and I. Traiger. Association for Computing Machinery. San Francisco, CA: ACM Press, May 1987, pp. 454–466.

[Segev and Shoshani 1988] Segev, A. and A. Shoshani. *The Representation of a Temporal Data Model in the Relational Environment*, in *Proceeding of the 4th International Conference on Statistical and Scientific Database Management.* 1988.

[Snodgrass & Ahn 1985] Snodgrass, R. and I. Ahn. *A Taxonomy of Time in Databases*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 236–246.

[Snodgrass & Ahn 1986] Snodgrass, R. and I. Ahn. *Temporal Databases. IEEE Computer*, 19, No. 9, Sep. 1986, pp. 35–42.

[Snodgrass 1987] Snodgrass, R. T. *The Temporal Query Language TQuel. ACM Transactions on Database Systems*, 12, No. 2, June 1987, pp. 247–298.

[Snodgrass 1992] Snodgrass, R.T. *Temporal Databases*, in *Proceedings of the International Conference on GIS: From Space to Territory.* Pisa: Springer-Verlag, Sep. 1992.

[Snodgrass 1993] Snodgrass, R.T. *An Overview of TQuel*, in Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings Pub. Co., 1993. Chap. 6.

[Soo & Snodgrass 1992] Soo, M. and R. Snodgrass. *Mixed Calendar Query Language Support for Temporal Constants.* TempIS Technical Report 29. Computer Science Department, University of Arizona. Revised May 1992.

[Soo 1991] Soo, M. D. *Bibliography on Temporal Databases. ACM SIGMOD Record*, 20, No. 1, Mar. 1991, pp. 14–23.

[Stam & Snodgrass 1988] Stam, R. and R. Snodgrass. *A Bibliography on Temporal Databases. Database Engineering*, 7, No. 4, Dec. 1988, pp. 231–239.

[Stonebraker 1987] Stonebraker, M. *The Design of the POSTGRES Storage System*, in *Proceedings of the Conference on Very Large Databases.* Ed. P. Hammersley. Brighton, England: Sep. 1987, pp. 289–300.

[Tansel 1986] Tansel, A.U. *Adding Time Dimension to Relational Model and Extending Relational Algebra. Information Systems*, 11, No. 4 (1986), pp. 343–355.

[Tansel & Arkun 1986A] Tansel, A.U. and M.E. Arkun. *HQuel, A Query Language for Historical Relational Databases*, in *Proceedings of the Third International Workshop on Statistical and Scientific Databases.* July 1986.

[Tansel & Arkun 1986B] Tansel, A.U. and M.E. Arkun. *HQUEL, A Query Language for Historical Relational Databases*, in *Proceedings of the Third International Workshop on Statistical and Scientific Databases.* July 1986.

[Tansel et al. 1989] Tansel, A.U., M.E. Arkun and G. Özsoyoğlu. *Time-By-Example Query Language for Historical Databases. IEEE Transactions on Software Engineering*, 15, No. 4, Apr. 1989, pp. 464–478.

[Thompson 1991] Thompson, P.M. *A Temporal Data Model Based on Accounting Principles.* PhD. Diss. Department of Computer Science, University of Calgary, Mar. 1991.

[Tuzhilin and Clifford 1990] Tuzhilin, A. and J. Clifford. *A Temporal Relational Algebra as a Basis for Temporal Relational Completeness*, in *Proceedings of the Conference on Very Large Databases.* Brisbane, Australia: Aug. 1990.

[Van Benthem 1982] Van Benthem, J.F.K.A. *The Logic of Time.* Reidel, 1982.

[Wiederhold et al. 1975] Wiederhold, G., J.F. Fries and S. Weyl. *Structured Organization of Clinical Data Bases*, in *Proceedings of the AFIPS National Computer Conference.* AFIPS. 1975, pp. 479–485.

[Worboys 1990] Worboys, M.F. *Reasoning About GIS Using Temporal and Dynamic Logics.* 1990. (Unpublished paper.)

[Yeung 1986] Yeung, C.S. *Query Languages for a Heterogeneous Temporal Database.* Master's Thesis, EE/CS Department, Texas Tech University, 1986.