THE UNIVERSITY OF AALBORG

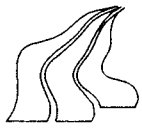
Specialized Temporal Relations

by

Christian S. Jensen & Richard Snodgrass

R 91-26 June 1991

INSTITUTE FOR ELECTRONIC SYSTEMS
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
Fredrik Bajers Vej 7 — DK 9220 Aalborg Ø — Denmark
Tel.: +45 98 15 85 22 — TELEX 69 790 aub dk



ISSN 0106-0791

Specialized Temporal Relations

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

Richard Snodgrass
Department of Computer Science
University of Arizona
Tucson, AZ 85721
rts@cs.arizona.edu

Abstract

In *temporal specialization*, the database designer restricts the relationship between the valid time-stamp (recording when something is true in the reality being modeled) and the transaction time-stamp (recording when a fact is stored in the database). An example is a *retroactive* temporal event relation, where the event must have occured before it was stored, i.e., the valid time-stamp is restricted to be less than the transaction time-stamp. We discuss some two dozen useful restrictions, defining as many specialized types of temporal relations and indicate some of their applications. We present a detailed taxonomy of specialized temporal relations. This taxonomy may be employed during database design to specify the particular time semantics of temporal relations. Additionally, the DBMS may exploit such a characterization to more efficiently store and access those temporal relations.

Many previous research efforts that considered only one kind of time also apply to certain specialized temporal relations with both kinds of time. We classify a wide range of such efforts, identifying the particular specialization each concerns. Similarly, implementation approaches that assume only one kind of time often also apply to specialized temporal relations. We analyze the extent that each technique may be modified to work with temporal relations, thereby achieving improved performance when supporting such relations. An implication of this work is that much of previous and current research that heretofore has applied only to rollback or historical databases is also relevant to restricted forms of temporal databases.

1 Introduction

In a previous paper [SA85], a taxonomy of time was introduced. This taxonomy defined three kinds of time, *user-defined time* (with no DBMS-interpreted semantics), *valid time* (when something is true in reality), and *transaction time* (when a fact is stored in the database), that induce four different types of relations. A *snapshot* relation[1] contains neither valid nor transaction time; conventional databases support snapshot relations. A *rollback* relation contains only transaction time. Such a relation records not only the current state, but it records also all states that were current at some past point in time, and associated with each state is the *transaction time* when it became current

*The work was conducted while the first author visited the University of Arizona.

[1] Though we use relational terminology throughout this paper, most of the analysis applies analogously to other data models.

and the transaction time when it ceased to be current. As a consequence, rollback relations are ever-growing. While a rollback relation reflects the history of activities in the part of reality surrounding it, a *historical* relation models the part of reality modeled by the database. Such a relation contains only valid time. A *temporal relation* contains both valid and transction time; it records both the previous states of the relation and the history of reality.

These relation types support various kinds of queries. *Current* queries, on the current state of the database and concerning what is currently true in reality, are possible by all four kinds of relations; indeed, conventional DBMS's support only this kind of query. *Historical* queries, that extract information on the history of entities in the real world, are possible on historical and temporal relations. *Rollback* queries, that extract information as recorded in the database at some point in the past, are possible on rollback and temporal relations. Queries that involve user-defined time require no special support from the DBMS, and are thus possible on all four types of relations.

This taxonomy is inadequate in its characterization of temporal relations, for several reasons. First, it assumes that each tuple is associated with at most two events or intervals (one valid and one transaction), when in fact a tuple may be associated with more than two events or intervals. As an example, when a person is hired, her salary must generally be approved at several levels of the organization, and thus the hiring record may have several transaction times associated with it. Second, many proposals for adding time to databases have considered storing only one time-stamp (e.g., [SR85, SS87]), yet it appears that both rollback and historical queries are possible in these schemes; the taxonomy explicitly forbids both kinds of queries on a relation with only one time-stamp. Third, the use of temporal databases in, e.g., monitoring and process control has introduced situations where particular relationships exist between the valid and transaction time stored in a tuple. Because the taxonomy focused on the orthogonality of the three kinds of time, it did not consider the possible relationships between them.

In this paper we extend the previously proposed taxonomy. We retain the definition of temporal relations as containing tuples having one valid time event or interval and one transaction time interval. We *specialize* such relations by adding restrictions on the transaction and valid time-stamps that already exist.

The term specialization has been used previously within data modeling. A subclass may be created from a class by means of specialization, i.e., by adding more restrictive defining properties to the definition of the class (the intension) and thus restricting the set of examples (the extension) of the class. Reversely, a superclass may be created from a class by means of generalization, i.e., by making the intension of the class less restrictive and thus expanding the extension of the class [HM81, EN89, SS77].

Temporal generalization and specialization concern the time-stamps present in a relation. They apply equally well at the conceptual and logical level, and they have consequences even at the physical level, as we shall see. Being duals of each other, temporal specialization *couples* two time-stamps by restricting their interaction, and temporal generalization *decouples* time-stamps by allowing them to be less dependent on each other.

In this paper we define various specialized types of temporal relations by restricting the relationship between valid and transaction time-stamps. These specialized types may be employed during schema design to incorporate more of the semantics of the application, and they may be utilized by the DBMS to more efficiently store and access time-varying data.

We first present an adequately general model of a temporal relation which does not restrict unnecessarily the applicability of the concepts we develop. We then examine in Section 3 the kinds of constraints one might impose, considering in turn restrictions on isolated events, on collections of events, on isolated intervals, and on collections of intervals. Many previously proposed time-oriented data models do not support general temporal relations, and some support only a single time dimension. In Section 4 we use the taxonomy to classify existing data models, and we show that even some one-dimensional models in fact support specialized temporal relations. Section 5 contains a brief analysis of how existing approaches to efficiently store and retrieve one-dimensional time-varying data may be modified to support specialized temporal relations, thereby contributing to the largely unexplored area of support for two-dimensional temporal data.

2 A Conceptual Model of a Temporal Relation

We present a conceptual model of a temporal relation as a prelude to the extensions discussed in Section 3.

The adjective "temporal" (snapshot, transaction time, and historical, as well) most often has been attributed to databases. We will take a more general approach and use it only for relations because a single database may consist of relations of several types.

A temporal relation is a sequence of *historical states* indexed by transaction time. The transaction time is generated by the system and is monotonically increasing; it is thus unique for each historical state. Transaction time is stepwise constant—the database contents do not change between transactions. As such, its semantics is entirely independent of the application or the enterprise being modeled. An historical state is composed either of a set of interval elements or of a set of event elements. An *element* has four components: an *element surrogate* (a system-generated unique identifier of this element that can be referenced and compared for equality, but not displayed to the

user), an *object surrogate* (a unique identifier, possibly user-generated, of the entity or relationship being modelled by this element), data values associated with the entity or relationship being modeled, and a valid time-stamp (an event or interval). A query on a temporal database consists of a rollback component, which selects a particular historical state for each underlying relation, followed by an historical component, which computes a new historical state, to be displayed to the user or stored back in the database.

