Unification of Temporal Data Models

Christian S. Jensen

Department of Mathematics and Computer Science
Aalborg University
Fredrik Bajers Vej 7E
DK-9220 Aalborg Ø, DENMARK
csj@iesd.auc.dk

Michael D. Soo Richard T. Snodgrass

Department of Computer Science
University of Arizona
Tucson, AZ 85721

{soo, rts}@cs.arizona.edu

Abstract

To add time support to the relational model, both first normal form (1NF) and non-1NF approaches have been proposed. Each has associated difficulties. Remaining within 1NF when time support is added may introduce data redundancy. The non-1NF models may not be capable of directly using existing relational storage structures or query evaluation strategies.

This paper describes 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 define a conceptual notion of a bitemporal relation where tuples are stamped with sets of two-dimensional chronons in transaction-time/valid-time space. We introduce a tuple-timestamped 1NF representation to exemplify how the conceptual bitemporal data model is related, by means of snapshot equivalence, with representational models. We then consider querying within the two-level framework. We first define an algebra at the conceptual level. We proceed to map this algebra to the sample representational model in such a way that new operators compute equivalent results for different representations of the same conceptual bitemporal relation. This demonstrates that the representational model is faithful to the semantics of the conceptual data model, with many choices available that may be exploited to improve performance.

1 Introduction

Adding time to the relational model has been a daunting task [BADW82, McK86, SS88, Soo91]. More than a dozen extended data models have been proposed over the last decade [Sno92, JS92]. Most of these models support *valid time*, that is, the time a fact was valid in the modeled reality. A few, notably [BZ82, BG89, Sno87, Sno93], have also supported *transaction time*, the time a fact was recorded in the database; such models are termed *bitemporal*, because they support both kinds of time [JCG*92].

While these data models differ on many dimensions, perhaps the basic distinction that has been oft stated is between first normal form (1NF) and non-1NF. A related distinction is between tuple timestamping and attribute value timestamping. Each has associated difficulties. Remaining within 1NF, an example being the timestamping of tuples with valid and transaction start and end times [Sno87], 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 [Gad88B], may not be capable of directly using existing relational storage structures or query evaluation strategies that depend on atomic attribute values.

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 primary task of capturing the time-varying semantics. The result has been a plethora of incompatible data models and query languages, and a corresponding surfeit of database design and implementation strategies that may be employed across these models.

We advocate instead a very simple *conceptual* data model that captures the essential semantics of time-varying relations, but has no illusions of being suitable for presentation, storage, or query evaluation. We instead rely on existing data model(s) for these tasks, by exploiting equivalence mappings between the conceptual model and the *representational* models. 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. Finally, while not addressed here, we feel that the conceptual data model is the appropriate location for database design [JSS92A] and logical query optimization.

In essence, we advocate moving the distinction between the various existing temporal data models from a semantic basis to a physical, performance-relevant basis, utilizing our proposed conceptual data model to capture the time-varying semantics.

The paper has the following outline. In the next section we define the conceptual model. We then examine a previously proposed representational data model, namely tuple timestamping (e.g., [NA89, Sad87, Sar90, Sno87, Sno93]). We provide a mapping between the conceptual model and this model, then briefly discuss additional representational models.

Having presented both the conceptual data model and exemplified representational data models, Sec-

tion 4 presents an overview of the interaction among the data models. Snapshot equivalence is the subject of Section 5. Ironically, while definitions of snapshot equivalence are particular to individual data models (as the definitions rely on model-specific operations), the notion of snapshot equivalence allows us to relate relation instances, as well as operators, of different representations, and also allows us to relate representations to the semantics ascribed to the conceptual model. Section 6 is devoted to generalizing algebraic operators of the relational model to apply to objects in the bitemporal conceptual model as well as the tuple timestamped representational model. As with data instances, we demonstrate correspondence of these operators.

2 Conceptual Bitemporal Relations

The primary reason behind the success of the relational model is its simplicity. A bitemporal relation is necessarily more complex. Not only must it associate values with facts, as does the relational model, it must also specify *when* the facts were valid in reality, as well as *when* the facts were current in the database. Since our emphasis is on semantic clarity, we will extend the conventional relational model as small an extent as necessary to capture this additional information.

2.1 Definition

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. For both domains, we assume that the database system has limited precision, and we term the smallest time unit a *chronon*. As we can number the chronons, the domains are isomorphic to the domain of natural numbers.

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, $x = (a_1, a_2, \ldots, a_n | t_b)$, 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* in the two-dimensional space spanned by valid time and transaction time. Such a set is termed a *bitemporal element*[1], denoted t_b, and is represented graphically as a set of rectangles. 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.

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 1 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 predictively on June 5th. This is shown in Figure 1(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 1(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 1(c). Lastly, Figure 1(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.

