Temporal Specialization and Generalization

Christian S. Jensen, *Member, IEEE*, and Richard Snodgrass, *Senior Member, IEEE*

***Abstract*—A standard relation has two dimensions: attributes and tuples. A *temporal relation* contains two additional orthogonal time dimensions, namely, valid time and transaction time. Valid time records when facts are true in the modeled reality, and transaction time records when facts are stored in the temporal relation.**

Although, in general, there are no restrictions between the valid time and transaction time associated with each fact, in many practical applications, the valid and transaction times exhibit more or less restricted interrelationships that define several types of *specialized temporal relations*. The paper examines five different areas where a variety of types of specialized temporal relations are present.

In application systems with multiple, interconnected temporal relations, multiple time dimensions may be associated with facts as they flow from one temporal relation to another. For example, a fact may have an associated transaction time indicating when it was stored in a previous temporal relation. The paper investigates several aspects of the resulting *generalized temporal relations*, including the ability to query a predecessor relation from a successor relation.

The presented framework for generalization and specialization allows researchers as well as database and system designers to precisely characterize, compare, and thus better understand temporal relations and the application systems in which they are embedded. The framework's comprehensiveness and its use in understanding temporal relations are demonstrated by placing previously proposed temporal data models within the framework. The practical relevance of the defined specializations and generalizations is illustrated by sample realistic applications in which they occur. The additional semantics of specialized relations are especially useful for improving the performance of query processing.

***Index Terms*— Query processing, specialized temporal relations, generalized temporal relations, taxonomy, time attributes, temporal database, temporal semantics, transaction time, valid time**

I. Introduction

THIS paper explores a variety of specialized semantics of ordinary and generalized n-dimensional temporal relations. The time of validity of a fact in a temporal relation and the time when the fact was recorded in the relation are ostensibly independent. Yet, in many applications of temporal relations, the two times interact in restricted ways. For example, in the monitoring of temperature during a chemical experiment, temperature measurements are recorded in the temporal relation *after* they are valid, as a result of the transmission delays. The resulting relation is termed *retroactive*. Alternatively, salary payments recorded in the temporal relation of a bank are recorded *before* the time the funds become accessible to employees, resulting in a *predictive* relation.

We explore a variety of temporal relations with specialized relationships between transaction and valid time [30]. Such *specialized* temporal relations occur in many practical applications, and the framework presented here is a means of capturing more of the semantics of temporal relations, with two primary benefits. Used by designers and researchers, the framework conveys a more detailed understanding of temporal relations. The additional semantics, when captured by an appropriately extended database system, may also be used for selecting appropriate storage structures, indexing techniques, and query processing strategies.

When facts flow between temporal relations, several time dimensions may be associated with individual facts, resulting in *generalized* temporal relations. For example, consider the fact that an employee is given a salary raise by a manager. This fact has an associated time when the raise is effective, as well as the time when it is entered into the relation on the manager's workstation. Later this fact is copied into the centralized departmental personnel relation, and is associated with an additional time value, namely, the time when it was stored there. Thus, the personnel relation has three time dimensions. Sometimes it is possible to query one relation from another relation. In the example, it is possible to query the time-varying relation on the manager workstation indirectly via the personnel relation on the centralized machine.

The paper extends a previously presented taxonomy on time in databases [54], [55]. The previous taxonomy defined three kinds of time that could be associated with facts: *user-defined time* (with no database system-interpreted semantics), *valid time* (when a fact is true in reality), and *transaction time* (when a fact is stored in the database).

Depending on which kinds of time are associated with its facts, a relation may be of one of four types. In a *snapshot* relation, a fact has neither a valid time nor a transaction time; conventional databases support snapshot relations. In a *rollback* relation, a fact has a transaction time only. Such a relation records the current state in addition to each state that was current at some past point in time. Associated with each state is a transaction time when it became current and a transaction time when it ceased to be current. Consequently, a rollback relation is ever-growing. While a rollback relation

Manuscript received November 25, 1991; revised June 16, 1992. This work was conducted while the first author visited the University of Arizona, and it was supported in part by the National Science Foundation under Grant ISI-8902707, and in part by the Danish Natural Science Research Council (Statens Naturvidenskabelige Forskningsråd) under Grant 11-8696-1 SE.

C. S. Jensen is with the Department of Mathematics and Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark; e-mail: csj@iesd.auc.dk.

R. Snodgrass is with the Department of Computer Science, University of Arizona, Tucson, AZ 85721 USA; e-mail: rts@cs.arizona.edu.

IEEE Log Number 9213340.

reflects the history of update activities, a *historical* relation models the part of reality modeled by the database. A fact in such a relation has valid time only. Finally, a fact in a *temporal relation* has both a valid and a transaction time. A temporal relation inherits the properties of both rollback and historical relations, and it records both the previous states of the relation and the history of reality.

The four relation types support three kinds of queries. All four kinds of relations support *current* queries, queries on the current state of the database; indeed, conventional database systems support only this kind of query. Historical and temporal relations support *historical* queries, which extract facts about the history of objects from the modeled reality. Rollback and temporal relations support *rollback* queries, which extract facts as stored in the database at some point in the past. All four types of relations support queries that involve user-defined time; these queries require no special support from the database system.

The original taxonomy falls short in its characterization of temporal relations in three ways. First, the taxonomy fails to give an adequate understanding of some time-extended relations. Many proposals for adding time to databases advocate storing a single timestamp per fact (e.g., [28], [49], [58]), yet it appears that both rollback and historical queries are possible in these schemes. However, the taxonomy explicitly forbids both kinds of queries on a relation with only one timestamp per tuple. Second, because the taxonomy focuses on the orthogonality of the three kinds of time, it ignores restricted interrelationships between the valid and transaction times of facts in temporal relations. Third, the taxonomy assumes that each fact has at most one transaction time and one valid timestamp (interval or event).[1] However, in application systems with multiple, interconnected temporal relations, multiple time dimensions may be associated with facts as they flow from one temporal relation to another.

In order to address the first and second of the shortcomings, we explore the space of restricted interrelations—in between the extremes of identity and no interrelation at all—that are possible between the valid and transaction times of facts. To address the third shortcoming, we provide the means for specifying the application system contexts of temporal relations.

The paper is structured as follows. In Section II, we present a general definition and description of a temporal relation. In the following section, we examine the kinds of restrictions that one might impose on temporal relations, considering in turn restrictions on isolated events, collections of events, isolated intervals, and collections of intervals. We are not concerned here with the semantics of time-varying attributes, i.e., how to use timestamp values and stored attribute values to derive the value of a time-varying attribute. For example, we do not address the issue of how to derive the temperature of a chemical reaction at an arbitrary point in time from timestamped and stored temperature measurements. We are interested only in the semantics of the timestamps themselves.

[1] From now on, we use the shorter, but not quite precise, terms "valid timestamp" and "transaction timestamp."

The framework developed here allows researchers as well as database and system designers to precisely characterize, compare, and thus better understand specialized temporal relations. Many previously proposed time-oriented data models do not support general temporal relations, and some support only a single time dimension. In Section IV, we use the framework to classify existing data models, and we show that some 1-D models do in fact support specialized temporal relations. To show how the framework may be used to characterize and compare types of temporal relations, we place the temporal relations of all time-oriented data models known to the authors within the framework. This also indicates that we have succeeded in making the framework comprehensive, an important property.

In Section V, we introduce generalized temporal relations. In Section VI, we present two sample application systems with embedded, generalized temporal relations. Queries on generalized relations may provide the same answers as queries on the underlying relations. In Section VII, we examine means for the database system to ensure that such queries always yield correct results.