At this point, we do not assume any particular type system on surrogates, historical states, or attributes, nor a particular data model (elements could be tuples in a relational database [Codd70] or records in a network database [Dat90] or events in a time sequence collection [SK86]), nor a particular representation. We use object surrogates to establish a correspondence between real-world objects (entities or relationships) and their database representations. At any point in time, each real-world object may have, in a single relation, a set of associated elements, all with the same object surrogate (a "life-line" [Sch77] or a "time sequence"[SK86]). Thus, a relation (a "time sequence collection" [SK86]) can be partitioned into a collection of sets so that elements of distinct sets have distinct object surrogates and elements of any single set have the same object surrogate.

While we generally apply the restrictions defined in this paper to an arbitrary relation, the concepts apply equally well to a subset of the elements of a relation and to any partition of a relation as a whole. The most notable partition is the per-object surrogate partition just mentioned.

For each element e there exists two transaction times, $tt_e^{\vdash}$ and $tt_e^{\dashv}$, such that for all historical states that contain e, the transaction time tt that indexes each historical state is in the *existence interval* for e, $[tt_e^{\vdash}, tt_e^{\dashv})$. The element surrogate identifies the element for the purpose of defining the existence interval (in the database) for the element. If a particular event or interval is (logically) deleted, then immediately re-inserted, the two resulting elements will have different element surrogates, allowing the deletion ($tt_e^{\dashv}$) and insertion ($tt_e^{\vdash}$) points to be unambiguously defined. If a modification is made by a transaction executed on the database, the element in the current historical state is (logically) deleted, and a new element, recording the modified information, is stored in the new historical state, indexed by the transaction time of the transaction making the change. In summary, there are three transaction time-stamps associated with each element e occuring in a particular historical state: tt_e, the transaction time index of that historical state, $tt_e^{\vdash}$, the transaction time index of the historical state in which the element first appeared, and $tt_e^{\dashv}$, the index of the historical state succeeding the one in which the element last appeared.

This model has been defined to be independent of representation. In particular, while an element is associated with a valid time-stamp, the model makes no mention of whether tuple-time-stamping or attribute-time-stamping is employed. The conceptual model of a sequence of historical states does

not imply (nor disallow) a particular physical representation. For example, a temporal relation can be represented as a collection of tuples with a transaction time interval time-stamp [Sno87] or with three transaction time event time-stamps [BZ82], as a backlog relation of insertion, modification, and deletion operations [JMRS90], and as attributes time-stamped with temporal elements (which are finite unions of intervals, distinct from the term element used in this paper) [Gad88].

3 Specialized Temporal Relations

In this section, we characterize temporal relations according to the inter-relations of their time-stamps. In Sections 3.1 and 3.2, we consider singly stamped elements (event stamped), and in Sections 3.3 and 3.4, we consider doubly stamped elements (interval stamped). In Sections 3.1 and 3.3, we characterize relations on the basis of the stamps of individual elements in isolation, and in Sections 3.2 and 3.4, we characterize relations on the basis of the inter-relations of stamps of distinct elements.

All the definitions of relation types in this section are intensional definitions, i.e., for a relation schema to have a particular type, all its possible (non-empty) extensions must satisfy the definition of the type. The restrictions ususally apply only to the historical state in which the element was inserted or only the historical state in which the element was logically deleted (i.e., the one following the historical state in which the element last appears). Throughout we assume that the valid and transaction time-stamps are drawn from the same domain, which must be totally ordered, i.e., has an associated less-than equality relation. We do not consider alternative transaction time domains such as version numbers that cannot be compared with valid time.

3.1 Taxonomy on Isolated Events

In this section we consider only *events* that take place at an instant of time in reality. Let R be a temporal relation, and let e be an element of R. Each element e has a single valid time, vt_e, indicating when the event took place in reality. We consider only a single transaction time, tt_e, which is either the insertion or the deletion time, that is, either $tt_e^{\vdash}$ or $tt_e^{\dashv}$. Each property (e.g., *retroactive*) is relative to one of these two times. For example, it is possible for a relation to be *deletion retroactive* but not *insertion retroactive*. As discussed earlier, a modification consists of a a deletion followed by an insertion. If a relation is, say, deletion retroactive *and* insertion retroactive, it can also be considered modification retroactive. The definitions that follow will mention only a single valid time vt_e and a single transaction time tt_e. In examples where we illustrate the definitions, we will assume that tt_e is $tt_e^{\vdash}$ (i.e., we consider insertion, not deletion nor modification).

We formally define a number of specialized temporal relations by restricting the allowed inter-relations between valid and transaction time-stamp values of isolated elements. Eleven of the specialized relations are illustrated in Figure 1. The bold, vertical line in the center represents the transaction time, tt_e, of a element. The valid time of the element has certain relationships with this transaction time. The surrounding dotted lines represent bounds. In a non-specialized temporal relation (termed *general*), there are no restrictions on the inter-relations of the transaction and valid time-stamps of a element.

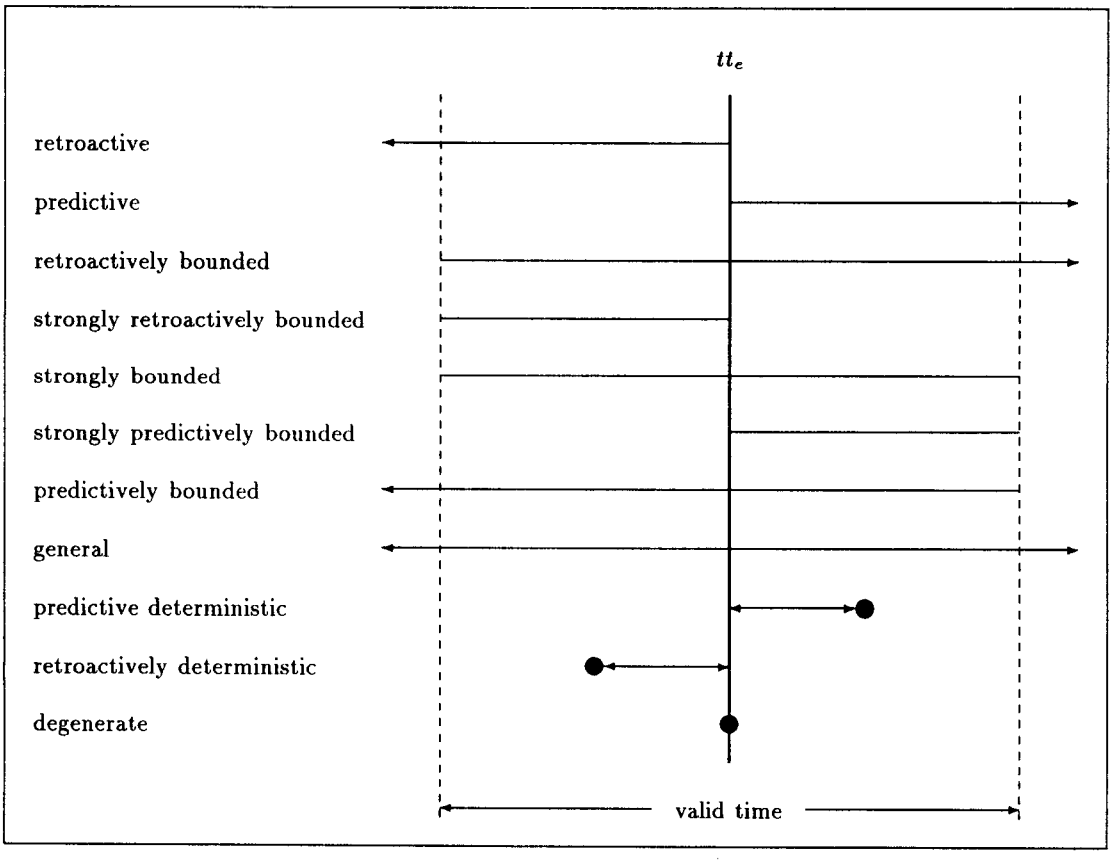


Figure 1: Possible Values of the Valid Time-stamp Relative to the Transaction Time-stamp