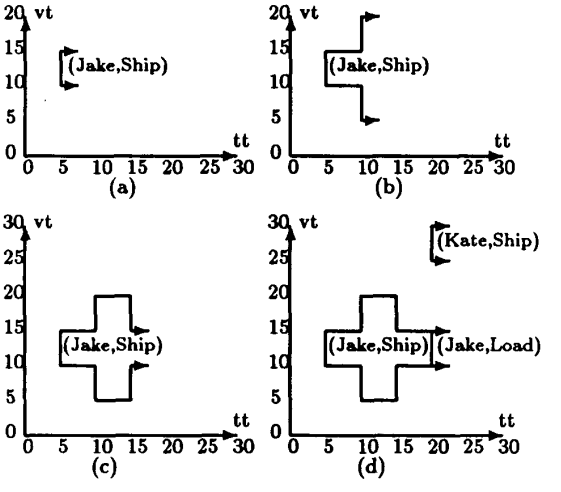

Figure 1: Bitemporal Elements

We note that the number of bitemporal chronons in a given bitemporal element is the area enclosed by

[1] Alternative, equally desirable terms include *time period set* [BZ82] and *bitemporal lifespan* [CC87].

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 1(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	$\{(NOW,25),\ldots,(NOW,30)\}$

□

2.2 Update

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

An insertion is issued when we want to record in bitemporal relation instance r 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.)

$$\texttt{insert}(r,(a_1,\ldots,a_n),t_v) =$$

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

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. (Note that representational data models, to be discussed shortly, which actually store the data on disk will not require such a special routine; it is present only in the *conceptual data model.*) 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)}
```

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

$$\texttt{delete}(r,(a_1,\ldots,a_n)) =$$

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

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

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

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

EXAMPLE: The sequence of bitemporal elements shown in Figure 1 is created by the following sequence of commands, invoked at the indicated transaction time (TT).

Command	*TT*
`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 an associated set of valid-time chronons (termed a *valid-time element* and denoted t_v), and a transaction-time tuple has an associated 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.

3 Representation Schemes

A conceptual bitemporal relation is structurally simple—it is a set of facts, each timestamped with a bitemporal element which is a set of bitemporal chronons. In this section we examine a previously proposed representation scheme for bitemporal relations. We specify the objects defined in the representation, provide the mapping to and from conceptual bitemporal relations to demonstrate that the same information is being stored, and show how updates of conceptual bitemporal relations may be mapped into updates on relations in the representation. We end by briefly considering four additional representations.

3.1 A Sample Representation Scheme

In the conceptual model, the timestamp associated with a tuple is an arbitrary set of bitemporal chronons. As such, a relation schema in the conceptual model is non-1NF, which may be difficult to implement directly. We describe here how to represent conceptual relations by 1NF snapshot relations, allowing the use of existing, well-understood implementation techniques.

Let a bitemporal relation schema $\mathcal{R}$ contain the attributes $A_1, \ldots, A_n$, T where T is the timestamp attribute defined on the domain of bitemporal elements. This schema is represented by a snapshot relation schema R as follows.

$$R = (A_1, \ldots, A_n, \mathrm{T}_s, \mathrm{T}_e, \mathrm{V}_s, \mathrm{V}_e)$$

The additional attributes T_s, T_e, V_s, V_e are atomic-valued timestamp attributes containing a starting and ending transaction-time chronon and a starting and ending valid-time chronon, respectively. These four values represent the bitemporal chronons in a rectangular region, the idea being to divide the complete region, covered by the bitemporal element of a single tuple in a conceptual relation instance, into a number of rectangles and then represent the conceptual tuple by a set of value-equivalent tuples, one for each rectangle.

There is a multitude of possible ways of covering a bitemporal element. We require that any function that covers a bitemporal element $x[\mathrm{T}]$ of a bitemporal tuple x satisfy two properties.

1. Any bitemporal chronon in $x[\mathrm{T}]$ must be contained in at least one rectangle.

2. Each bitemporal chronon in a rectangle must be contained in $x[\mathrm{T}]$.

Apart from these requirements, the covering function is purposefully left unspecified—an implementation is free to choose a covering with properties it finds desirable. For example, a set of covering rectangles need not be disjoint. Overlapping rectangles may reduce the number of tuples needed in the representation, at the possible expense of additional processing during update.

EXAMPLE: The 1NF relation corresponding to the conceptual relation in Figure 1(d) is shown below.

Emp	*Dept*	T_s	T_e	V_s	V_e
Jake	Ship	6/5	6/9	6/10	6/15
Jake	Ship	6/10	6/14	6/5	6/20
Jake	Ship	6/15	6/19	6/10	6/15
Jake	Load	6/20	*NOW*	6/10	6/15
Kate	Ship	6/20	*NOW*	6/25	6/30

Here we use a non-overlapping covering function that partitions the bitemporal elements by transaction time. □

Throughout the paper, we will use R and S to denote relation schemas. Relation instances are denoted r, s, and t, and $r(R)$ means that r is an instance of R. Attributes are denoted A_i, B_i, and C_i. For brevity, we let A denote the set of all attributes A_i. For tuples we use x, y, and z (possibly indexed), and the notation $x[A_i]$ is defined to be the value of attribute A_i for tuple x. As a shorthand, we define $x[V]$ to be the closed interval from $x[\mathrm{V}_s]$ to $x[\mathrm{V}_e]$ (i.e., a set of one-dimensional valid-time chronons), and similarly for $x[\mathrm{T}]$, a set of transaction-time chronons.

The following functions convert between a conceptual bitemporal relation instance and a corresponding instance in the representation scheme. The second argument, *cover*, of the routine **concep_to_snap** is a covering function. It returns a set of rectangles, each denoted by a set of bitemporal chronons.