Database systems may exploit the additional semantics of temporal relations, captured using the framework, to enhance performance. The additional semantics may be used to improve display, to aid in integrity checking, and to improve the performance of query processing on the specialized relations. Section VIII contains a brief analysis of how existing approaches to efficiently store and retrieve 1-D time-varying data may be modified to support specialized temporal relations, thereby contributing to the lightly researched area of support for 2-D temporal data. We show how much of the research that heretofore has applied only to rollback or historical databases is also relevant to restricted forms of temporal databases. New research efforts targeted at directly supporting 2-D temporal data may also exploit the additional semantics discussed in this paper. The final section summarizes our work and points to future research.

II. A Conceptual Model of a Temporal Relation

We present a conceptual model of a temporal relation as a prelude to the extensions discussed in the remainder of the paper. Note that the adjective "temporal" (snapshot, rollback, and historical as well) has most often been attributed to databases. We 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 has two orthogonal time dimensions: valid time and transaction time. *Valid time* is used for capturing the time-varying nature of the part of reality being modeled by the relation. *Transaction time* models the update activity of the relation. Thus, a temporal relation may be envisioned as a sequence of *historical states* indexed by transaction time.

A temporal relation consists of a set of *temporal items*, each of which records one or more facts about an object (entity or relationship) from the part of reality being modeled by the temporal relation. Temporal items have the following attribute values:

- item surrogate,
- object surrogate,
- transaction timestamp,
- valid timestamp (interval or event),
- time-invariant attribute values,
- time-varying attribute values, and
- user-defined times.

An *item surrogate* is a system-generated unique identifier of an item that can be referenced and compared for equality, but not displayed to the user [10], [22]. We discuss item surrogates in more detail shortly.

An *object surrogate* is a unique identifier of the object being modeled by an item. It is used for identifying all the database representations of individual real-world objects. At any point in time, each real-world object may have, in a single relation, a set of associated items, all with the same object surrogate (cf. a "lifeline" [46] or a "time sequence" [51]). Thus, a relation (cf. a "time sequence collection" [51]) can be partitioned into a collection of sets so that items of distinct sets have distinct object surrogates, and items of any single set have the same object surrogate. This is termed a *per surrogate* partitioning.

Transaction times are generated by the database system itself using monotonically increasing timestamp generators (TSG's); thus, each historical state has an associated unique transaction time. The granularity of transaction timestamps is arbitrary, as long as uniqueness is ensured. Transaction time models the update activity of the temporal relation, and, as such, its semantics are entirely independent of the application and the enterprise being modeled. The transaction time of an item is the time when the facts recorded by the item were stored in the relation. No stored transaction time exceeds the current time. The historical state resulting from a transaction remains unchanged from the time of that transaction to the time of the next transaction. Therefore, the semantics of transaction time has been characterized as stepwise constant.

We associate two transaction times, $tt_e^{\vdash}$ and $tt_e^{\dashv}$, with each item e in a temporal relation. The first, $tt_e^{\vdash}$, is the time when the item e is stored in the relation. The second, $tt_e^{\dashv}$, is the time when the item e is logically removed from the relation. The *existence interval* for e, $[tt_e^{\vdash}, tt_e^{\dashv})$, is thus the time between the transaction time of the historical state in which the item first appeared and the transaction time of the historical state succeeding the one in which the item last appeared.

The item surrogate identifies the item for the purpose of defining the existence interval (in the database) for the item. If a particular event or interval is (logically) deleted, and then immediately reinserted, the two resulting items will have different item 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 item in the current historical state is (logically) deleted, and a new item, recording the modified information, is stored in the new historical state, indexed by the transaction time of the transaction making the change.

The database system uses the transaction times for items for implementing the rollback operator [8], [46]. In general, any domain of items with an identity relation and a total ordering is suitable for transaction time. Example domains include the natural numbers and regular data and time values [13].

Valid times are usually supplied by the user, but they may be system-generated. The valid timestamp of an item records when the facts represented by the time-varying and user-defined time attribute values are true in reality. Valid times are always drawn from the domain of times and data. The items of a relation may represent events, in which case the valid timestamp of an item is a single valid time value. Alternatively, the facts represented by the items of a relation may be true for a duration of time, in which case the valid timestamp of an item is an interval consisting of two valid time values. The valid timestamps are used by the database system for implementing the time-slice operator [8], [31].

An item may contain a number of *time-invariant attribute values*, i.e., values that never change. An important example is the *time-invariant key* [40], which, although it resembles the object surrogate, is still necessary. Social Security, account, and membership numbers are important time-invariant keys in many applications. Non-key time-invariant attribute values also exist, e.g., race.

An item may record several facts about a real-world object, using several *time-varying attribute values*. For example, an item may record both the title and the salary of an employee. Each relation may have an individual valid timestamp granularity, or the database system may impose a fixed granularity on all relations managed by the database system. Although different granularities may be ascribed to individual time-varying attributes within an item, it may still be necessary to fix the (overall) item granularity.

Just as an item may have several time-varying attribute values, it may have several *user-defined times*. User-defined times are drawn from a domain of dates and times with an identity relation and a total ordering, i.e., has an associated less-than relation. User-defined times may be manually supplied or computed by an application program. The system gives no special semantics to user-defined times, and user-defined times are most appropriately thought of as specialized kinds of time-varying attribute values [56], [57].

In this paper, we focus on the timestamp attributes of temporal relations alone. The treatment of the time-varying attributes is a separate issue that is beyond the scope of the presentation.

Note that in this conceptual model, we do not assume any particular type system on historical states or attributes. In particular, while an item is associated with a valid timestamp, the model makes no mention of whether tuple timestamping or attribute-value timestamping is employed. Neither do we assume a particular data model; items could be tuples in a relational database [9], records in a network database [11], or events in a time sequence collection [51]. Finally, the conceptual model of a sequence of historical states does not imply (nor disallow) a particular physical representation. For example, a temporal relation may be represented as a collection of tuples with an event or interval valid timestamp and an interval transaction timestamp [53], or with one or two valid timestamps and three transaction timestamps [8], as tuples containing attributes timestamped with one or more

finite unions of intervals (termed *temporal elements* [17]), and as a backlog relation of insertion, modification, and deletion operations (tuples) with single transaction timestamps [29] or with time warp attributes [62].

III. Specialized Temporal Relations

In this section, we characterize temporal relations according to the interrelations of their timestamps. In Sections III-A and III-B, we consider singly stamped times (event stamping), and in Sections III-C and III-D, we consider doubly stamped items (interval stamping). In Section III-A and III-C, we characterize relations considering the timestamps of individual times in isolation, and in Section III-B and III-D, we characterize relations considering the interrelations of timestamps of distinct items. In Section III-E, we present a final, orthogonal specialization of temporal relations. Then, in Section III-F, we relate the specializations of event and interval temporal relations. In Section III-G, we apply properties on a per-relation basis to portions of a relation. We provide examples for most of the specialized temporal relations defined here. The section concludes with a summary.

All the definitions of relation types in this section are intentional definitions; i.e., in order for a relation schema to have a particular type, all its possible (nonempty) extensions must satisfy the definition of the type. The restrictions usually apply only to the historical state in which the item was inserted or the historical state in which the item was logically deleted (i.e., the one following the historical state in which the item last appears). Throughout we assume that the valid and transaction timestamps are drawn from the same domain, which must be totally ordered. We do not consider in this section transaction time domains such as version numbers that cannot be compared with valid time.

The specializations presented in this paper apply to temporal relations, i.e., sets of items, and they are all defined in terms of an ordered pair of timestamp attributes. Specializations apply to any ordered pair of timestamp attributes. The natural choice is the pair consisting of the primary transaction time attribute and the valid time attribute, both of which are present in all temporal relations.

Just as the specializations may be applied to an entire relation, i.e., on a *per-relation* basis, they may be applied in turn to each partition of a relation, i.e., on a *per-partition* basis. This is true because the partitions are sets of items. Specifically, a relation satisfies a specialization on a per-partition basis if every partition of the particular partitioning in turn satisfies the specialization on a per-relation basis. Although many partitionings are possible, the most useful partitioning is the per-surrogate partitioning mentioned in the previous section. It is solely for simplicity that we state explicitly specializations on mainly a per-relation basis. In fact, the application of the specializations on a per-partition basis may, in many situations, prove to be more relevant, as discussed in Section III-G.