DEFINITION: Temporal relation R is *retroactive* if

$$\forall e \in R\,(vt_e \leq tt_e)$$

Thus, the values of an element are valid before they are entered into the relation, i.e., the event

occured before it was stored. Retroactive relations are common in monitoring situations, such as process control in a chemical production plant, where variables such as temperature and pressure are periodically sampled and stored in a database for subsequent analysis.

DEFINITION: Temporal relation R is *predictive* if

$$\forall e \in R\ (vt_e \geq tt_e)$$

Thus, the values of an element are not valid until some time after they have been entered into the relation. An example is a relation that records direct-deposit payroll checks. Generally a copy of this relation is made on magnetic tape near the end of the month, and sent to the bank so that the payments can be effective on the first day of the next month.

In elements of retroactively bounded temporal relations, the valid time stamp never is less than the transaction time-stamp by more than a bounded time interval. In all bounded relations, the bounds are fixed at schema definition time.

DEFINITION: Temporal relation R is *retroactively bounded* if there exists a bound $\Delta t_1 \geq 0$ such that

$$\forall e \in R\ (vt_e \geq tt_e - \Delta t_1)$$

Note that in a retroactively bounded relation, the valid time-stamp may exceed the transaction time-stamp. An example is a relation recording the project each employee is assigned to. While assignments arbitrarily into the future may be recorded, it is generally feasible to require that an assignment will be recorded in the database no later than a month after it is effective.

A strongly retroactively bounded relation is a retroactively bounded temporal relation where the valid time-stamp is less than the transaction time-stamp.

DEFINITION: Temporal relation R is *strongly retroactively bounded* if there exists a bound $\Delta t_1 \geq 0$ such that

$$\forall e \in R\ (tt_e - \Delta t_1 \leq vt_e \leq tt_e)$$

The sample relation just discussed is a strongly retroactively bounded relation if future assignments are not stored in the relation.

A strongly predictively bounded relation is symmetrical to the previous specialized temporal relation. Here the valid time-stamp is in a bounded time interval after the transaction time-stamp.

DEFINITION: Temporal relation R is *strongly predictively bounded* if there exists a bound $\Delta t_2 \geq 0$ such that

$$\forall e \in R\ (tt_e \leq vt_e \leq tt_e + \Delta t_2)$$

Direct deposit pay checks are usually of this form: the company wants them to be valid on the first of the month, but also wants to cut the checks (i.e., make the tape to be sent to the bank) as late as possible, generally one week before.

In a strongly bounded relation, the valid time-stamp may only deviate from the transaction time-stamp within both upper and lower bounds.

DEFINITION: Temporal relation R is *strongly bounded* if there exist bounds $\Delta t_1 \geq 0$ and $\Delta t_2 \geq 0$ such that

$$\forall e \in R\ (tt_e - \Delta t_1 \leq vt_e \leq tt_e + \Delta t_2)$$

Here, information concerns only the current situation, except that recently valid information and information valid in the near future can be recorded and updated. An example is an accounting relation recording the current month's transactions. Corrections to entries of previous months are stored as compensating transactions in the current month; transactions concerning future months are made to a separate relation.

In elements of predictively bounded temporal relations, the valid time stamp never exceeds the transaction time-stamp by more than a bounded delay. Thus, this kind of relation is symmetrical to retroactively bounded relations.

DEFINITION: Temporal relation R is *predictively bounded* if there exists a bound $\Delta t_2 \geq 0$ such that

$$\forall e \in R\ (vt_e \leq tt_e + \Delta t_2)$$

Note that in a predictively bounded relation, the valid time-stamp may be less than the transaction time-stamp. In such relations, only information concerning the near-term future may be stored. An

example is an order database in which pending orders, constrained by company policy to be no more than 30 days in the future, are stored along with previously filled orders.

A temporal relation is degenerate if the transaction and valid time-stamps of an element are identical (within the selected granularity).

DEFINITION: Temporal relation R is *degenerate* if

$$\forall e \in R\,(vt_e = tt_e)$$

An example is a monitoring situation in which there is no time delay (within the time-stamp granularity) between sampling a value and storing it in the database.

At the implementation level, a degenerate temporal relation can be advantageously treated as a rollback relation due to the fact that relations are append-only and elements are entered in time-stamp order—this will be discussed in more detail in Section 5.

A *mapping function* m for a relation R takes as argument an element e of a relation and returns a valid time-stamp, computed using any of the attributes of e, excluding vt_e. A temporal relation R is deterministic if it has a mapping function that correctly computes the valid time-stamps of its elements. Sample mapping functions include $m_1(e) = tt_e^{\vdash} + \Delta t$ ("valid after a fixed delay"), $m_2(e) = \lfloor tt_e^{\vdash} - \Delta t \rfloor_{hrs}$ ("valid from the most recent hour"), and $m_3(e) = \lceil tt_e^{\vdash} + 8 \text{ hrs} \rceil_{day}$ ("valid from the next closest 8:00 a.m.").

DEFINITION: Temporal relation R is *deterministic* if there exists a mapping function m such that

$$\forall e \in R\,(vt_e = m(e))$$

Similarly, a relation is *non-determinate* if such a function does not exist.

In analogy to the retroactive and predictive relations, a deterministic relation can be either retroactively deterministic or predictively deterministic.

DEFINITION: Temporal relation R is *retroactively deterministic* if there exists a mapping function m such that

$$\forall e \in R\,((vt_e = m(e)) \wedge (m(e) \leq tt_e))$$

For example, a relation is retroactively deterministic if each element is valid from the beginning of the most recent hour during which it was stored.

DEFINITION: Temporal relation R is *predictively deterministic* if there exists a mapping function m such that

$$\forall e \in R\,((vt_e = m(e)) \wedge (m(e) \geq tt_e))$$

For example, a relation is predictively deterministic if it is valid from the next closest 8:00 a.m. Such a relation might be relevant in banking applications for deposits that are not effective until the start of the next business day.

Finally, the above two types of relations can be bounded.