```
concep_to_snap(r', cover):
    s ← ∅;
    for each x ∈ r'
        z[A] ← x[A];
        for each t ∈ cover(x[T])
            z[T_s] ← min_1(t); z[T_e] ← max_1(t);
            z[V_s] ← min_2(t); z[V_e] ← max_2(t);
            s ← s ∪ {z};
    return s
```

The functions *min_1* and *min_2* select a minimum first and second component, respectively, in a set of binary tuples. The function *max_1* returns the value *NOW* if encountered as a first component; otherwise, it returns a maximum first component. The function *max_2* selects a maximum second component.

```
snap_to_concep(r):
    s ← ∅;
    for each z ∈ r
        r ← r − {z};
        x[A] ← z[A];
        x[T] ← bi_chr(z[T], z[V]);
        for each y ∈ r
            if z[A] = y[A]
                r ← r − {y};
                x[T] ← x[T] ∪ bi_chr(y[T], y[V]);
            s ← s ∪ {x};
    return s
```

The function *bi_chr* computes the bitemporal chronons covered by the argument rectangular region.

The **concep_to_snap** routine generates possibly many representational tuples from each conceptual tuple, each corresponding to a rectangle in valid/transaction-time space. The **snap_to_concep** routine merges the rectangles associated with a single fact into a single bitemporal element.

Note that the functions are the inverse of each other, i.e., for any conceptual relation instance r',

$$\texttt{snap\_to\_concep}(\texttt{concep\_to\_snap}(r', cover)) = r'.$$

For the update routines, the most convenient covering functions partition on either valid or transaction time and do not permit overlaps. The current transaction time is c_t.

```
insert(r, (a_1, ..., a_n), t_v, cover_v):
    cvr ← cover_v(t_v);
    for each x ∈ r
        if x[T_e] = NOW and x[A] = (a_1, ..., a_n)
            for each t ∈ cvr
                if x[V] ∩ t ≠ ∅
                    cvr ← (cvr − t) ∪ (t − x[V]);
    for each t ∈ cvr
        z[A] ← (a_1, ..., a_n);
        z[T_s] ← c_t;  z[T_e] ← NOW;
        z[V_s] ← t[s];  z[V_e] ← t[e];
        r ← r ∪ {z};
    return r