By its very nature, a taxonomy should be comprehensive. While striving toward achieving this, we have attempted to include only specializations that are of practical interest. We show in Section III-A that with some restrictions, the taxonomy based on isolated events is complete in that all possible interactions are accommodated. The interevent-based taxonomy is restricted to cover the concepts of sequentiality and regularity, and the isolated interval-based taxonomy covers only regularity. The interinterval-based taxonomy distinguishes between temporal relations, where items successive in transaction time have valid time intervals related in one of the 13 possible ways of ordering two intervals. In this sense, the taxonomy is comprehensive within its scope.

The space of specialized temporal relations in the taxonomy may be too large for some uses. To address this potential problem, we have organized the specializations in generalization/specialization hierarchies. Applications that require a small number of specializations may simply consider only the more general specializations.

A. 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 item of R. Each item 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*, where an item is valid before it is operated on in the database) 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 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 mention only a single valid time vt_e and a single transaction time tt_e. In examples where we illustrate the definitions, we assume that tt_e is $tt_e^{\vdash}$ (i.e., we consider insertion, not deletion or modification).

We formally define a number of specialized temporal relations by restricting the allowed interrelations between valid and transaction timestamp values of isolated items. Fifteen of the specialized relations are illustrated in Fig. 1 (an alternative depiction of subareas of the 2-D valid time/transaction time space is given elsewhere [30]). The bold, vertical line in the center represents the transaction time, tt_e, of an item. The valid time of the item may have a certain relationship with this transaction time. The surrounding dotted lines represent bounds. In a nonspecialized temporal relation (termed *general*), there are no restrictions on the interrelations of, or correlation between, the transaction and valid timestamps of an item. The dots for the three last cases in the figure symbolize specific valid times computed in terms of corresponding transaction times.

Definition: Temporal relation R is *retroactive* if

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

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

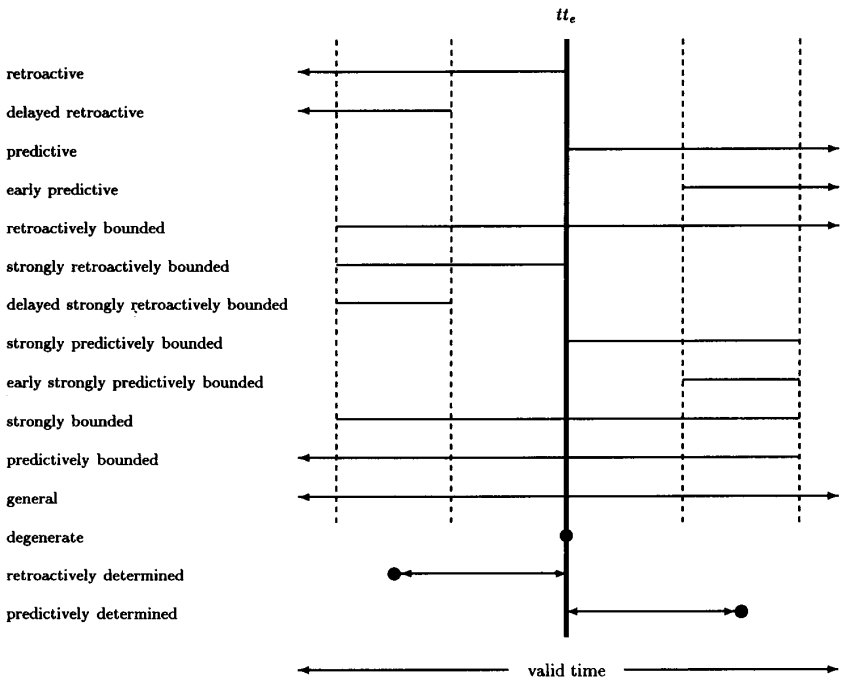


Fig. 1. Possible values of the valid timestamp relative to the transaction timestamp.

sampled and stored in a database for subsequent analysis. Further, it is often the case that some (non-negative) minimum delay between the actual time of measurement and the time a storage can be determined. For example, a particular setup for the sampling of temperatures may result in delays that always exceed 30 s. This gives rise to a delayed retroactive relation.

Definition: Temporal relation R is *delayed retroactive with bound* $\Delta t \geq 0$ if

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

In this and in the other specializations that refer to a time bound Δt, this time bound is a *duration* that may be fixed in length (e.g., 30 s, one day) or calendar-specific. An example of the latter is one month, where a month in the Gregorian calendar contains 28 to 31 days, depending on the data to which the duration is added or subtracted.

Definition: Temporal relation R is *predictive* if

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

Thus, the values of an item 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 is sent to the bank so that the payments can be effective on the first day of the next month. Analogously with the delayed retroactive temporal relation that specializes the retroactive temporal relation, the early predictive temporal relation is the specialization of the predictive temporal relation.

Definition: Temporal relation R is *early predictive with bound* $\Delta t \geq 0$ if

$$\forall e \in R\ (vt_e \geq tt_e + \Delta t).$$

The direct deposit payroll check relation is an example if the tape must be received by the bank at least, say, three days before the day on which the deposits are to be made effective. Also, this type of relation may be encountered within early warning systems where warnings must be received sometime in advance.

In items of retroactively bounded temporal relations, the validtimestamp never is less than the transaction timestamp by more than a bounded time interval. In all bounded, delayed, and early relations, the bounds are fixed at schema definition time.

Definition: Temporal relation R is *retroactively bounded with bound* $\Delta t \geq 0$ if

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

Note that in a retroactively bounded relation, the valid timestamp may exceed the transaction timestamp. An example is a relation recording the project to which each employee is assigned. Although assignments may be recorded arbitrarily into the future, an assignment is required to be recorded in the database no later than one month after it is effective.

Definition: Temporal relation R is *strongly retroactively bounded with bound* $\Delta t \geq 0$ if the following is true:

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

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

In a delayed strongly retroactively bounded relation, an additional upper bound (minimum delay) is also imposed.

Definition: Temporal relation R is *delayed strongly retroactively bounded with bound* $\Delta t_1 \geq 0$ and $\Delta t_2 \geq 0$, where $\Delta t_1 \leq \Delta t_2$, if

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

The relation that records the assignments of employees is an example of this type of relation if only past assignments are recorded, e.g., if assignments are recorded at most one month after they were effective and if it takes at least two days from the time when an assignment is finished until this is known by the data entry clerk.

The strongly predictively bounded and the early strongly predictively bounded relations are symmetrical to the two previous specialized temporal relations. Here, the valid timestamp is in a bounded duration after the transaction timestamp, and the early specialization also adds a (positive) lower bound on the valid timestamp.

Definition: Temporal relation R is *strongly predictively bounded with bound* $\Delta t \geq 0$ if the following is true:

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

Definition: Temporal relation R is *early strongly predictively bounded with bounds* $\Delta t_1 \geq 0$ and $\Delta t_2 \geq 0$, where $\Delta t_1 \leq \Delta t_2$, if

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

Direct deposit of paychecks illustrates both types of specialization. The company wants the checks to be valid on the first of the month, but it wants also to make the tape to be sent to the bank as late as possible, generally at most one week before the end of the month. In addition, the bank needs the tape at least three days in advance.

Definition: Temporal relation R is *strongly bounded with bounds* $\Delta t_1 \geq 0$ and $\Delta t_2 \geq 0$ if the following is true:

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

Here, the valid timestamp may deviate only from the transaction timestamp within both upper and lower bounds. Intuitively, 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.

Definition: Temporal relation R is *predictively bounded with bound* $\Delta t \geq 0$ if

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

Note that in a predictively bounded relation, the valid timestamp may be less than the transaction timestamp. In such relations, only information concerning the past and the near-term future may be stored. An example is an order database in which pending orders, constructed by company policy to be no more than 30 days in the future, are stored along with previously filled orders. This kind of relation is symmetric with retroactively bounded relations.

