12

The TSQL2 Data Model

Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo

1 Introduction

Adding time to the relational model has been a daunting task [1, 2, 3, 4]. More than two dozen time-extended relational data models have been proposed over the last fifteen years [5]. 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 [6, 7, 8, 9, 10]; such models are termed *bitemporal*. As a whole, these data models are referred to as *temporal* data models [11].

This chapter introduces the data model upon which TSQL2 is based. 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 chapter we focus on the objects, temporal relations. Subsequent chapters 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 is 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 [8]) 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 [12], 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 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, it is possible to use the existing data models. Specifically, the notion of *snapshot equivalence* may be used to demonstrate equivalence mappings between the conceptual model and several *representational* models [13]. 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, in the next section, we start 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, and subsequent comparison, of 23 existing temporal data models provides context for presenting the TSQL2 data model. The bitemporal conceptual relation is presented in Section 5. We summarize the chapter in Section 6.

2 Dimensions of Time

In this section, we focus on the various dimensions of time. In the next section, we show how previous proposals have combined time with facts to model time-varying information.

Time is multi-dimensional [14]. *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 [11]).

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.

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* [15], 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. In this section, we discuss the association of facts with times. Specifically, we survey 23 existing data models that have been proposed over the last fifteen years. We consider each model in turn, starting with valid-time models, continuing with transaction-time models, and ending with bitemporal models. As a foundation, 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.

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 [11]. (We use consensus terminology in this chapter. The TSQL2 equivalents for events and intervals, used in the language definition are *datetimes* and *periods*, respectively. "Temporal element" is defined identically by TSQL2 and the consensus glossary.)

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.

Attribute Variability

Attributes are commonly categorized based on how they interact with time. A *time-invariant* attribute [16] 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. 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 represent by means of surrogates which are system-generated, unique identifiers that can be referenced and compared for equality, but not displayed to the user [17]. 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* [14]; 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.

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 need 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.

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 [12]. A temporal relation is said to be temporally homogeneous if its tuples are temporally homogeneous [11]. 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 snapshot 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.

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 generally 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 [18], which was a precursor to his heterogeneous model (Gadia-2), and Gadia's two-dimensional temporal relational database model [19], which is a

Data Model	*Citation*	*Time Dimension(s)*	*Identifier*
—	[14]	both	Ahn
Temporally Oriented Data Model	[27]	valid	Ariav
Time Relational Model	[6]	both	Ben-Zvi
—	[21]	valid	Brooks
Historical Data Model	[26]	valid	Clifford-1
Historical Relational Data Model	[35]	valid	Clifford-2
Homogeneous Relational Model	[12]	valid	Gadia-1
Heterogeneous Relational Model	[41]	valid	Gadia-2
TempSQL	[53]	both	Gadia-3
DM/T	[47]	transaction	Jensen
LEGOL 2.0	[24]	valid	Jones
DATA	[45]	transaction	Kimball
Temporal Relational Model	[43]	valid	Lorentzos
—	[50]	both	McKenzie
Temporal Relational Model	[16]	valid	Navathe
HQL	[29]	valid	Sadeghi
HSQL	[32]	valid	Sarda
Temporal Data Model	[34]	valid	Segev
TQuel	[8]	both	Snodgrass
Postgres	[46]	transaction	Stonebraker
HQuel	[37]	valid	Tansel
Accounting Data Model	[10]	both	Thompson
Time Oriented Data Base Model	[22]	valid	Wiederhold

Table 1: Temporal data models

precursor to Gadia-3. We also do not include the data model used as the basis for defining temporal relational completeness [20] 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 conclude 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 [21]. 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 [22] 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 [23]. 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.

Example 1 For the patient whose record is shown in Figure 1 [23], 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 [24] 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

P327
↓
Name ⟶ “John Smith”
↓
Dates ⟶ DATE_ARRAY
↓
Temperatures ⟶ (1, 37.1) ⟶ (2, 37.3) ⟶ (3, 37.0)
↓
Hepatitis ⟶ (Intervals (3, 17), (21, 26)) ⟶ (Interval-Values (850, 1235))

Figure 1: A time-oriented record for a hypothetical patient

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

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 2 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 [25, 26].

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* [27]. Associated with this data model is a calculus-based query language, TOSQL.