delete(r, (a_1, ..., a_n)):
    for each x ∈ r
        if x[A] = (a_1, ..., a_n) and x[T_e] = NOW
            x[T_e] ← c_t;
    return r
```

The function $cover_v$ in the **insert** routine returns a set of valid-time intervals (each a set of contiguous valid-time chronons). The routine first reduces the valid time elements, produced by the covering function, to avoid overlap with the valid times of existing tuples that have a transaction time extending to *NOW* and that are value equivalent to the one to be inserted. Then, one tuple is inserted for each of the remaining valid time elements. The **delete** routine simply replaces the transaction end time with the current time, c_t.

As for the conceptual data model, **modify** is simply a combination of **delete** and **insert**.

3.2 Other Representations

The representation just discussed is a representative of the five representations that have been proposed so far to support both valid and transaction time. We briefly review each of the four remaining representations—an analysis similar to the one in Section 3.1 may be performed for each [JSS92B].

BenZvi introduced the first bitemporal representation, similar to the tuple timestamping scheme in Section 3.1, but with five timestamps: (1) valid begin, (2) valid end, (3) the transaction time when valid begin was recorded, (4) the transaction time when valid end was recorded, and (5) the transaction time when the tuple was logically deleted [BZ82].

In representations based on attribute-value timestamping (e.g., [CC87, Tan86, Gad88B, LJ88, MS91]), all information about an object is grouped within a single tuple. This capability has made attribute value timestamped representations popular for data modeling. In Gadia's TempSQL model [Gad92], which is the only model based on attribute-value timestamping that supports bitemporal relations, each attribute value has associated a transaction-time interval and a valid-time interval. Like in the representation in Section 3.1, these intervals together encode a bitemporal rectangle.

Another representation often mentioned is a sequence of valid-time states indexed by transaction time [SA85]. This representation is derived by first partitioning the transaction-time dimension according to the beginning and ending points of the transaction-time intervals of all the tuples in the bitemporal relation. Second, for each partition, all tuples current in the partition are collected along with their valid-time intervals. These sets are valid-time relations indexed by transaction time. The transaction-time interval of a partition is the existence interval of the valid-time relation, i.e., the time when the entire valid-time relation was the current state of the bitemporal relation. Alternatively, we can envision a bitemporal relation as a sequence of transaction-time states indexed by valid time.

In the backlog-based representation scheme, bitemporal relations are represented by backlogs, which are also 1NF relations [Kim78, JMRS92]. The most important difference between this and the previous schemes is that tuples (termed update requests) in bitemporal backlogs are never updated, i.e., backlogs are append-only. In addition to the explicit attribute values, a update request has four attribute values: (1) valid begin, (2) valid end, (3) the transaction time when the update request is inserted, and (4) a value indicating whether the update is an insertion or a deletion.

As for the sample representation scheme, it is possible for each of these representation schemes to devise mapping functions to and from the conceptual bitemporal relations. Thus, the results of the rest of the paper apply also to these other representations.

4 Data Model Interaction

The previously proposed representations arose from several considerations. They were all extensions of the conventional relational model that attempted to capture the time-varying nature of both the enterprise be-

ing modeled and the database, and hence incorporated support for both valid and transaction time. They attempted to retain the simplicity of the relational model; the two 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 model was perhaps best at that. And they attempted to ensure ease of implementation and query evaluation efficiency; the backlog representation may be advantageous here.

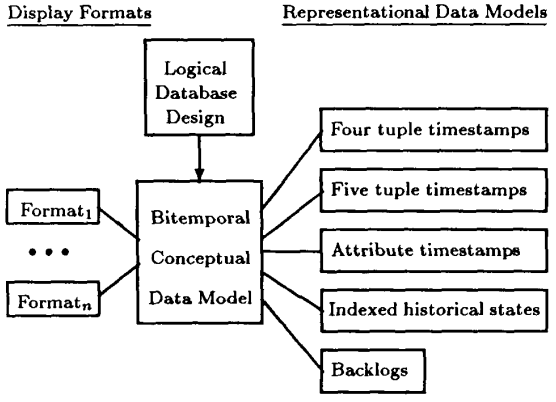

Figure 2: Interaction of Conceptual and Representational Data Models

It is clear from the number of proposed representations that meeting all of these goals simultaneously is a difficult, if not impossible task. We therefore advocate a separation of concerns. The time-varying semantics is obscured in the representation schemes by presentation and implementation considerations. We feel that the bitemporal conceptual data model proposed in this paper is the most appropriate basis for expressing this semantics. This data model is notable in its use of bitemporal chronons to stamp facts. Clearly, in most situations, this is not the most appropriate way to present the stored data to users, nor is it the best way to physically store the data. However, since there are mappings to other representations 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 2 shows the placement of the bitemporal conceptual data model and the five representational data models with respect to storage representation and display. It indicates that 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. The sample conceptual relation introduced in Section 2.1 can be expressed in the various representational data models, and each resulting relation may be appropriate for presentation in some situation, independent of how the relation is stored.

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 discussed next.

5 Semantic Equivalence