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 timestamp 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 because relations are append-only and items are entered in timestamp order. This is discussed in more detail in Section VIII. The process of recording degenerate relations is referred to as the *asynchronous method* [62].

A *mapping function* m for a relation R takes as argument an item e of a relation and returns a valid timestamp, as computed by using any of the attributes of e, excluding vt_e, but including the surrogate and transaction timestamp attributes. Sample mapping functions include $m_1(e) = tt_e^{\vdash} + \Delta t$ ("valid after a fixed delay"), $m_2(e) = \lfloor tt_e^{\vdash} - \Delta t \rfloor_{\text{hrs}}$ ("valid from the most recent hour"), and $m_3(e) = \lceil tt_e^{\vdash} \rceil_{\text{day}} + 8$ hrs. ("valid from the next closest 8:00 A.M.").

Definition: Temporal relation R is *determined with mapping function* m if the following is true:

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

Similarly, a relation is *undetermined* if such a function does not exist. For each of the undetermined specialized temporal relations defined already in this section, there exists a determined version. To illustrate, we consider three determined versions.

Definition: Temporal relation R is *retroactively determined with mapping function* m if the following is true:

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

Thus, a determined relation has a given type if its mapping function obeys the requirement of the type. For example, a relation is retroactively determined if each item is valid from the beginning of the most recent hour during which it was stored.

Definition: Temporal relation R is *predictively determined with mapping function* m if the following is true:

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

For example, a relation is predictively determined 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.

For further illustration, we present a bounded version.

Definition: Temporal relation R is *strongly retroactively bounded determined with mapping function* m *and bound* $\Delta t \geq 0$ if

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

The examples given previously were in fact bounded.

The generalization/specialization structure of the specialized temporal relations defined above is presented in Fig. 2. A relation type can be specialized into any of the successor

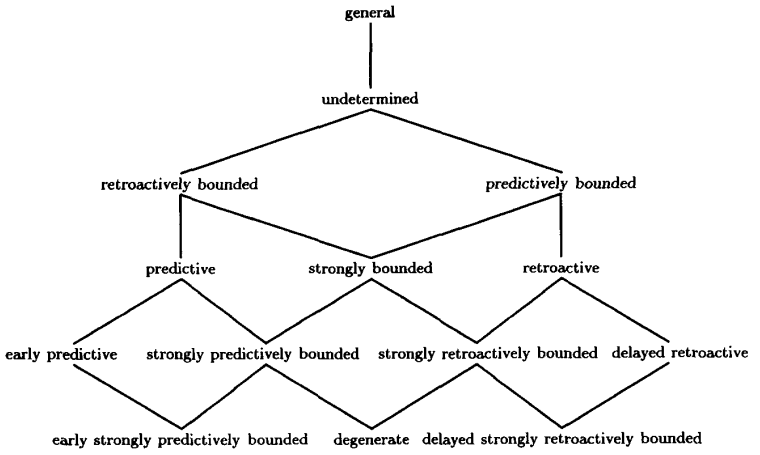


Fig. 2. Generalization/specialization structure of the event-based taxonomy.

relation types, and a relation type inherits all the properties of its predecessor relation types (as well as adding additional properties). For clarity, we have included only undetermined relation types; there exist determined counterpart of all the undetermined specialized temporal relations, e.g., strongly bounded determined.

The isolated event–based taxonomy is complete with certain assumptions. Note that the specializations in this section correspond to regions of the 2-D space spanned by transaction and valid time [30]. There are five assumptions. First, we are interested only in undetermined relationships. Second, we are interested only in regions bounded by lines parallel to the line $tt_e = vt_e$. This means that we do not wish to consider relationships that are dependent on absolute values of the timestamps, such as, say, the specialization that $vt_e \geq 2 \cdot tt_e$. Third, we consider only relative restrictions on the relationship between valid and transaction times. In combination with the previous assumptions, this implies that only three kinds of lines are of interest when describing restricted regions of the 2-D space, namely, lines parallel to $tt_e = vt_e$ for which (1) $vt_e > tt_e$, (2) $vt_e = tt_e$, or (3) $vt_e < tt_e$. Absolute bounds may be added later by the user of the taxonomy. Fourth, we consider only $\leq$-versions; pure $<$-versions and mixed versions may be obtained easily. Fifth, only connected regions are considered. Such resigns may be used as building blocks to form nonconnected regions. As a consequence of the assumptions, at most two lines are required for describing any possible region.

With zero lines, we can form no restrictions. Thus, we have a general temporal event relation. With one line, there are two distinct regions for each of the three line types, resulting in six distinct, specialized temporal event relations: early predictive and predictively bounded, predictive and retroactive, and retroactively bounded and delayed retroactive, respectively. With two lines, there are five possibilities corresponding to the combinations (using the numbering of the previous paragraph): (1) and (1) (early strongly predictively bounded), (1) and (2) (strongly predictively bounded), (1) and (3) (strongly bounded), (2) and (3) (strongly retroactively bounded), and (3) and (3) (delayed strong retroactively bounded). The result is a total of 11 types of specialized temporal relations, each of which is included in the taxonomy.

B. Interevent-Based Taxonomy

The previous definitions were based on predicates on individual event timestamped items. A relation schema had a given property if each individual item 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 event timestamped items in all possible extensions. We examine two aspects: orderings between items and regularity. In this and later sections, we continue to assume in the examples and explanations that tt_e is $tt_e^{\vdash}$. Recall that though the definitions are made on a per-relation ("global") basis, they may also be made on a per-partition basis with an arbitrary partitioning, e.g., the per-surrogate partitioning.

Definition: Temporal relation R is *globally sequential* if the the following condition[2] is true:

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

In globally sequential relations, each event must occur *and* be stored before the next event occurs or is (predictively) stored. Therefore, valid time can be approximated with transaction time, yielding an append-only relation that can support historical (as well as transaction time) queries. Such relations may be viewed as approximations to degenerate relations. As an example of the application of this property on a per-partition level, 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.

Definition: Temporal relation R is *globally nondecreasing* if the following is true:

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

[2] Alternatively, we could define sequentiality as follows. $\forall e \in R \quad \forall e' \in R \; ((e = e') \vee (\max(tt_e, vt_e) \leq \min(tt_{e'}, vt_{e'})) \vee (\min(tt_e, vt_e) \geq \max(tt_{e'}, vt_{e'})))$

In such a relation, items are entered in valid timestamp order. Sequentiality is generally a stronger property than nondecreasing. However, if the relation is degenerate, then the two properties are identical.

Definition: Temporal relation R is *globally nonincreasing* if

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

In such relations, as transaction time proceeds, we enter information that is valid further and further into the past. An example is an archeological relation that records information about progressively earlier periods uncovered as excavation proceeds.

Regularity—where transaction time, valid time, or both times occur in regular intervals—is often encountered in temporal relations.

Definition: Temporal relation R is *transaction time event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\forall e \in R\ \forall e' \in R\ \exists k_e^{e'}\ (tt_e = tt_{e'} + k_e^{e'}\Delta t).$$

Note that the transaction timestamps of successively stored items need not be evenly spaced; they are merely restricted to be separated by an integral multiple ($k_e^{e'}$) of a specified duration, Δt. An example is a periodic sampling of some physical variable such as temperature. The process of recording transaction time event regular relations is referred to as the *synchronous method* [62].

Definition: Temporal relation R is *valid time event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\forall e \in R\ \forall e' \in R\ \exists k_e^{e'}\ (vt_e = vt_{e'} + k_e^{e'}\Delta t).$$

The concept of *granularity* of valid timestamps can be expressed in terms of this property. For example, if the valid timestamp granularity is 1 s, then, equivalently, the relation is valid time event regular with the time unit 1 s.