DEFINITION: Temporal relation R is *strongly retroactively bounded deterministic* if there exists a mapping function m and a bound Δt_1 such that

$$\forall e \in R\,((vt_e = m(e)) \wedge (tt_e - \Delta t_1 \leq m(e) \leq tt_e))$$

DEFINITION: Temporal relation R is *strongly predictively bounded deterministic* if there exists a mapping function m and a bound Δt_2 such that

$$\forall e \in R\,((vt_e = m(e)) \wedge (tt_e \leq m(e) \leq tt_e + \Delta t_2))$$

The examples given previously were in fact bounded.

The generalization/specialization structure of the specialized temporal relations defined above is presented in Figure 2. A relation type can be specialized into any of the successor relation types (top to bottom), and a relation type inherits all the properties of its predecessor relation types (bottom to top).

There is one type of restriction that has not yet been discussed, namely *transaction time incompleteness*. A temporal relation must record all previous historical states to permit arbitrary rollback. A temporal relation may be approximated by a series of periodic snapshots [AL80, Adi81, LHM+86]. A snapshot rate may be associated with each relation. Rollback is possible only to those transaction times at which the snapshot was taken, and thus the relation is incomplete with respect to the

transaction time dimension. For example, if periodic or user-initiated snapshots are taken of an historical relation then that relation along with its snapshots form a temporal relation on which rollbacks to particular transaction times are possible, i.e., a *transaction time incomplete general* temporal relation. Similarly, the application of periodic snapshots to a standard relation yields a *transaction time incomplete degenerate* temporal relation.

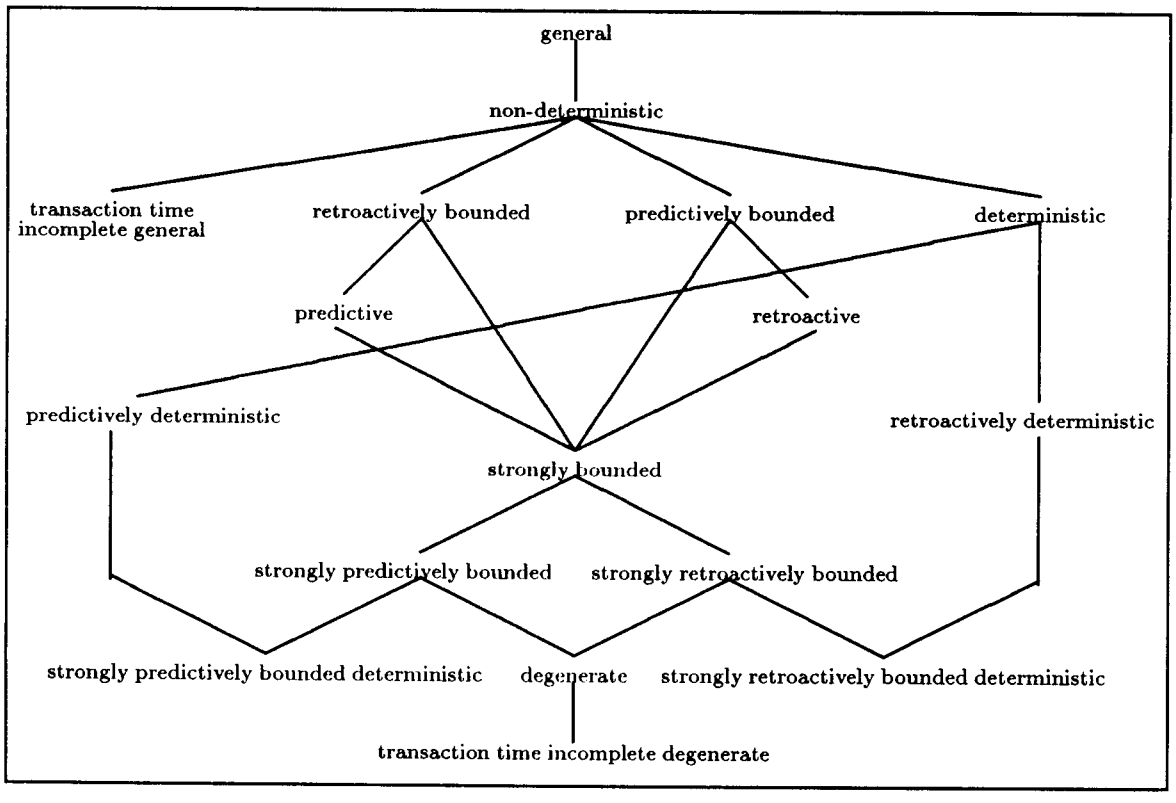


Figure 2: Generalization/specialization Structure of the Event-oriented Taxonomy

3.2 Inter-event Based Taxonomy

The previous definitions were based on predicates on isolated events (elements) of relations: a relation schema had a given property if each individual element of any extension, meaningful in the modeled reality, of the schema satisfied the relevant predicate. We now define restrictions on relation schemas based on the interrelationships of multiple events (elements) in all possible extensions. In this and later sections, we continue to assume in the examples and explanations that tt_e is $tt_e^{\vdash}$.

DEFINITION: Temporal relation R is *globally sequential* if[2]

$$\forall e \in R\, \forall e' \in R\, (tt_e < tt_{e'} \Rightarrow (\max(tt_e, vt_e) \leq \min(tt_{e'}, vt_{e'})))$$

As a weaker condition, R is *per-surrogate sequential* if $\forall x \in \pi_{Id}R$, $\sigma_{Id=x}R$ is globally sequential, where Id is the surrogate attribute. In a globally sequential relation, each event must occur *and* be stored before the next event occurs or is (predictively) stored.

Now we introduce the notion of ordered temporal relations. A relation is ordered if elements are entered in valid time-stamp order.

DEFINITION: Temporal relation R is *globally ordered* if

$$\forall e \in R\, \forall e' \in R\, (tt_e < tt_{e'} \Rightarrow vt_e \leq vt_{e'})$$

As above, R is *per-surrogate ordered* if $\forall x \in \pi_{Id}R$, $\sigma_{Id=x}R$ is globally ordered (again, Id is the surrogate attribute). Sequentiality is generally a stronger property than ordered. However, if the relation is degenerate then the two properties are identical.

DEFINITION: Temporal relation R is *transaction time event regular* if there exists a periodic interval $\Delta_t \geq 0$ and some integer $k_e^{e'}$ for each pair of elements e and e' such that

$$\forall e \in R\, \forall e' \in R\, \exists k_e^{e'} (tt_e = tt_{e'} + k_e^{e'} \Delta_t)$$

Note that the transaction time-stamps of successively stored elements need not be evenly spaced; they are merely restricted to be separated by an integral multiple ($k_e^{e'}$) of a specified interval Δ_t. An example is a periodic sampling of some physical variable such as temperature.

DEFINITION: Temporal relation R is *valid time event regular* if there exists a periodic interval $\Delta_t \geq 0$ and some integer $k_e^{e'}$ for each pair of elements e and e' such that

$$\forall e \in R\, \forall e' \in R\, \exists k_e^{e'}\ (vt_e = vt_{e'} + k_e^{e'} \Delta_t)$$

[2] Alternatively, we could define sequentiality as follows:

$$\forall e \in R\, \forall e' \in R\, ((e = e') \vee (\max(tt_e, vt_e) \leq \min(tt_{e'}, vt_{e'})) \vee (\min(tt_e, vt_e) \geq \max(tt_{e'}, vt_{e'})))$$

The concept of *granularity* of valid time-stamps can be expressed in terms of this property. For example, if the valid time-stamp granularity is one second, then equivalently all relations in the database are valid time event regular relative to one second.