The previous section claimed that many equivalent representations of the same conceptual relation may co-exist. In this and the next section, we explore in more detail this relationship between the conceptual data model and the sample representational data model. We focus here on the objects in the models; the next section will examine operations on these objects.

5.1 Transaction and Valid Timeslice Operators

We use snapshot equivalence to formalize the notion of equivalent representations. Snapshot equivalence makes use of the notions of transaction and valid timeslice, which we define for the sample representation.

The *transaction timeslice* operator, ρ^{B}, takes two arguments, a bitemporal relation and a time value, the latter appearing as a subscript. The result is a valid-time relation. In order to explain the semantics of ρ^{B}, we describe its operation on a conceptual bitemporal relation. Each tuple is examined in turn. If any of its associated bitemporal chronons have a transaction time matching the argument time, the explicit attribute values and each of the valid-time chronons with a matching transaction time become a tuple in the result. The transaction timeslice operator may also be applied to a transaction-time relation, in which case the result is a snapshot relation.

The *valid timeslice* operator, τ^{B}, is very similar. It also takes two arguments, a bitemporal relation and a time value. The difference is that this operator does the selection on the valid time and produces a transaction-time relation. The valid timeslice operator may also be applied to a valid-time relation, in which case the result is a snapshot relation.

DEFINITION: Define a relation schema $R = (A_1, \ldots, A_n, \mathrm{T}_s, \mathrm{T}_e, \mathrm{V}_s, \mathrm{V}_e)$, and let r be an instance of this schema. Let t_2 denote an arbitrary time value and let t_1 denote a time not exceeding NOW.

$$\rho^{\mathrm{B}}_{t'}(r) = \{z^{(n+2)} \mid \exists x \in r \; (z[A] = x[A] \wedge z[\mathrm{V}] = x[\mathrm{V}] \wedge t' \in x[\mathrm{T}])\}$$
$$\tau^{\mathrm{B}}_{t}(r) = \{z^{(n+2)} \mid \exists x \in r \; (z[A] = x[A] \wedge z[\mathrm{T}] = x[\mathrm{T}] \wedge t \in x[\mathrm{V}])\} \quad \square$$

The transaction timeslice operator for transaction-time relations (ρ^{T}) and the valid timeslice operator for valid-time relations (τ^{V}) are straightforward special cases. Note further that transaction and valid timeslice may be defined for the other representational data models as well.

5.2 Snapshot Equivalence

We can now define snapshot equivalence so that it applies to each representational data model for which

the valid timeslice and transaction timeslice operators have been defined[2].

DEFINITION: Two relation instances, r and s, are *snapshot equivalent*, $r \overset{S}{\equiv} s$, if for all times t_1 not exceeding NOW and all times t_2,

$$\tau^{V}_{t_2}(\rho^{B}_{t_1}(r)) = \tau^{V}_{t_2}(\rho^{B}_{t_1}(s)) \; . \qquad \square$$

There is no reason to apply ρ before τ in this definition, as the following theorem states. Proofs of all theorems in terms of the tuple-timestamped representational data model may be found elsewhere [JSS92B].

THEOREM 1 Let r be a bitemporal relation. Then for all times t_1 not exceeding NOW and for all times t_2,

$$\tau^{V}_{t_2}(\rho^{B}_{t_1}(r)) \overset{S}{\equiv} \rho^{T}_{t_1}(\tau^{B}_{t_2}(r)) \; . \qquad \square$$

Snapshot equivalence precisely captures the notion that relation instances in the representation scheme have the same information content. More precisely, all representations of the same conceptual bitemporal relation instance are snapshot equivalent, and two bitemporal relations that are snapshot equivalent represent the same conceptual bitemporal relation.

THEOREM 2 Snapshot equivalent bitemporal relations represent the same conceptual bitemporal relation:

1. If $\texttt{concep\_to\_snap}(r', cover_1) = r_1$ and $\texttt{concep\_to\_snap}(r', cover_2) = r_2$ then $r_1 \overset{S}{\equiv} r_2$.
2. If $s_1 \overset{S}{\equiv} s_2$ then $\texttt{snap\_to\_concep}(s_1) = \texttt{snap\_to\_concep}(s_2)$. $\square$

This theorem has important consequences. For each representation, and for a given covering function, snapshot equivalence partitions the relation instances into equivalence classes. Each instance in an equivalence class maps to the same conceptual bitemporal relation instance. The semantics of the representation instance is thus identical to that of the conceptual instance. This correspondence provides a way of converting instances between disparate representations: this conversion can proceed through a conceptual instance. Finally, the correspondence provides a way of demonstrating that two instances in different representations are semantically equivalent, again by examining the conceptual instance(s) to which they map.

[2] The concept of snapshot equivalence is due to Gadia and was first defined for valid-time relations [Gad86] and was later generalized to multiple dimensions [GY88]. We have chosen to avoid the original term *weakly equivalent* to avoid confusion with the different notion of *weak equivalence* over algebraic expressions (e.g., [Ull82]). Disambiguating the original term by prefixing with "temporally" is awkward.

6 An Algebra for Bitemporal Conceptual Relations

We now examine the operational aspects of the data models just introduced. A major goal is to demonstrate the existence of the operational counterpart of the structural equivalence established in the previous section. We first define operations on conceptual bitemporal relations and then define corresponding operations on the tuple-timestamped representation. We prove that the operators preserve snapshot equivalence and are natural generalizations of their snapshot counterparts.

6.1 Definition