Definition: Temporal relation R is *temporal event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\forall e \in R\ \forall e' \in R\ \exists k_e^{e'}\ (vt_e = vt_{e'} + k_e^{e'}\Delta t \wedge tt_e = tt_{e'} + k_e^{e'}\Delta t).$$

A periodic degenerate relation is trivially temporal event regular. Note that the same values of $k_e^{e'}$ must satisfy both transaction and valid time. Therefore, temporal event regular is more restrictive than both valid and transaction time event regular together. Next we define strict versions of the three different variants of regular specialized temporal relations.

Definition: Temporal relation R is *strict transaction time event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\begin{aligned}\forall e \in\ & R\ (\exists e' \in R\ (tt_{e'} = tt_e + \Delta t \\ & \wedge \neg\exists e'' \in R\ (tt_e < tt_{e''} < tt_{e'})) \\ & \vee \neg\exists e' \in R\ (tt_{e'} > tt_e)).\end{aligned}$$

Thus, either e' is the next item after e, or e is the last item stored.

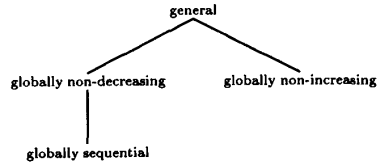


Fig. 3. Generalization/specialization structure of the interevent-based taxonomy. Part I: Orderings.

Definition: Temporal relation R is *strict valid time event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\begin{aligned}\forall e \in\ & R\ (\exists e' \in R\ (vt_{e'} = vt_e + \Delta t \\ & \wedge \neg\exists e'' \in R - \{e, e'\}(vt_e \leq vt_{e''} \leq vt_{e'})) \\ & \vee \neg\exists e' \in R\ (vt_{e'} > vt_e)).\end{aligned}$$

This definition is slightly more complicated than the previous one, because we want to disallow items with identical valid items (which is already impossible with transaction time).

Definition: Temporal relation R is *strict temporal event regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\begin{aligned}\forall e \in R\ ((\exists e' \in R\ (& tt_{e'} = tt_e + \Delta t \wedge vt_{e'} = vt_e + \Delta t \\ & \wedge \neg\exists e'' \in R\ (tt_e < tt_{e''} < tt_{e'}) \\ & \wedge \neg\exists e'' \in R - \{e, e'\}(vt_e \leq vt_{e''} \leq vt_{e'}))) \\ & \vee (\neg\exists e' \in R\ (tt_{e'} > tt_e) \wedge \neg\exists e' \in R\ (vt_{e'} > vt_e))).\end{aligned}$$

Although somewhat complex, this definition is just the combination of the two previous definitions using the same time unit for both valid and transaction time.

Note that if relation R' is transaction time event regular with time unit Δt_1 and valid time event regular with time unit Δt_2, then R' is also temporal event regular, with the temporal time unit Δt_3 being some common divisor of Δt_1 and Δt_2. Thus, if $\Delta t_1 = 28$ s and $\Delta t_2 = 6$ s, then $\Delta t_3 = 2$ s (largest common divisor). For the strict case, however, valid and transaction time event regulariy does not imply temporal event regularity.

Analogously with the ordering properties, the above regularity properties can be defined in a global or per-partition fashion. However, the nonstrict versions have the additional property (not shared with ordering and strictness) that the per-partition variant implies the global variant. Note that regularity is a different property than *periodicity*, which encodes facts such as something is true from 2 P.M. to 4 P.M. during weekdays [39].

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

The generalization/specialization structures for the temporal relations defined in this section are outlined in Figs. 3 and 4. The two structures are orthogonal.

C. Taxonomy on Isolated Intervals

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

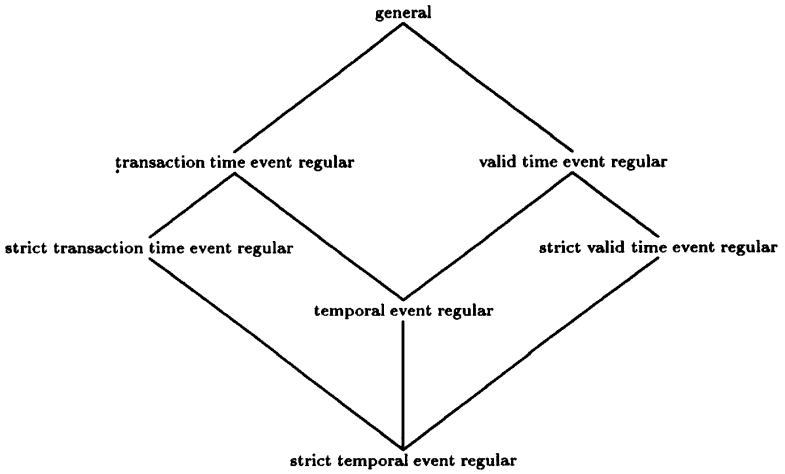


Fig. 4. Generalization/specialization structure of the interevent-based taxonomy. Part II: Regularity.

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. Note again that these properties concern durations rather than starting events, and that they can be calendar-specific, e.g., one month.

Definition: Temporal relation R is *transaction time interval regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\forall e \in R\ \exists k_e (tt_e^{\dashv} = tt_e^{\vdash} + k_e \Delta t).$$

Definition: Temporal relation R is *valid time interval regular with time unit* $\Delta t \geq 0$ if the following is true:

$$\forall e \in R\ \exists k_e (vt_e^{\dashv} = vt_e^{\vdash} + k_e \Delta t).$$

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 and terminations that observes a company policy that all such hires and terminations be effective on either the first or the fifteenth of each month.

Definition: Temporal relation R is *temporal interval regular with time unit* Δt if the following is true:

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

Hence, the time unit must be identical for both transaction and valid time.

The situations where all intervals have the same length are interesting special cases of the above definitions with k_e, k_e^1, and k_e^2 equal to 1. We term these special cases *strict transaction time interval regular, strict valid time interval regular*, and *strict temporal interval regular.*

Recall that the concept of regularity may be applied to relations on a per-partition basis as well as globally (as discussed at the beginning of this section).

The specializations in the previous section concern event relations, and the specializations in this section concern interval relations; they are quite different. However, the generalization/specialization structure of the specializations in this section is identical to that of the previous section, as illustrated in Fig. 4, with the exception that "event" is replaced by "interval."

D. Interinterval-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, but they may be applied on a per-partition basis as well. Many of these same terms also apply to event relations, and where defined in Section III-B. Context should differentiate these uses.

Definition; Temporal relation R is *globally sequential* if the following is true:

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

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 of the next week is recorded during the weekend, then this relation will be per-surrogate sequential.

A relation is nondecreasing if items are entered in valid timestamp order, and it is nonincreasing if items are entered in reverse valid timestamp order.

Definition: Temporal relation R is *globally nondecreasing* if

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

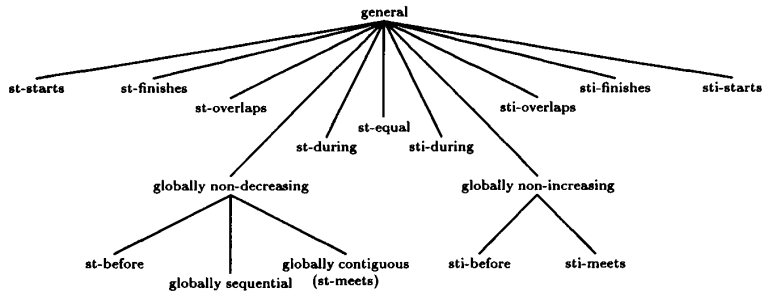


Fig. 5. Generalization/specialization structure of the interinterval-based taxonomy.