DEFINITION: Temporal relation R is *temporal event regular* if there exists a periodic interval $\Delta_t \geq 0$ and some integer $k_e^{e'}$ for each pair of elements e and e' such that

$$\forall e \in R\, \forall e' \in R\, \exists k_e^{e'}\, ((vt_e = vt_{e'} + k_e^{e'} \Delta_t) \wedge (tt_e = tt_{e'} + k_e^{e'} \Delta_t))$$

A periodic degenerate relation is trivially temporal event regular.

Analogously to the ordering properties, these regularity properties can be defined in a global or per-surrogate fashion. However, they have the additional aspect (not shared with ordering) that the per-surrogate variant implies the global variant.

All of the these characterizations are orthogonal to those given in the previous sections for individual events, except that a degenerate event relation must be globally ordered.

3.3 Taxonomy on Isolated Intervals

We now turn to interval relations, that is, those relations for which, for each element e of the relation, the valid time is an interval, $[vt_e^{\vdash}, vt_e^{\dashv})$. The transaction times of the element, $tt_e^{\vdash}$ and $tt_e^{\dashv}$ are defined as before. As in Section 3.2, k (possibly indexed) is an integer.

The previous characterizations of events may also be applied to either $vt_e^{\vdash}$ or $vt_e^{\dashv}$. For example, if an interval is stored as soon as it terminates, a designer may state that the interval relation is $vt^{\vdash}$-retroactive and $vt^{\dashv}$-degenerate. If the relation is, say, $vt^{\vdash}$-retroactive and $vt^{\dashv}$-retroactive, it may simply be termed retroactive.

A temporal relation is transaction time regular, valid time regular, or temporally regular if the transaction time intervals, valid time intervals, or both transaction time and valid time intervals are regular, respectively.

DEFINITION: Temporal relation R is *transaction time interval regular* if there exists a unit of time $\Delta_t \geq 0$ such that

$$\forall e \in R\, \exists k_e\, (tt_e^{\dashv} = tt_e^{\vdash} + k_e \Delta_t)$$

DEFINITION: Temporal relation R is *valid time interval regular* if there exists a unit of time $\Delta_t \geq 0$ such that

$$\forall e \in R \, \exists k_e \, (vt_e^{\dashv} = vt_e^{\vdash} + k_e \Delta_t)$$

Alternatively, the duration of all intervals in such a relation is an integral multiple of a specified time unit.

An example is a relation recording new hires that observes a company policy that all such hires be effective on either the first or the fifteenth of each month.

DEFINITION: Temporal relation R is *temporal interval regular* if there exists a unit of time $\Delta_t \geq 0$ such that

$$\forall e \in R \, \exists k_e^1 \, \exists k_e^2 \, ((tt_e^{\dashv} = tt_e^{\vdash} + k_e^1 \Delta_t) \wedge (vt_e^{\dashv} = vt_e^{\vdash} + k_e^2 \Delta_t))$$

Above we applied the concept of regularity to relations. The concept may be applied also on, e.g., a per-surrogate basis (as discussed in section 2).

3.4 Inter-interval Based Taxonomy

As with events, we distinguish restrictions that are applied individually to all intervals and restrictions on the interrelationship between multiple intervals in a relation. The restrictions listed below apply to relations. Similar restrictions may be applied on, e.g., a per-surrogate basis.

DEFINITION: Temporal relation R is *globally sequential* if

$$\forall e \in R \, \forall e' \in R \, (tt_e < tt_{e'} \Rightarrow (\max(tt_e, vt_e^{\dashv}) \leq \min(tt_{e'}, vt_{e'}^{\vdash})))$$

In such a relation, each interval must occur and be stored before the next interval commences. An example involves the relation previously discussed that records the weekly assignments for employees. If the assignment for the next week is recorded during the weekend then this relation will be per-surrogate seqential.

A relation is ordered if elements are entered in valid time-stamp order.

DEFINITION: Temporal relation R is *globally ordered* if

$$\forall e \in R \, \forall e' \in R \, (tt_e < tt_{e'} \Rightarrow vt_e^{\dashv} \leq vt_{e'}^{\vdash})$$

Concerning the example just discussed, let us now record the next week's assignment on the Thursday of the current week. In this case the transaction time of the next week's interval (on a per-surrogate basis) will overlap the valid time of the next week's interval, and the relation will be per-surrogate ordered.

As with events, sequentiality is a stronger property than ordered.

DEFINITION: Temporal relation R is *globally valid time contiguous* if

$$\forall e \in R\,(\exists e' \in R\,(vt_e^{\dashv} = vt_{e'}^{\vdash} \wedge tt_e < tt_{e'} \wedge \neg(\exists e''(tt_{e''} \neq tt_e \wedge tt_{e''} \neq tt_{e'} \wedge tt_e < tt_{e''} < tt_{e'}))) \vee (\forall e' \in R\,(vt_e^{\dashv} > vt_{e'}^{\vdash})))$$

This definition states that in a valid time contiguous relation, the end of one event coincides with the start of the next event that is stored, unless the event is the last one in the sequence, in which case it occurs after all the other events. An example will be given shortly.

The final property is stronger than the others.

DEFINITION: Temporal relation R is *globally temporally contiguous* if it is globally valid time contigous and if

$$\forall e \in R\,(vt_e^{\vdash} \leq tt_e \leq vt_e^{\dashv})$$

3.5 Event and Interval Interrelationships

Let us consider how event and interval properties relate to one another. A common implementation technique is to store incoming events in a *backlog* relation [Kim78, Jen90], and derive an interval relation by interpreting each event as ending an interval started by the previous event (on a global or per-surrogate basis) and starting a new interval. An example is an event relation recording promotions and their associated title and salary changes; the resulting interval relation records when the salaries and titles were in effect. If the backlog of events is globally (alternatively, per-surrogate) sequential then the derived interval relation will be globally (per-surrogate) sequential. The same holds for globally/per-surrogate ordered. If the backlog is transaction time (valid time, temporal) event regular, then the derived interval relation will be interval regular. In all cases, the

derived interval relation will be globally (per-surrogate) valid time contiguous. Hence, our example interval relation will be per-surrogate ordered, sequential, and valid time contiguous.

Also observe that a temporal interval relation is valid time interval regular or temporal interval regular if both its starting ($vt^{\vdash}$) and ending ($vt^{\dashv}$) times are valid time event regular or temporal event regular, respectively. In such relations, the starting and ending time of each element are related to the starting and ending time of other elements by an integral multiple of Δ_t.

4 Classification of Existing Temporal Data Models

The taxonomy of specialized temporal relations provides a coherent framework of concepts that covers all the existing proposed temporal relational data models known to the authors and allows us to more faithfully describe, distinguish, and understand these data models. We illustrate this by using the taxonomy to perform such a characterization. We proceed by successively applying greater temporal specialization.