Define a relation schema $R = (A_1, \ldots, A_n \mid \mathrm{T})$, and let r be an instance of this schema. Let t_2 denote an arbitrary time value and let t_1 denote a time not exceeding NOW. Then the valid timeslice and transaction timeslice operators, defined in Section 5.1 for the tuple-timestamped representational model, may be defined as follows for the conceptual data model.

$$\begin{aligned}\rho^{B}_{t_1}(r) = \{z^{(n+1)} \mid \exists x \in r\,(z[A] = x[A] \wedge \\ z[\mathrm{T}_v] = \{t_2 \mid (t_1, t_2) \in x[\mathrm{T}]\} \wedge z[\mathrm{T}_v] \neq \emptyset)\} \\ \tau^{B}_{t_2}(r) = \{z^{(n+1)} \mid \exists x \in r\,(z[A] = x[A] \wedge \\ z[\mathrm{T}_t] = \{t_1 \mid (t_1, t_2) \in x[\mathrm{T}]\} \wedge z[\mathrm{T}_t] \neq \emptyset)\}\end{aligned}$$

Let D be an arbitrary set of $|D|$ non-timestamp attributes of relation schema R. The projection on D of r, $\pi^{B}_{D}(r)$, is defined as follows.

$$\begin{aligned}\pi^{B}_{D}(r) = \{z^{(|D|+1)} \mid \exists x \in r\,(z[D] = x[D]) \wedge \\ \forall y \in r\,(y[D] = z[D] \Rightarrow y[\mathrm{T}] \subseteq z[\mathrm{T}]) \wedge \\ \forall t \in z[\mathrm{T}]\,\exists y \in r\,(y[D] = z[D] \wedge t \in y[\mathrm{T}])\}\end{aligned}$$

The first line ensures that no chronon in any value-equivalent tuple of r is left unaccounted for, and the second line ensures that no spurious chronons are introduced.

Let P be a predicate defined on $A_1, \ldots, A_n$. The selection P on r, $\sigma^{B}_{P}(r)$, is defined as follows.

$$\sigma^{B}_{P}(r) = \{z \mid z \in r \wedge P(z[A])\}$$

To define the union operator, $\cup^{B}$, let both r_1 and r_2 be instances of R.

$$\begin{aligned}r_1 \cup^{B} r_2 = \{z^{(n+1)} \mid (\exists x \in r_1\,\exists y \in r_2 \\ (z[A] = x[A] = y[A] \wedge z[\mathrm{T}] = x[\mathrm{T}] \cup y[\mathrm{T}])) \vee \\ (\exists x \in r_1\,(z[A] = x[A] \wedge \\ (\neg\exists y \in r_2 (y[A] = x[A])) \wedge z[\mathrm{T}] = x[\mathrm{T}])) \vee \\ (\exists y \in r_2\,(z[A] = y[A] \wedge \\ (\neg\exists x \in r_1 (x[A] = y[A])) \wedge z[\mathrm{T}] = y[\mathrm{T}]))\}\end{aligned}$$

The first clause handles value-equivalent tuples found in both r_1 and r_2; the second clause handles those found only in r_1; and the third handles those found only in r_2.

With r_1 and r_2 defined as above, relational difference is defined as follows.

$$\begin{aligned}r_1 -^{B} r_2 = \{z^{(n+1)} \mid \exists x \in r_1\,(z[A] = x[A]) \wedge \\ ((\exists y \in r_2\,(z[A] = y[A] \wedge z[\mathrm{T}] = x[\mathrm{T}] - y[\mathrm{T}])) \vee \\ (\neg\exists y \in r_2\,(z[A] = y[A]) \wedge z[\mathrm{T}] = x[\mathrm{T}]))\}\end{aligned}$$

The last two lines compute the bitemporal element, depending on whether a value-equivalent tuple may be found in s.

In the bitemporal natural join, two tuples join if they match on the join attributes and have overlapping bitemporal element timestamps. Define r and s to be instances of R and S, respectively, and let R and S be bitemporal relation schemas given as follows.

$$\begin{aligned} R &= (A_1, \ldots, A_n, B_1, \ldots, B_l \mid \mathrm{T}) \\ S &= (A_1, \ldots, A_n, C_1, \ldots, C_m \mid \mathrm{T}) \end{aligned}$$

The bitemporal natural join of r and s, $r \bowtie^{\mathrm{B}} s$, is defined below. As can be seen, the timestamp of a tuple in the join result is computed as the intersection of the timestamps of the two tuples that produced it.

$$\begin{aligned} r \bowtie^{\mathrm{B}} s = \{ & z^{(n+l+m+1)} \mid \exists x \in r\, \exists y \in s \\ & (x[A] = y[A] \wedge x[\mathrm{T}] \cap y[\mathrm{T}] \neq \emptyset \wedge \\ & z[A] = x[A] \wedge z[B] = x[B] \wedge z[C] = y[C] \wedge \\ & z[\mathrm{T}] = x[\mathrm{T}] \cap y[\mathrm{T}])\} \end{aligned}$$

We have only defined operators for bitemporal relations. The similar operators for valid time and transaction time relations are special cases. The valid and transaction time natural joins are denoted $\bowtie^{\mathrm{V}}$ and $\bowtie^{\mathrm{B}}$, respectively; the conventional snapshot natural join is denoted $\bowtie^{\mathrm{S}}$. The same naming convention is used for the remaining operators.

6.2 Mapping the Algebra to a Representation Scheme

For each of the algebraic operators defined in the previous section, we now define counterparts for the sample representation scheme. Throughout this section, R and S denote tuple-timestamped bitemporal relation schemas, and r and s are instances of these schemas. Initially, R is assumed to have the attributes $A_1, \ldots, A_n, \mathrm{T_s}, \mathrm{T_e}, \mathrm{V_s}, \mathrm{V_e}$.

As the transaction- and valid-timeslice operators were defined already in Section 3.1, we now define in turn projection, selection, union, difference, and natural join.

To define projection, let D be an arbitrary set of $|D|$ attributes among $A_1, \ldots, A_n$. The projection on D of r, $\pi^{\mathrm{B}}_D(r)$, is defined as follows.

$$\begin{aligned} \pi^{\mathrm{B}}_D(r) = \{ & z^{(|D|+4)} \mid \exists x \in r \\ & (z[D] = x[D] \wedge z[T] = x[T] \wedge z[V] = x[V])\} \end{aligned}$$

Next, let P be a predicate defined on $A_1, \ldots, A_n$. The selection P on r, $\sigma^{\mathrm{B}}_P(r)$, is defined as follows.

$$\sigma^{\mathrm{B}}_P(r) = \{z^{(n+4)} \mid \exists x \in r\, (z = x \wedge P(x[A]))\}$$

To define the union operator, $\cup^{\mathrm{B}}$, let both r_1 and r_2 be instances of schema R.

$$r_1 \cup^{\mathrm{B}} r_2 = \{z^{(n+4)} \mid \exists x \in r_1\, \exists y \in r_2\, (z = x \vee z = y)\}$$

With r_1 and r_2 defined as above, relational difference is defined using several functions, each introduced in Section 3.1.

$$\begin{aligned} r_1 -^{\mathrm{B}} r_2 = \{ & z^{(n+4)} \mid \exists x \in r_1\, (z[A] = x[A] \wedge \\ & \exists t \in \mathit{cover}(\mathit{bi\_chr}(x[\mathrm{T}], x[\mathrm{V}]) - \\ & \{\mathit{bi\_chr}(y[\mathrm{T}], y[\mathrm{V}]) \mid y \in r_2 \wedge y[A] = x[A]\}) \wedge \\ & z[\mathrm{T_s}] = \mathit{min\_1}(t) \wedge z[\mathrm{T_e}] = \mathit{max\_1}(t) \wedge \\ & z[\mathrm{V_s}] = \mathit{min\_2}(t) \wedge z[\mathrm{V_e}] = \mathit{max\_2}(t))\} \end{aligned}$$