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

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

Definition: Temporal relation R is *globally nonincreasing* if

$$\forall e \in R \quad \forall e' \in R \; (tt_e < tt_{e'} \Rightarrow vt^{\vdash}_{e'} \leq vt^{\dashv}_{e}).$$

Definition: Temporal relation R is *globally contiguous* if the following is true:

$$\begin{aligned} \forall e \in R \; (\exists e' \in R - \{e\} (vt^{\dashv}_{e} = vt^{\vdash}_{e'} \wedge tt_e < tt_{e'} \\ \wedge \neg \exists e'' \in R - \{e, e'\} \\ (tt_e < tt_{e''} < tt_{e'})) \\ \vee \forall e' \in R - \{e\} (vt^{\vdash}_{e} \geq vt^{\dashv}_{e'})). \end{aligned}$$

This definition states that in a globally 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 is given in Section III-F.

Allen has demonstrated that there exist a total of 13 possible relationships between two intervals [7]. These relationships may be denoted *before, meets, overlaps, during, starts, finishes, equal*, and the inverse relationships for all but *equal*, e.g., *inverse before* and *inverse finishes*. For each such relationship, **X**, we can define a property *successive transaction time* **X** that requies that items, successive in transaction time, are related by **X**. For example, the property *successive transaction time overlaps* requires that intervals that are adjacent in transaction time overlap in valid time, ensuring that the next item began before the previous one completed.

Definition: Temporal relation R is *successive transaction time* **X** if the following is true:

$$\begin{aligned} \forall e \in R \; (\exists e' \in R - \{e\} (vt_e \mathbf{X} vt_{e'} \wedge tt_e < tt_{e'} \\ \wedge \neg \exists e'' \in R - \{e, e'\} (tt_e < tt_{e''} < tt_{e'})) \\ \vee \forall e' \in R - \{e\} \; (tt_e \geq tt_{e'})). \end{aligned}$$

Of these, the most interesting is *successive transaction time meets*, which is defined above as *globally contiguous*.

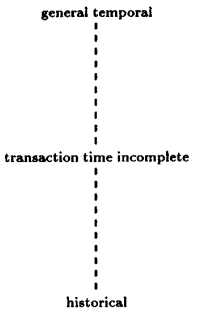


Fig. 6. Generalization/specialization structure of transaction time incomplete temporal relations.

Fig. 5 illustrates the specilization/generalization structure for the properties discussed above. In this figure, *successive transaction time* is abbreviated 'st-', and *successive transaction time inverse* is abbreviated 'sti-'.

E. Transaction Time Incompleteness

There is one type of restriction, orthogonal to the previously mentioned restrictions, 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 is transaction time incomplete if some previous historical states are missing. At one end of the spectrum of incompleteness, we find a historical relation (i.e., only the current historical relation is recorded). At the other end, we have a complete temporal relation where all historical relations that were current at some point are retained. In between, many options exist. Such options include storing every nth historical state, saving the historical state at periodic intervals (yielding a transaction time event regular relation), and saving the historical state at arbitrary, manually specified transaction times.

The specialization/generalization structure of transaction time incomplete temporal relations is shown in Fig. 6, where the dashed lines indicate intermediate, incomplete relations.

F. 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 [25], [32] and derive

$$\left\{ \begin{array}{c} \text{per item} \\ \text{inter-item} \end{array} \right\} \times \left\{ \begin{array}{c} \text{simple} \\ \text{parameterized} \end{array} \right\}$$

Fig. 7. Four types of specializations on temporal relations.

an interval relation by interpreting each event as ending an interval started by the previous event (on a global or per-partition 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-partition) sequential, then the derived interval relation will be globally (per-partition) sequential. The same holds for globally/per-partition 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-partition) contiguous. Hence, our example interval relation will be per-partition ordered, sequential, and 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 item are related to the starting and ending time of other items by an integral multiple of a duration, Δt.

G. Interrelations Between Per-Relation and Per-Partition Specializations

We now consider the interrelations of specializations when applied on a per-relation basis, on the one hand, and when applied on a per-partition basis, on the other.

For a specialization (e.g., retroactively bounded with bound Δt) on a relation on hold on a *per-relation* basis, the set of all items in the relation must satisfy the specialization. In order for a specialization to hold on a *per-partition* basis, for some given partitioning (e.g., per-surrogate), the specialization must be satisfied in turn by the set of items of each partition of the partitioning.

We proceed by dividing specializations into four categories, as shown in Fig. 7. A specialization is *per-item* if it applies to individual items in isolation (cf. Sections III-A and III-C); otherwise, it is *interitem* (cf. Sections III-D and III-E). Orthogonally, specializations can be *simple*, e.g., "retroactive," or they can be parameterized. For example, "retroactively bounded with bound Δt" is parameterized with parameter Δt.

Let us assume that a relation schema in turn satisfies each of the four types of specializations on a per-relation basis. Then we consider how to characterize the relation schema on a per-partition basis. Let R be a sample extension of the relation schema. If R satisfies a *per-item, simple* specialization on a per-relation basis, then R also satisfies that specialization on a per-partition basis. This observation, and each of the observations in the following, is true for any partitioning and any specialization. For example, if R is retroactive per relation, then R is also retroactive on a per-surrogate basis. Let an arbitrarily chosen partitioning be given that divides R into k partitions. If R satisfies a *per-item, parameterized* specialization with parameter x, then R satisfies that specialization on a per-partition basis with parameters $x_1, x_2, \cdots, x_k$, where each of the x_i are at least as restrictive as x. For example, if R is retroactively bounded with bound Δt per relation, then there exists tighter bounds $\Delta t_1, \Delta t_2, \cdots, \Delta t_k$, so that R is retroactively bounded with these bounds per surrogate.

We now assume that a relation R satisfies specializations of the four types on a per-partition basis for some given, arbitrarily chosen partitioning that divides R into k partitions. Again, the particular specialization may be chosen arbitrarily. If R satisfies a *per-item, simple* specialization per partition, then R also satisfies that property on a per-relation basis. If R satisfies, on a per-partition basis, a *per-item, parameterized* specialization with parameters $x_1, x_2, \cdots, x_k$, then R also satisfies the specialization on a per-relation basis, and the parameter x is equal to the least restrictive parameter among the x_i.

In the remaining four cases where we consider interitem specializations instead of per-item specializations, no general statements may be made.

H. Summary

We have presented an extensive taxonomy of specialized properties of temporal relations. The practical relevance of the definitions is emphasized by examples. The properties apply to either event or interval temporal relations. A relation may have specialized per-item properties (Sections III-A and III-C) as well as specialized interitem properties (Sections III-B and III-D). A relation may also be transaction time incomplete (Section III-E). All three types of properties may be applied on either a per-relation or on a per-partition basis. Partitionings may be chosen arbitrarily, but the most important partitioning is the per-object surrogate partitioning.

We described how an event relation may be naturally interpreted as an interval relation, and we discussed how the event properties would transform into corresponding interval properties. Additionally, we described how per-item properties, simple as well as parameterized, when satisfied on a per-relation basis, would essentially be satisfied on a per-partition basis and, conversely, independently of the particular partitioning.

IV. Classification of Existing Temporal Data Models

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

A. General Temporal Relations

General temporal relations are found in only a few data models [8], [53]. The snapshot mechanism [6] may be ex-

tended to support general temporal relations. A *snapshot* of a relation is an independent copy of a 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 [5], [36]. The snapshot mechanism may be applied to a relation in three ways [2]–[4]. 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").

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

B. Retroactive Temporal Relations

Gadia presents a multidimensional temporal data model that is in turn restricted to a 2-D data model with valid and transaction time as the dimensions [18]. 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, "As of now, Dr. Iovanni is hired as an assistant professor from September 1, 1991, until 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 a (retroactive) historical data model where event timestamps are used for the modeling of real-world activity [17].

Sarda proposes another specialized temporal data model in which current facts may be appended and where so-called retrospective updates (change to information about the past) are possible [45]. Hence, the transaction time is always equal to or after the valid time, and, like the previous model, this model supports retroactive temporal relations.