Navathe The Temporal Relational Model [28] and its associated algebra were defined primarily to support TSQL [16], 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 [29] is similar in many ways to Navathe's. It was designed to support the calculus-based valid-time query language HQL [30], which in turn is based on DEAL [31]. 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 [33] were designed to support the calculus-based query language HSQL [32]. 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 3 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

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

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.

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 [34]. 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* [35], a refinement of the model associated with a valid-time algebra [36], 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 4 Figure 4 illustrates the valid-time relation instance in the Historical

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

Figure 4: The example relation in the Clifford-2 data model

Relational Data Model, where {Name → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, Course → {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 [36, 37] was designed to support the calculus-based query language HQuel [38] and, later, the Time-by-Example language [52]. 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 5 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.

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

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

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 [12] allows two types of objects: valid-time elements [39] 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.

Example 6 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 [40] 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 [18] and Yeung's heterogeneous models [41, 42] 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 [41] 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* [43, 44] 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 7 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 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

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

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 [45], 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 [46] 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.

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

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.

Jensen As in the previous two models, the association of facts with time is invisible in the data model DM/T [47].

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 8 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* [6] 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

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

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 9 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 George's enrollment in the Math course has been (logically) deleted. □

Ahn In differentiating valid and transaction time, a four-dimensional data model was used [48, 14]. 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 [8, 9].

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

McKenzie McKenzie's bitemporal model [49, 50] 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

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

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

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.

Example 11 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 [51]. 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 [53], attributes are timestamped with finite unions of rectangles in valid-

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

Figure 12: The example relation in the Gadia-3 data model

time/transaction-time space [19], i.e., effectively bitemporal elements.

Example 12 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 time warp attribute [10]. The accounting time roughly corresponds to valid time, and the engineering time corresponds to transaction time (a more detailed comparison may be found elsewhere [54]). 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 attribute 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*.

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,

	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

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.

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 state is stamped, no such attribute is necessary. One state is current from its chronon and until it is superseded 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 *UC*, *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

	Single chronon	Interval (pair of chronons)	Three Chronons	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

deleted.

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

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.

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). 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). No model is coalesced on transaction time. Event-stamped data models are by necessity not valid-time coalesced.

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

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.
- *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 [37].

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* [55]. 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 either the reality being modeled, the state of the database, or both. 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.

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

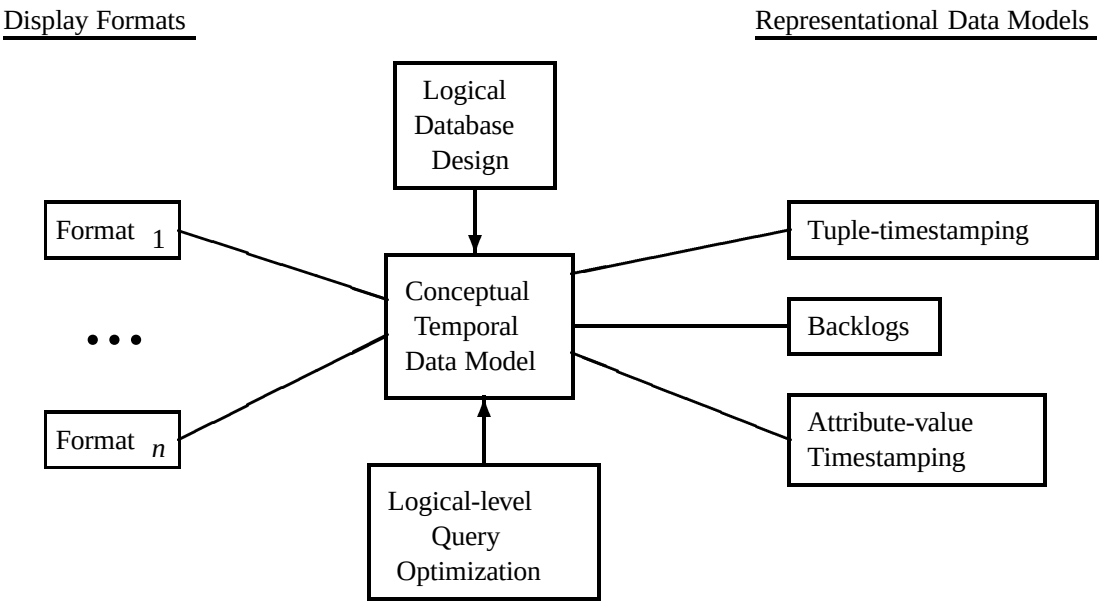

Figure 13: Interaction of conceptual and representational data models

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 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 discussed 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 TSQL2's 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 Chapter 6, Section 3.