The new timestamp is conveniently determined by set difference on bitemporal elements.

To define the bitemporal natural join, we need two bitemporal relation schemas R and S with overlapping attributes.

$$\begin{aligned} R &= (A_1, \ldots, A_n, B_1, \ldots, B_m, \mathrm{T_s}, \mathrm{T_e}, \mathrm{V_s}, \mathrm{V_e}) \\ S &= (A_1, \ldots, A_n, C_1, \ldots, C_k, , \mathrm{T_s}, \mathrm{T_e}, \mathrm{V_s}, \mathrm{V_e}) \end{aligned}$$

In the bitemporal natural join of r and s, $r \bowtie^{\mathrm{B}} s$, two tuples join if they match on the join attributes and overlap in both valid time and transaction time.

$$\begin{aligned} r \bowtie^{\mathrm{B}} s = \{ & z^{(n+m+k+4)} \mid \exists x \in r\, \exists y \in s \\ & (x[A] = y[A] \wedge \\ & x[\mathrm{T}] \cap y[\mathrm{T}] \neq \emptyset \wedge x[\mathrm{V}] \cap y[\mathrm{V}] \neq \emptyset \wedge \\ & z[A] = x[A] \wedge z[B] = x[B] \wedge z[C] = y[C] \wedge \\ & z[\mathrm{T}] = x[\mathrm{T}] \cap y[\mathrm{T}] \wedge z[\mathrm{V}] = x[\mathrm{V}] \cap y[\mathrm{V}])\} \end{aligned}$$

6.3 Equivalence Properties

We have seen that a conceptual bitemporal relation is represented by a class of snapshot equivalent relations in the representation scheme. We now define the notion of an operator preserving snapshot equivalence.

DEFINITION: An operator α *preserves snapshot equivalence* if, for all parameters X and snapshot relation instances r and r' representing bitemporal relations,

$$r \overset{\mathrm{S}}{\equiv} r' \Rightarrow \alpha_X(r) \overset{\mathrm{S}}{\equiv} \alpha_X(r').$$

This definition may be trivially extended to operators that accept two or more argument relation instances. □

In the snapshot relational algebra, an operator, e.g., natural join, must return identical results every time it is applied to the same pair of arguments. In our framework, we require only preservation of snapshot equivalence. Thus, we add flexibility in implementing the bitemporal operators by accepting that they return different, but snapshot equivalent, results when applied to identical arguments at different times.

The operators preserve snapshot equivalence. That is, given snapshot equivalent operands each operator produces snapshot equivalent results. This ensures that the result of an algebraic operation will be correct, irrespective of covering.

THEOREM 3 The algebraic operators preserve snapshot equivalence. □

The next step is to combine the conceptual and representation level transformation functions with the representation level operators to create corresponding conceptual level operators. Given a representation level operator, α^{p}, its corresponding conceptual level operators, α^{pc}, is defined as follows.

$$\alpha^{\mathrm{pc}}_X(r') = \mathtt{snap\_to\_concep}(\alpha^{\mathrm{p}}_X(\mathtt{concep\_to\_snap}(r')))$$

Theorems 2 and 3 in combination make this meaningful and ensure that the conceptual level operators behave like the snapshot relational algebra operators—with identical arguments, they always return identical results. This is required because, like snapshot relations, conceptual bitemporal relations are unique, i.e., two conceptual relations have the same information content if and only if they are identical.

Now, we have two sets of operators defined on the conceptual bitemporal relations, namely the directly defined operators in Section 6.1 and the induced operators. In fact, we have constructed the two sets of operators to be identical. Put differently, the operators in Section 6.1 are the explicitly stated conceptual level operators, induced from the representation level operators (Section 6.2) and the transformation algorithms in Section 3.1.

Next we show how the operators in the various data models, snapshot, transaction-time, valid-time, and bitemporal, are related. Specifically, we show that the semantics of an operator in a more complex data model reduces to the semantics of the operator in a simpler data model. Reducibility guarantees that the semantics of simpler operators are preserved in their more complex counterparts.

For example, the semantics of the transaction-time natural join reduces to the semantics of the snapshot natural join in that the result of first joining two transaction-time relations and then transforming the result to a snapshot relation yields a result equivalent to that obtained by first transforming the arguments to snapshot relations and then joining the snapshot relations. This is shown in Figure 3 and stated formally in the first equivalence of the following theorem.

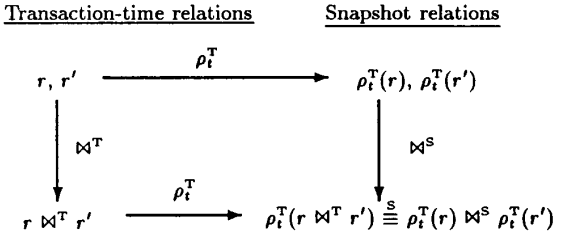

Figure 3: Reducibility of Transaction-Time Natural Join to Snapshot Natural Join.

THEOREM 4 Let t denote an arbitrary time that, when used with a transaction timeslice operator, does not exceed *NOW*. In each equivalence, let r and s be relation instances of the proper types for the given operators. Then the following hold.

$$\rho_t^T(r \bowtie^T s) \stackrel{S}{\equiv} \rho_t^T(r) \bowtie^S \rho_t^T(s)$$
$$\tau_t^V(r \bowtie^V s) \stackrel{S}{\equiv} \tau_t^V(r) \bowtie^S \tau_t^V(s)$$
$$\tau_t^B(r \bowtie^B s) \stackrel{S}{\equiv} \tau_t^B(r) \bowtie^T \tau_t^B(s)$$
$$\rho_t^B(r \bowtie^B s) \stackrel{S}{\equiv} \rho_t^B(r) \bowtie^V \rho_t^B(s)$$

A similar analysis can be made for the other operators. □

7 Summary and Future Research