C. Strongly Retroactively Bounded Relations

In real-time databases, transactions have hard or firm real-time deadlines [1]. If the deadline passes before the transaction is executed, the transaction is unscheduled [41]. Hence, the transaction time of information read 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, which must be known if it can be ensured that deadlines will always be met [41].

D. Degenerate Temporal Relations

Relations representing time sequences and time sequence collections of the TSC model [42], [47], [49], [51] 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 time sequence collections [48] is a per-surrogate contiguous relation that is also per-surrogate sequential.

The POSTGRES data model [43], [60] 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 [59] supports rollback (viewing the time dimension as transaction time) and historical time-slice (viewing the time dimension 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 [25]. Thus, all updates are physically append-only. Only event timestamps are possible, and they are unique, increasing, and system-supplied. Additionally, the assumption is made that time-varying attributes have stepwise constant semantics [27], [28]. As a result, the model is appropriate only for modeling the history of the update activity of the database. However, because it allows for irregular timestamps reflecting real time, it may be used as a temporal data model when the transaction and valid times of items coincide; hence, it is also a degenerate temporal model. Similarly, successive snapshots of a conventional relation (a movie) produce a degenerate temporal relation [4].

In the Applicative Data Model [24], changes cannot be made to data that have already been stored; hence, an applicative historical relation is a degenerate temporal relation.

Adiba introduced an append-only relation that may be modified using special error-correcting operations [3]. 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 cannot change or record future events.

Finally, a variety of data formats are available for time series analysis [12]. Some are degenerate, some are transaction time event regular, and most are globally ordered.

E. 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 roll back 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 in time which rollbacks will be needed later.

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

F. Summary

We have demonstrated how the taxonomy of specialized temporal relations may be used for characterizing previously

proposed time-oriented data models. We showed how many of the previously proposed data models that incorporate only one time dimension may be viewed as specialized temporal relations over both valid and transaction time. Interestingly, no one, to our knowledge, has studied the predictive, determined, early, or delayed variants, even though situations exist where such specialized temporal relations are useful.

V. Generalized Temporal Relations

To this point, we have considered individual temporal relations in isolation. We have focused on temporal specialization, considering the restrictions that may be placed on the valid and transaction timestamps of a temporal relation, thereby coupling the two timestamps in some fashion. Now we change perspective and consider temporal relations as part of larger application systems where items move between multiple temporal relations. We investigate *temporal generalization*, which involves decoupling timestamps. In the next section, we describe two example applications that use specialized and generalized temporal relations.

The concepts of specialization and generalization have been used previously within data modeling. A subclass may be created from a class by means of specialization, i.e., by making the defining properties (the intension) of the class more restrictive, and thus also restricting the set of examples (the extension) of the class. As the dual, 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 [15], [23], [52].

Temporal specialization and generalization are also duals. As we have seen, specialization contracts the space of possible interrelations of timestamps. Temporal generalization appears in at least three guises, each of which expands the space of possible interrelations of timestamps. The first is removing restrictions. For example, an early strongly predictively bounded relation may be generalized to a strongly predictively bounded relation, which may be generalized to a predictively bounded relation, which may be generalized to a general temporal relation. Specialization involves moving down the lattices given in Section III, thereby contracting the 2-D space of possible interrelations; generalization involves moving up these lattices, expanding the space of possible interrelations.

A second way to define a generalized temporal relation is to simply add completely new, orthogonal time dimensions. In systems where items flow between multiple temporal relations, items may accumulate timestamps by keeping their previous timestamps and gaining new timestamps as they are entered into new temporal relations. Consequently, an item in a generalized temporal relation has several kinds of timestamps: a valid timestamp, which records when the item was true in reality, a *primary* transaction timestamp, which records when the item was stored in this relation, one or more *inherited* transaction timestamps, which record when the time was stored in previous relations, and one or more TSG-*generated* timestamps that record when the item was manipulated elsewhere in the system.

A third, more involved, means of defining generalized relations is to have derived relations inherit transaction timestamps from their underlying relations. For example, consider process control in a chemical manufacturing plant. Values from temperature and pressure sensors may be stored in temporal relations. The sensed data may later be processed further to derive new data, such as the rate at which the reaction appears to be progressing [41]. This derivation typically would depend on past temperature and pressure trends. Stored in the derived temporal relation recording the reaction rate would be the transaction time at which the rate was recorded, along with one or more inherited transaction times, specifying when the underlying data, the temperature, and pressure readings were originally recorded, thereby providing an indication of the relevance of the calculated rate.

Specialization may be applied to any two time dimensions. Consequently, standard 2-D temporal relations may be perceived as multidimensional generalized temporal relations in which the values of the additional time dimensions are specialized to be identical to those of the standard transaction time dimension.

Transaction timestamps added to a relation by temporal generalization are generated elsewhere in the system by timestamp generators. During the design of a generalized temporal relation, the sources of these additional timestamps must be identified; this is done by specifying a *system topology*.

A system may have two kinds of passive components, *temporal relations* and *buffers*. Temporal relations, already described in detail, contain temporal data that may be updated and queried from outside the system. Buffers are internal data stores that are not seen from outside the system.

A system may contain several kinds of active components. Data is received by either *sensors* or *processors*. In a monitoring scenario, data are recorded by sensors that observe a portion of the real world being modeled. In a manual scenario, data are received by a processor, as a result of either data input by a user or data retrieved from an on-line source. Data may be timestamped. A *timestamp generator* (TSG) is a mechanism that returns timestamp values on demand. Manually supplied data may or may not contain a valid timestamp. If not, the receiving processor must append a valid timestamp, obtained upon request from an available TSG. A processor responsible for entering data into a temporal relation also uses a TSG for transaction timestamping.

A sensor, s_i, forms events (observations). To do so, it sends requests, r_{ik}, to a TSG and receives timestamps, t_{ik}. The sensor then appends t_{ik} to the remaining part of the event and passes the result on.

A processor may request a timestamp from a TSG whenever it performs an action. For example, a processor may request a timestamp when it stores data in a buffer or when it retrieves data from a buffer, and the timestamp may be made part of the data that is stored. These are TSG-generated time dimensions. A timestamp added by a processor when it stores an item in a relation is a transaction timestamp.

Finally, in a system, components of the type mentioned above may be connected via *data channels* with no storage capacity. Connections may be specified between a processor

and a relation, indicating that the processor may read or write data in that relation, between a processor and a buffer, indicating that the processor inserts and deletes data in the buffer, and between a processor (or sensor) and a TSG, indicating that the processor (or sensor) may obtain timestamps from the TSG. The system topology does not specify the order in which data are sent along data channels, though specializations of relations may imply a certain data ordering. In Section VII, we discuss the possible ways of interconnecting temporal relations.

The system topology is used in this paper only for specifying the source of inherited timestamp attributes. We do not differentiate between logically centralized, distributed, heterogeneous, federated or multidatabases [37], [50], [61]. The system description used here should be applicable in varying degrees to all of these systems.

We emphasize that generalization expands the space of allowable timestamp values; specialization, which can be applied to a single timestamp attribute (e.g., a particular timestamp attribute may be event regular) or to a pair of timestamp attributes (e.g., degeneracy specifies that both values are identical), contracts the space of allowable timestamp values. It is in this sense that generalization and specialization are exact duals of each other.

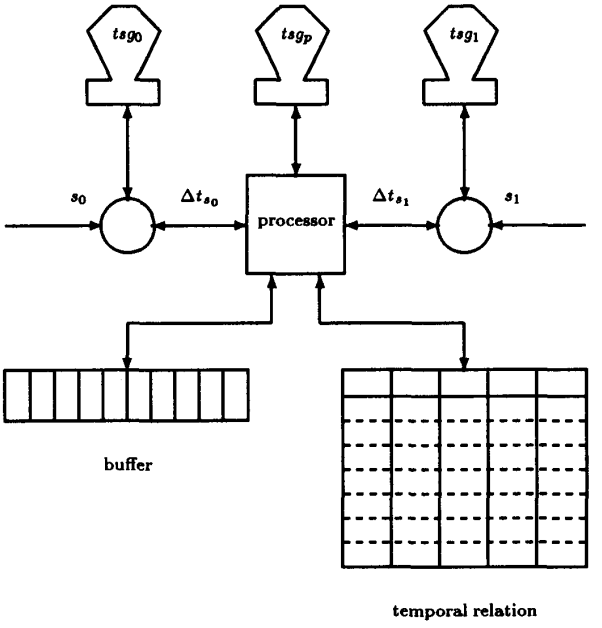


Fig. 8. Buffering of temporal data.