5 A New Data Model

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 data model, as opposed to a representational data model, in the sense just described.

We begin by specifying the structural aspects of the time domain assumed by the data model. In Section 5.2, we describe the objects (temporal relations) of the model and consider how these objects may be updated.

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 [56, Chapter 25] on timestamp representation [57]. 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 intrerval-typed attributes; 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 bitemporal conceptual relation instance are associated with time values from both valid time and transaction time. For both domains, we assume that the database system has limited precision; the smallest time unit is termed a *chronon* [11]. The time domains have total orders and both are isomorphic to subsets of the domain of natural numbers. The domain of valid times may be given as $D_{VT} = \{t_1, t_2, \ldots, t_k\}$ and the domain of transaction times may be given as $D_{TT} = \{t'_1, t'_2, \ldots, t'_j\} \cup \{UC\}$ where UC is a distinguished value which is used during update as will be explained later in this section. We expect that the valid time domain is chosen so that some times are before the current time and some times are after the current time.

We also define a set of attribute names $\mathcal{D}_A = \{A_1, A_2, \ldots, A_{n_A}\}$ and a set of attribute domains $\mathcal{D}_D = \{D_1, D_2, \ldots, D_{n_D}\}$. In general, the schema of a bitemporal conceptual relation, $\mathcal{R}$, consists of an arbitrary number of explicit attributes from $\mathcal{D}_A$, A_1, A_2, $\ldots$, A_n, with domains in $\mathcal{D}_D$, encoding some fact (possibly composite) and an implicit timestamp attribute, T, with domain $D_{TT} \times D_{VT}$. Thus, a tuple, $x = (a_1, a_2, \ldots, a_n | \ t_b)$, in a bitemporal conceptual 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 a 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. Any subset of transaction times less than the current time and including the value *UC* may be associated with a valid time. Notice that while the definition of a bitemporal chronon is symmetric, the explanation is asymmetric. This asymmetry is also present in the the update operations to be defined shortly, and it reflects the different semantics of transaction and valid time.

Thus, associated with a tuple is a bitemporal element, denoted t_b, consisting of bitemporal chronons ("tiny rectangles") in the two-dimensional space spanned by valid time and transaction time. Because no two tuples with mutually identical explicit attribute values (termed *value-equivalent*) are allowed in a bitemporal relation instance, the full time history of a fact is contained in a single tuple.

In graphical representations of bitemporal space, we choose the x-axis as the transaction-time dimension, and the y-axis as the valid-time dimension. Hence, the ordered pair (t, v) represents the bitemporal chronon with transaction time t and valid time v.

Example 13 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 some given month in a given year, e.g., June 1992. Throughout, we use integers as timestamp components. The reader may informally think of these integers as dates, e.g., the integer 15 in a timestamp represents the date June 15th.

Figure 14 shows how the bitemporal element in an employee's department tuple changes. Employee Jake was hired by the company as temporary help in the shipping department for the interval from time 10 to time 15, and this fact became current in the database at time 5. 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 *UC*(*until_changed*).

Figure 14(b) shows a correction. The personnel department discovers that Jake had really been hired from time 5 to time 20, and the database is corrected beginning at time 10. Later, the personnel department is informed that the correction was itself incorrect; Jake really was hired for the original time interval, time 10 to time 15, and the correction took effect in the database at time 15. This is shown in Figure 14(c). Lastly, Figure 14(d) shows the result of three updates to the relation, all of which become current starting at time 20. These three updates could have been entered in a single transaction, or as separate transactions occurring during the

same chronon. While 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 time 25 to time 30.