4.1 General Temporal Relations

General temporal relations are found in the data models presented in [BZ82, Sno87].

The snapshot mechanism [AL80] may be extended to support general temporal relations. A *snapshot* of a relation is an independent copy of the current state of that relation at the time of the snapshot. Thus, snapshots are derived from base relations, but they do not change when the underlying base relations change [Adi81, LHM+86]. The snapshot mechanism may be applied to a relation in three ways [ABQPdO85, AQ86, AQ87]. First, there is the manual snapshot where a GENERATE-VERSION command is used to create a shapshot (termed a "manual album"). Second, there is the periodic snapshot (termed an "automatic album") where, for example, the user may specify, "keep snapshots for the end of the month for a window of 12 months". Third, there is the successive snapshot where the system creates a new snapshot every time the underlying relation is updated (termed a "movie").

While Adiba only applies the snapshot mechanisms to conventional relations, there is no reason why they cannot be applied also to historical relations. Successive snapshots of an historical relation (an historical movie) result in a general temporal relation. Applying the snapshot mechanism manually or automatically to historical or conventional relations produces specialized temporal relations, as we shall see shortly.

4.2 Retroactive Temporal Relations

In [GY88], a multi-dimensional temporal data model is presented which in turn is restricted to a two-dimensional data model with valid and transaction time as the dimensions. In this model, however, only data valid in the past may be stored. For example, it is impossible to store on May 11, 1991 the fact that "as of now, Dr. Jones is hired as an assistant professor from September 1, 1991 till August 31, 1997." Therefore, the model does not support fully general temporal relations; instead it supports retroactive temporal relations. The restriction to retroactive temporal data is inherited from the (retroactive) historical data model of [Gad88] where event time stamps are used for the modeling of real-world activity.

Sarda proposes another specialized temporal data model in which current facts may be appended and where retrospective updates (changes to information about the past) are possible [Sar90]. Hence, the transaction time is always equal to or after the valid time, and, like the previous model, this model supports retroactive temporal relations.

4.3 Strongly Retroactively Bounded Relations

In real-time databases, transactions are associated with hard real-time deadlines [AGM90]. If the deadline passes before the transaction is executed, the transaction is unscheduled. Hence, the transaction time of information used by a transaction associated with a deadline must be strongly retroactively bounded; otherwise the transaction deadline makes no sense. Also, the transaction time of the information stored or modified by the transaction is strongly retroactively bounded, with its bound being the bound of the information triggering the transaction plus the bound of the deadline.

4.4 Degenerate Temporal Relations

Relations representing time sequences and time sequence collections of the TSC model [SK86, RS87, SS87, SS88a] may be classified as degenerate temporal relations. Such sequences are totally ordered in time; presumably facts are recorded in the database as soon as they are collected. Among the representations given for TSCs [SS88b] is a per-surrogate valid time contiguous relation that is also per-surrogate sequential.

The Postgres data model [Re87, Sto87] supports degenerate temporal relations, in that facts valid now in the real world are stored now, and all past states are retained. The Postgres query language [SRH90] supports rollback (viewing the time dimension as transaction time) and historical time-slice (viewing the time dimention as valid time), but does not support general historical queries. This

query language may be viewed alternatively as an extended rollback query language or as a highly restricted historical/temporal query language.

Jensen's data model is fundamentally a transaction time model because it models the history of the update activity of the database and makes the assumption that time-varying attributes have stepwise constant semantics [JMR91, JM91, Jen90]. However, because it allows for irregular time stamps reflecting real time, it may be used as a temporal data model when transaction and valid time coincide, and hence is also a degenerate temporal model. Similarly, successive snapshots of a conventional relation (a movie) produce a degenerate temporal relation.

In the applicative data model [HC89], changes cannot be made to data that has already been stored; hence, an applicative historical relation is a degenerate temporal relation.

In [AQ86], an append-only relation is presented which may be modified using special error-correcting operations. Without the ability to modify, this is a degenerate temporal relation. With the ability to change the past, it is an historical relation, restricted in that one can't change or record future events.

Finally, a variety of data formats are available for time series analysis [Dub91]. Some are degenerate; some are transaction time event regular; and most are globally ordered.

4.5 Transaction Time Incomplete Temporal Relations

When applied to ordinary relations, manual and periodic snapshots produce transaction time incomplete degenerate relations. Because a snapshot is a copy of the current state when the snapshot is made, it is possible to rollback to a previously current state if a snapshot was made during the time when that state was current. Thus, unless a snapshot is made whenever the current state is updated (i.e., unless we have a movie), one must guess ahead of time which roll-backs will be needed later.

When applied to historical relations, manual and periodic snapshots produce transaction time incomplete general relations. Here, historical queries are fully supported, but rollback to only some of the transaction times is possible.

4.6 Other Properties

Interestingly, no one to our knowledge has studied the predictive or deterministic variants, even though situations exist where such specialized temporal relations are useful.

5 Implications for Query Optimization and Execution

In this section, we consider the performance implications for query processing resulting from querying specialized temporal relations as opposed to general temporal relations. Specifically, we indicate how query processing algorithms and indexing techniques designed for one-dimensional time-varying data may be extended naturally to apply to specialized temporal relations as well. This represents a simple but significant contribution to the largely unexplored field of efficient support of temporal data. We proceed in two steps. First, we describe the general idea of applying one-dimensional approaches to two-dimensional data; second, we briefly review related research and show how the general idea applies. We do not attempt to give a detailed analysis of the application of one-dimensional approaches to specialized temporal relations. That would require us to select specific stored representations, indexes, and processing strategies for temporal data, which is beyond the scope of this paper.

5.1 Exploiting Specified Restrictions

The general idea can be stated as follows: In order to apply existing techniques, used to improve the performance of queries on one-dimensional data, utilize the specific inter-relation between valid and transaction time stamps, guaranteed by the type of a specialized temporal relation, to simply disregard one time dimension and only pay interest to the other as far as physical organization is concerned. Note that both the existing techniques for transaction time alone and the existing techniques for valid time alone are applicable. Because elements resulting from update activity arrive, by definition, in transaction time stamp order at temporal relations, we find it natural to utilize the transaction time dimension and ignore the valid time dimension.

This approach applies, with some variations, to all specialized temporal relations. Rather than consider each type of specialized temporal relation in turn, we confine the presentation to consider only strongly retroactively bounded relations as an example. Also, we assume that elements are physically clustered on transaction time within a complete relation (e.g., [JMRS90]) or for each object surrogate in a relation (e.g., [SG89, SL89]). These are straight-forward representations, particularly if write-once storage media are utilized. Assuming physical clustering and strongly retroactively bounded temporal data, the following important property holds: All elements with valid time stamps equal to some value, t_x, may be found within a limited number of elements after the element with transaction time value equal to t_x, if it exists and the element with the largest transaction time stamp less than t_x, otherwise. The limit in the number of elements depends on the particular retroactive bound and the intensity of the update activity for the relation.