In this paper, we defined the *bitemporal conceptual data model* which timestamps facts with bitemporal elements, which are sets of bitemporal chronons. We argued that it is a unifying model in that conceptual instances could be mapped into instances of existing *representational data models*. This was exemplified by a first normal form (1NF) tuple timestamped data model in which tuples were stamped with rectangles in the transaction-time/valid-time space. We also showed how an extension to the conventional relational algebraic operators could be defined in the conceptual data model, and be mapped to analogous operators in the representational models.

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: each models a different reality and thus has a distinct semantics. We employed *snapshot equivalence* to relate instances in different models, and we showed that the operators were equivalent, were snapshot-equivalence preserving, and were a natural extension of the snapshot operators.

We advocate a separation of concerns. Data presentation, storage representation, and time-varying semantics should be considered in isolation, utilizing different data models. Semantics, specifically as determined by logical database design, should be expressed in the conceptual model. Multiple presentation formats should be available, as different applications require different ways of viewing the data. The storage and processing of bitemporal relations should be done in a data model that emphasizes efficiency.

Additional research is needed in database design, utilizing the conceptual data model. It appears that normal forms may be more conveniently defined in this model than in the representational models [JSS92A]. Also, more work is needed in mapping existing temporal query language proposals into the conceptual data model.

Acknowledgements

This research was conducted while the first author visited the University of Arizona. Support was provided by the Danish Natural Science Research Council through grant no. 11-9675-1 SE, the National Science Foundation through grant IRI-8902707, the IBM Corporation through Contract #1124, and by Christian and Otilia Brorsons Mindelegat.

References

[BADW82] A. Bolour, T. L. Anderson, L. J. Dekeyser, and H. K. T. Wong. The Role of Time in Information Processing: A Survey. *SigArt Newsletter*, 80:28–48, April 1982.

[BG89] G. Bhargava and S. Gadia. Achieving Zero Information Loss in a Classical Database Environment. In *Proceedings of the Conference on Very Large Data Bases*, pages 217–224, Amsterdam, August 1989.

[BZ82] J. Ben-Zvi. *The Time Relational Model.* Ph.D. dissertation, Computer Science Department, UCLA, 1982.

[CC87] J. Clifford and A. Croker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, CA, February 1987.

[Gad86] S. K. Gadia. Weak temporal relations. In *Proceedings of ACM PODS*, pages 70–77, 1986.

[Gad88B] S. K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[Gad92] S. K. Gadia. A Seamless Generic Extension of SQL for Querying Temporal Data. Technical Report TR-92-02, Computer Science Department, Iowa State University, March 1992.

[GY88] S. K. Gadia and C. S. Yeung. A Generalized Model for a Relational Temporal Database. In *Proceedings of ACM SIGMOD*, pages 251–2259, 1988.

[JMRS92] C. S. Jensen, L. Mark, N. Roussopoulos, and T. Sellis. Using Caching, Cache Indexing, and Differential Techniques to Efficiently Support Transaction Time. *VLDB Journal*, to appear, 1992.

[JS92] C. S. Jensen and R. T. Snodgrass. Proposal of a Data Model for the Temporal Structured Query Language. TempIS Technical Report 37, Department of Computer Science, University of Arizona, Tucson, AZ, July 1992.

[JSS92A] C. S. Jensen, R. T. Snodgrass, and M. D. Soo Extending Normal Forms to Temporal Relations. Technical Report 92–17, Department of Computer Science, University of Arizona, Tucson, AZ, July 1992.

[JSS92B] C. S. Jensen, M. D. Soo and R. T. Snodgrass. Unification of Temporal Data Models. Technical Report TR–92–15, Department of Computer Science, University of Arizona, Tucson, AZ, July 1992.

[JCG*92] C. S. Jensen, J. Clifford, S. K. Gadia, A. Segev, and R. T. Snodgrass. A Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 21(3):35–43, September 1992.

[Kim78] K. A. Kimball. *The Data System.* MS thesis, University of Pennsylvania, 1978.

[LJ88] N. Lorentzos and R. Johnson. Extending Relational Algebra to Manipulate Temporal Data. *Information Systems*, 13(3):289–296, 1988.

[McK86] E. McKenzie. Bibliography: Temporal Databases. *ACM SIGMOD Record*, 15(4): 40–52, December 1986.

[MS91] E. McKenzie and R. Snodgrass. Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions. Technical Report TR–91–15, Department of Computer Science, University of Arizona, Tucson, AZ, August 1991.

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

[SA85] R. Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In *Proceedings of ACM SIGMOD*, pages 236–246, 1985.

[Sad87] R. Sadeghi. *A Database Query Language for Operations on Historical Data.* Ph.D. dissertation, Dundee College of Technology, Dundee, Scotland, December 1987.

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

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

[Sno90] R. Snodgrass. Temporal Databases: Status and Research Directions. *ACM SIGMOD Record*, 19(4):83–89, December 1990.

[Sno92] R. T. Snodgrass. Temporal Databases, in *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space,* Springer-Verlag, LNCS 639, pages 22–64, 1992.

[Sno93] R. Snodgrass. An Overview of TQuel, in *Temporal Databases: Theory, Design, and Implementation,* Benjamin/Cummings Pub. Co., to appear, 1993.

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

[SS88] R. Stam and R. Snodgrass. A Bibliography on Temporal Databases. *Database Engineering*, 7(4):231–239, December 1988.

[Tan86] A. U. Tansel. Adding Time Dimension to Relational Model and Extending Relational Algebra. *Information Systems*, 11(4):343–355, 1986.

[Ull82] J. D. Ullman. *Principles of Database Systems, Second Edition.* Computer Science Press, Potomac, Maryland, 1982.