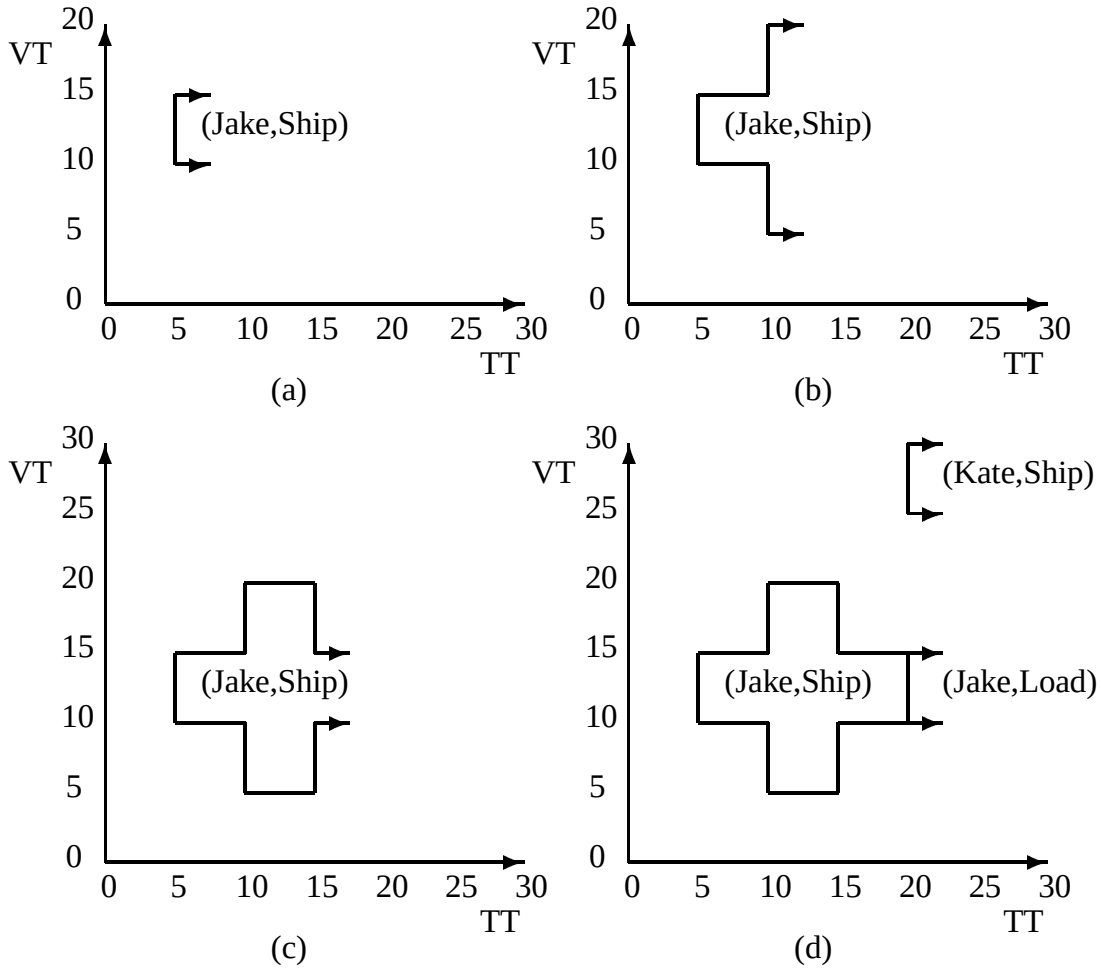

Figure 14: Bitemporal elements

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, consider when the fact (Jake, Ship) was first inserted into the database. Due to the semantics of insertion as described in the next section, facts are inserted to the relation during the chronon prior to when they first become current. Thus (Jake, Ship) is physically inserted into the relation at time 4, with six valid time chronons (10 to 15) each with the associated transaction time chronon *UC*.

At this time, the fact is not yet current in the database since no bitemporal chronons with a transaction time other than *UC* are associated with the tuple. At time 5, the fact logically becomes current in the database, and the six new bitemporal chronons, (5, 10), . . . , (5, 15), are appended. This continues until time 9, when a correction to the fact's valid time is made. 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 in Figure 15. 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), . . . , (5, 15), . . . , (9, 10), . . . , (9, 15), (10, 5), . . . , (10, 20), . . . , (14, 5), . . . , (14, 20), (15, 10), . . . , (15, 15) . . . , (19, 10), . . . , (19, 15)}
Jake	Load	{(*UC*, 10), . . . , (*UC*, 15)}
Kate	Ship	{(*UC*, 25), . . . , (*UC*, 30)}

Figure 15: Bitemporal relation instance

Valid-time relations and transaction-time relations are special cases of bitemporal relations that support only valid time or transaction time, respectively. Thus a 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.