This property of locality may have significant performance implications for some historical queries. From potentially having to search an entire ever-growing temporal relation, search may be confined to a restricted region. Indeed, the storage structure chosen for a temporal relation may be strongly dependent on the specialized type of that relation. Particularly, if bounds are satisfactorily tight, performance enhancing strategies used for one-dimensional data (valid or transaction time) may prove applicable. Order preserving physical organizations for one-dimensional data seem especially promising because order preservance in the transaction time dimension carries over to the valid time dimension and results in elements that are nearly clustered with respect to valid time.

5.2 Application to Previous Proposals

The field of efficient query processing against temporal relations is largely unexplored. While much research is still needed, the efficient support of queries on data with single time-dimensions of various kinds has been addressed. As stated, it appears that this research may be extended naturally to include the efficient support of specialized temporal data. Below, we briefly review some of this research.

First, we will consider two approaches to efficiently support various joins on one-dimensional temporal data.

In [LM90], a stream processing approach to temporal (semi-) joins is explored. In this approach, the input to, and the output from, stream processors consist of sets of streams of elements. A processor has a local state, and it is allowed to see only a single element from each stream at a time. For example, a join processor has two input streams and one output stream. When constructing a stream processor for computing a function such as a join, it is often necessary to make trade-offs between possible sort orderings of input and output streams, the size and contents of the local processor state, and the number of passes needed over the input streams. With this approach, the effect of different sort orderings on the efficiency and the size of the local state were considered for one temporal join (and as special cases, two semi-joins). A stream join processor that assumes a time ordered sequence of elements may be converted into a processer that will accept a transaction time ordered stream (nearly ordered in valid time) and yet efficiently computes a valid time temporal join. This may be done by simply adding two identical pre stream processors that each use a buffer to convert nearly ordered data into totally ordered data (an integration into a single processor may improve performance). The buffer sizes correspond to the sizes of the regions mentioned in the general discussion above.

In [SG89, GSS89, GS89], a more traditional approach to the processing of one-dimensional temporal joins is chosen. Most notably, a temporal event-join consisting of three time-oriented joins is

considered [SG89]. In this work, the proper ordering of argument relations has again been shown to significantly impact the efficiency with which joins can be performed. As above, this research may be applied to specialized temporal relations at the expense of some added complexity to the join algorithms. Thus, additional control structure and book-keeping is necessary to process nearly ordered data as opposed to the current totally ordered data. In particular, results obtained for append-only databases are highly relevant for specialized temporal relations.

Next we briefly survey recent contributions to the problem of indexing various kinds of one-dimensional temporal data. The reader should consult the references below for pointers to other work.

A number of research contributions aimed at supporting time-varying data subscribe to the philosophy that storing previously current/valid data as well as current data should not adversely affect the performance of queries accessing only current data to any significant degree. The Time-Split B-tree [SL89, LS90] is a recent contribution based on this philosophy. In addition to the key splits of the B-tree, this index structure allows for so-called time splits. The basic idea of the time-split is to migrate data to a separate and ever-growing historical database if the data resides in the current database (where it was initially inserted) and if it has also time stamps that are smaller than the split time. We believe that the time-split mechanism may be modified to make this indexing technique suitable for some types of specialized temporal data.

Also based on the above-mentioned philosophy, [KS89] generalizes R-trees to span both magnetical and optical disk media thus providing new intermediates in-between R-trees residing on only a single medium. This is relevant when the bulk of temporal data does not fit on magnetical disk and must be migrated [Re87] to optical disk. In [KS90], some tactics aimed at improving observed deficiencies of existing indexing techniques for historical data (e.g., R-trees) are introduced. We find it probable that this research may be extended to deal successfully with specialized temporal data.

In [EWK90, EKW91], an indexing technique based on the B-tree is presented. It uses end-points of intervals of validity for the indexing of elements. How to extend this technique to cover specialized temporal data is an interesting topic.

In [JMR91, JMRS90], transaction time data is stored in backlogs clustered on the time dimension. On top of the backlogs, indexed and selectively cached views together with differential (incremental and decremental) computation techniques are employed together with standard query processing techniques. The specialized temporal relations with "close" valid and transaction times may be easily integrated into and efficiently supported by this powerful query processing framework.

6 Conclusion

The original taxonomy in [SA85] argued that transaction time and valid time are orthogonal concepts. However, in certain situations, such as process monitoring, some accounting applications, and real time databases, the valid and transaction times are *coupled.* An extreme example is when an event is stored in the database as soon as it occurs, resulting in identical transaction and valid time-stamps.

In this paper we introduce *temporal specialization*, where the database designer restricts the relationship between the valid and transaction time-stamps, resulting in a specialized temporal relation. We presented a taxonomy with four parts, one each for isolated events, inter-event restrictions, isolated intervals, and inter-interval restrictions. This taxonomy may be employed during database design to specify the particular time semantics of temporal relations. We then classified existing temporal data models according to the taxonomy, discovering that many particular specialized relation types have appeared in the literature. Finally, we examined existing query optimization and execution techniques, demonstrating that research that heretofore has applied only to rollback or historical databases is also relevant to certain kinds of specialized temporal databases.

Future work is indicated in several areas. As we have shown, specialized temporal relations present an opportunity to optimize temporal queries; more work is needed to exploit restrictions stated by the database designer. Our contention is that most previous work in this area is relevant; still, the details need to be worked out. Also, the dual of temporal specialization, namely *temporal generalization*, should be explored. Just as a degenerate temporal relation containing one time-stamp is a specialization of a general temporal relation with two time-stamps, a temporal relation containing several transaction time-stamps may be characterized as a generalization of the original temporal relation. We are currently studying how temporal databases fit into the application system as a whole and how multiple transaction time-stamps arise. Finally, the taxonomy described here needs to be integrated into an overall approach to designing temporal databases. While much progress has been made in developing useful methodologies for conventional database design, very little is known about how to design temporal databases.

Acknowledgements

This topic was first explored during the SIGMod '90 banquet between the second author and Arie Segev, whose comments encouraged further exploration of what was then termed "coupled relations". Mike Stonebraker's use of valid time examples to illustrate rollback relations in Postgres (which only record transaction time) implied that Postgres supports temporal relations; the research into this

implication reported here shows that in a real sense it is true.

This work was supported in part by NSF grant ISI-8902707. In addition, the first author was supported by Danish NSF (Statens Naturvidenskabelige Forskningsråd) grant 11-8696-1 SE.

References

[ABQPdO85] M. Adiba, N. Bui Quang, and J. Palazzo de Oliveira. Time Concept in Generalized Data Bases. In *ACM Thirteenth Annual Computer Science Conference*, pages 214–223, October 1985.

[Adi81] Michel Adiba. Derived Relations: A Unifyed Mechanism for Views, Snapshots and Distributed Data. In *Proceedings of the Seventh International Conference on Very Large Data Bases*, pages 293–305, 1981.

[AGM90] R.K. Abbott and H. Garcia-Molina. Scheduling Real-Time Transactions: a Performance Evaluation. Technical report, Department of Computer Science, Princeton University, 1990.

[AL80] Michel E. Adiba and Bruce G. Lindsay. Database Snapshots. In *Proceedings of the Sixth International Conference on Very Large Databases*, pages 86–91, 1980.

[AQ86] M. Adiba and N. Bui Quang. Historical Multi-Media Databases. In *Proceedings of the Twelfth International Conference on Very Large Data Bases*, pages 63–70, August 1986.

[AQ87] M. Adiba and N.B. Quang. Dynamic Database Snapshots, Albums, and Movies. In *Proceedings of the Conference on Temporal Aspects in Information Systems*, pages 207–225, France, May 1987. AFCET.

[BZ82] J. Ben-Zvi. *The Time Relational Model.* PhD thesis, Computer Science Department, UCLA, 1982.

[Codd70] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[Dat90] C.J. Date. *An Introduction to Database Systems — Volume I.* Systems Programming Series. Addison-Wesley, Reading, MA, fifth edition, 1990.

[Dub91] B. Dubrovsky. Universal Data Access for Time Series Analysis. *PIXEL*, 2(1):42–44, March/April 1991.

[EKW91] Ramez Elmasri, Yeong-Joon Kim, and Gene T. J. Wuu. Efficient Implementation Techniques for the Time Index. In *Proceedings of the Seventh International Conference on Data Engineering*, 1991.

[EN89] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., 390 Bridge Parkway, Redwood City, California 94065, 1989.

[EWK90] Ramez Elmasri, Gene T. J. Wuu, and Yeong-Joon Kim. The Time Index: An Access Structure for Temporal Data. In *Proceedings of the Sixteenth International Conference on Very Large Data Bases*, pages 1–12, August 1990.

[Gad88] K. Shashi Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[GS89] Himawan Gunadhi and Arie Segev. A Framework for Query Optimization in Temporal Databases. Technical report, LBL-26417, School of Business Administration and Computer Science Research Department, Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720, 1989.

[GSS89] Himawan Gunadhi, Arie Segev, and J. George Shantikumar. Selectivity Estimation in Temporal Databases. Technical report, LBL-27435, School of Business Administration, University of California at Berkeley and Information and Computing Sciences Division, Lawrence Berkeley Laboratory, 1 Cyclotron Road, Berkeley, California 94720, 1989.

[GY88] Shashi K. Gadia and Chuen-Sing Yeung. A Generalized Model for a Relational Temporal Database. In *Proceedings of the ACM SIGMOD '88*, pages 251–259, 1988.

[HC89] J. P. Held and J. V. Carlis. The Applicative Data Model. *Information Sciences*, pages 249–283, 1989.

[HM81] M. Hammer and D. McLeod. Database Description with SDM: A Semantic Database Model. *ACM Transactions on Database Systems*, 6(3):351–386, September 1981.

[Jen90] Christian S. Jensen. Towards the Realization of Transaction Time Database Systems. Ph.D. Dissertation, CS-TR-2568, UMIACS-TR-90-144, Department of Computer Science. University of Maryland, College Park, MD 20742, December 1990.

[JM91] Christian S. Jensen and Leo Mark. Queries on Change in an Extended Relational Model. *IEEE Transactions on Knowledge and Data Engineering*, 3(4):to appear, December 1991.

[JMR91] Christian S. Jensen, Leo Mark, and Nick Roussopoulos. Incremental Implementation Model for Relational Databases with Transaction Time. *IEEE Transactions on Knowledge and Data Engineering*, 3(3):to appear, September 1991.

[JMRS90] Christian S. Jensen, Leo Mark, Nick Roussopoulos, and Timos Sellis. Using Caching, Cache Indexing, and Differential Techniques to Efficiently Support Transaction Time. Technical report, CS-TR-2413, UMIACS-TR-90-25, Department of Computer Science. University of Maryland, College Park, MD 20742, February 1990. Submitted for publication.

[Kim78] K. A. Kimball. The DATA System. Master's thesis, University of Pennsylvania, 1978.

[KS89] Curtis Kolovson and Michael Stonebraker. Indexing Techniques for Historical Databases. In *Proceedings of the Fifth International Conference on Data Engineering*, pages 127–137, February 1989.

[KS90] Curtis P. Kolovson and Michael Stonebraker. S-Trees: Database Indexing Techniques for Multi-Dimensional Interval Data. Memorandum, UCB/ERL M90/35, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, April 1990.

[LHM+86] Bruce Lindsay, Laura Hass, C. Mohan, Hamid Pirahesh, and Paul Wilms. A Snapshot Differential Refresh Algorithm. In *Proceedings of the ACM SIGMOD '86*, pages 53–60, May 1986.

[LM90] T. Y. Cliff Leung and Richard R. Muntz. Query Processing for Temporal Databases. In *Proceedings of the Sixth International Conference on Data Engineering*, pages 200–208, February 1990.

[LS90] D. Lomet and B. J. Salzberg. The Performance of a Multiversion Access Method. In *Proceedings of the ACM SIGMOD '90*, pages 353–363, May 1990.

[Re87] Lawrence A. Rowe and Michael R. Stonebraker (eds.). The Postgres Papers. Memorandum, UCB/ERL M86/85, University of California, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, June 1987.

[RS87] Doron Rotem and Arie Segev. Physical Organization of Temporal Data. In *Proceedings of the Third International Conference on Data Engineering*, pages 547–553, February 1987.

[SA85] Richard Snodgrass and Ilsoo Ahn. A Taxonomy of Time in Databases. In *Proceedings of the ACM SIGMOD '85*, pages 236–246, 1985.

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

[Sch77] Ben-Michael Schueler. Update Reconsidered. In *Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems*, pages 149–164, 1977.

[SG89] Arie Segev and Himawan Gunadhi. Event-join Optimization in Temporal Relational Databases. Technical report, LBL-26600, Computer Science Research Department, Lawrence Berkeley Laboratory, 1 Cyclotron Road, Berkeley, California 94720, 1989.

[SK86] Arie Shoshani and Kyoji Kawagoe. Temporal Data Management. In *Proceedings of the Twelfth International Conference on Very Large Data Bases*, pages 79–88, August 1986.

[SL89] B. J. Salzberg and D. Lomet. Access Methods for Multiversion Data. In *Proceedings of the ACM SIGMOD '89*, pages 315–324, June 1989.

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

[SR85] Michael Stonebraker and Lawrence A. Rowe. The Design of Postgres. Memorandum, UCB/ERL 85/95, Electronics Research Laboratory, College of Engineering, University of California, Berkeley 94720, California, November 1985.

[SRH90] M. Stonebraker, L.A. Rowe, and M. Hirohama. The Implementation of POSTGRES. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):125–142, mar 1990.

[SS77] John Miles Smith and Diane C. P. Smith. Database Abstractions: Aggregation and Generalization. *ACM Transactions on Database Systems*, 2(2):105–133, June 1977.

[SS87] Arie Segev and Arie Shoshani. Logical Modeling of Temporal Data. In *Proceedings of the ACM SIGMOD '87*, pages 454–466, May 1987.

[SS88a] A. Segev and A. Shoshani. Modeling Temporal Semantics. In M. Leonard C. Rolland, F. Bodart, editor, *Temporal Aspects in Information Systems*, pages 47–58. North-Holland, 1988.

[SS88b] A. Segev 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.

[Sto87] Michael Stonebraker. The Design of the Postgres Storage System. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 289–300, September 1987.