5.3 Logical Design

A confusing array of normal forms for temporal relations, including *First Temporal Normal Form* [58], *Time Normal Form* [16], and *P* and *Q Normal Forms* [59], have been proposed. None of these definitions is truly an extension of conventional normal forms, for a variety of reasons that we detail elsewhere [60]. 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 [60]. This framework is enabled by making a clear distinction between the logical con-

cept 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.

5.4 Evaluation

We briefly evaluate the bitemporal conceptual data model using the same criteria 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 in 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 BCDM is unique in that it timestamps tuples with bitemporal elements. 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 Summary

This chapter has compared the many existing temporal data models and has discussed a new one, the bitemporal conceptual data model (BCDM), as the basis for TSQL2.

The data model discussed in this chapter is a conceptual one, meant specifically for the purpose of capturing the semantics of time-varying data. It is based on the observation that different data models are appropriate for different tasks, such as data presentation, storage representation, and modeling 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 the time semantics of data has been obscured by data models. Finally, we feel that the BCDM is the appropriate location for database design and logical-level query optimization.

The BCDM is a simple data model, built on the experience gained from previous proposals. A BCDM 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 re-

lation. 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.

We have shown elsewhere [13] that the BCDM is a unifying model in that conceptual instances can be mapped into instances of five existing bitemporal *representational data models*. We have also shown how extensions to the conventional relational algebraic operators can be defined in a representational data model and then be meaningfully mapped to analogous operators in the conceptual data model. That two temporal relations have the same information content was formalized using the notion of snapshot equivalence. The semantic uniqueness of the relations in the BCDM implies that two BCDM relations are snapshot equivalent if and only if they are identical.

References

[1] 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.

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

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

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

[5] Snodgrass, R. T. "Temporal Databases," in A. U. Frank, I. Campari, and U. Formentini (eds.), Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Vol. 639 of Lecture Notes in Computer Science. Springer-Verlag, 1992. pp. 22–64.

[6] Ben-Zvi, J. "The Time Relational Model." PhD. Dissertation. Computer Science Department, UCLA, 1982.

[7] 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.

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

[9] Snodgrass, R. T., S. Gomez and E. McKenzie. "Aggregates in the Temporal Query Language TQuel." *IEEE Transactions on Knowledge and Data Engineering*, 5, Oct. 1993, pp. 826–842.

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

[11] Jensen, C. S., J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes and S. Jajodia [eds]. "A Glossary of Temporal Database Concepts." *ACM SIGMOD Record*, 23, No. 1, Mar. 1994, pp. 52–64.

[12] 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.

[13] 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.

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

[15] Jensen, C. S. and R. Snodgrass. "Temporal Specialization," in *Proceedings of the International Conference on Data Engineering.* Ed. F. Golshani. IEEE. Tempe, AZ: Feb. 1992, pp. 594–603.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

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

[26] 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.

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

[28] 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.

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

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

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

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

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

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] Bhargava, G. and S. K. Gadia. "Relational Database Systems with Zero Information-loss." *IEEE Transactions on Knowledge and Data Engineering*, (to appear) (1991).

[41] 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.

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

[43] Lorentzos, N. A. "A Formal Extension of the Relational Model for the Representation of Generic Intervals." PhD. Dissertation. Birkbeck College, 1988.

[44] Lorentzos, N. A. and R. G. Johnson. "Requirements Specification for a Temporal Extension to the Relational Model." *Data Engineering*, 11, No. 4 (1988), pp. 26–33.

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

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

[47] 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.

[48] Snodgrass, R. T. 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.

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

[50] McKenzie, E. L. and R. T. 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.

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

[52] 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.

[53] 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.

[54] Jensen, C. S. and R. Snodgrass. "Temporal Specialization and Generalization." *IEEE Transactions on Knowledge and Data Engineering*, 6, No. 6 (1994), pp. 954–974.

[55] 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.

[56] R. T. Snodgrass (editor). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, Boston, 1995.

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

[58] 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.

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

[60] 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.