VI. Example Applications

We first discuss a very simple application with a generalized temporal relation containing one inherited transaction timestamp. We then examine a more involved application containing several generalized relations.

A. Single Generalized Relation

The system, illustrated in Fig. 8, employs two sensors, s_0 and s_1, to collect temperature data as the temperature in a chemical experiment varies over time. Temperature values are timestamped with the current time when they arrive at a sensor. The timestamps are obtained from the timestamp generator tsg_0 and tsg_1. The valid timestamps of the measurement are assumed to be identical to these sensor timestamps. At the sensors, the measurements are also stamped with sensor identifiers. The sensors have no storage capacity; the items are simply passed on to the processor, which places them in the buffer. The buffer retains items for periods of time before they are transaction timestamped (using timestamps obtained from tsg_P) and entered into the relation. The relation thus contains three timestamps, the valid timestamp, the (primary) transaction timestamp (from tsg_P), and the sensor timestamp (from tsg_i).

The temporal relation is both specialized and generalized. It is specialized to a degenerate relation with respect to the valid and the sensor timestamps, which are identical; indeed, only one needs to be stored. It is generalized because two transaction timestamps are recorded in the relation.

Because of the varying maximum delays of items from the two sensors Δt_{s_0} and Δt_{s_1}, it may be the case that an item with a later valid time arrives before that of an item with an early valid time. This implies that the items arriving at the processor from the sensors are unordered in valid time. By delaying items at the buffer, we can ensure that the relation is ordered. Of course, that destroys the relationship between the sensor timestamp and the transaction timestamp, which is why the sensor timestamp must be included in the relation. The buffer allows us to characterize the relation as globally nondecreasing and as delayed retroactive, with a bound computable from the various delays in the system.

B. Multiple Generalized Relations

We now present a fairly complex application system that illustrates many types of specialized temporal relations as well as multiple transaction times resulting from temporal generalization. The system contains a collection of temporal relations maintained by the transportation department of a state government. Its topology is shown in Fig. 9. An *employee relation* is maintained on the workstation of each manger in this department, recording schedules, budgets, and salary levels for the employees under that manager. For the entire department, a single *personnel relation* is maintained on the administrative computer under the data processing group, which also maintains a *payroll relation*. The state's accounting office maintains a *financial relation*. The bank, responsible for salary payments, maintains an *accounts relation*. Finally, there are two log relations that will be discussed shortly.

Eric was hired by LeeAnn at a salary of $2000 per month. Because of a long and fractious session of the legislature, salary levels could not be agreed upon until well into the fiscal year. In mid-March, the state government finalized the budget. LeeAnn decided that Eric would receive a raise of $300 per month, effective retroactively to March 1 and to be paid to Eric on the first of the subsequent month. LeeAnn's secretary

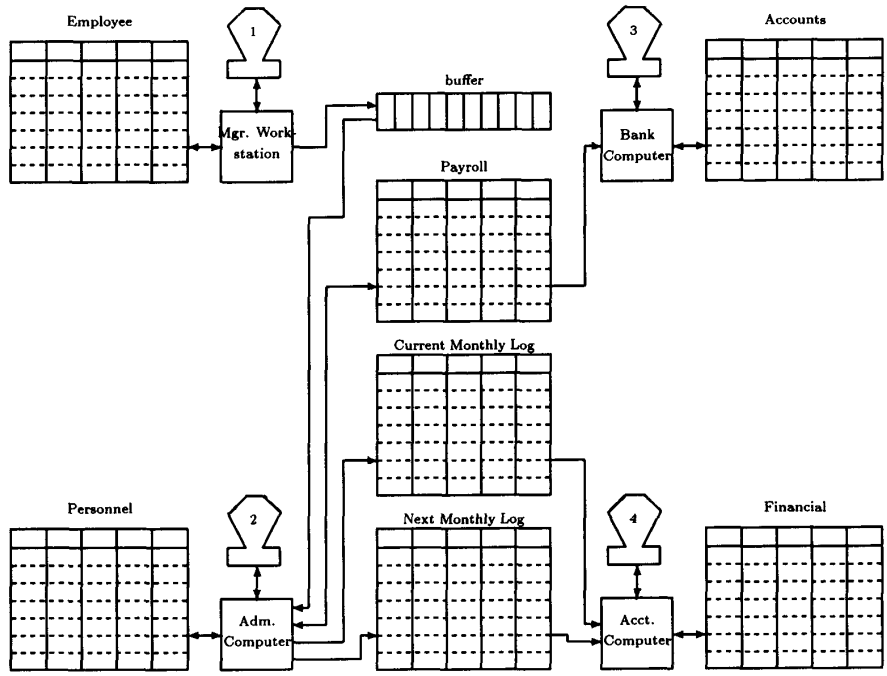


Fig. 9. System topology for the second example.

entered this information into the employee relation, which then contained the following item.

Employee:	*name*	*salary*	*vt*	*tt*
	Eric	$2300	Mar 1	Mar 26

The valid time is manually supplied by the manager; the transaction time is automatically recorded by the workstation, which requests a time from tsg_1. Because current, retroactive, and postactive updates are possible, this relation may be classified as *general*.

Once a week a batch job runs on the administrative computer to upload changes made on the managers' workstations. The job creates an update file, which is applied to the personnel relation one day later. This job ran on April 1, resulting in the following item being entered into the personnel database on April 2.

Personnel:	*name*	*salary*	*vt*	*tt*	tt_1
	Eric	$2300	Mar 1	Apr 2	Mar 26

The transaction timestamp, tt, records when the batch job executed the transaction recording this item; it is supplied by tsg_2. The inherited transaction timestamp, tt_1, records the transaction time of the information in the manager's workstation; it is copied from the transaction time attribute stored there. This is an example of a generalized temporal relation, with one primary transaction time and one inherited transaction time. The personnel relation is also specialized in the interaction between tt, supplied by tsg_2, and tt_1. Thus, tt_1 precedes tt by at least a day (the delay in processing the updated file) and by at most eight days, because an update may reside in a manager's relation for a week before being uploaded, followed by the one-day processing delay. Hence, the pair tt and tt_1 in this relation is *delayed strongly retroactively bounded with a delay of one to eight days*. Concerning vt and tt, the relation is *general*; it is also *general* concerning vt and tt_1.

The data processing group is responsible, in part, for producing paychecks. It does so by creating a tape that is taken to the bank. The bank requires that such tapes be received at least two days before the paychecks are to be issued; company policy dictates an additional day for safety. On March 29, the payroll relation, which will be copied to tape, contains the following item.

Payroll:	*name*	*salary*	*vt*	*tt*
	Eric	$2000	Apr 1	Mar 29

The date on which the check is to be issued, April 1, is the valid time. Note that Eric's March salary is actually $2300. However, this fact did not make it into the personnel relation until April 2 (tt in the personnel relation). On April 28, the payroll relation contains this item.

Payroll:	*name*	*salary*	*vt*	*tt*
	Eric	$2600	May 1	Apr 28

This amount consists of the monthly salary for April, $2300, plus an additional $300 that was omitted from the March check. We shall see shortly how this compensating payment is handled in the financial database.

In the payroll relation, vt will always precede tt by exactly three days, so this relation may be specialized to *predictively determined by three days*. It is also *temporal event regular with an interval of one month*, because both the transaction

timestamps and the valid timestamps differ by multiples of one month.

The payroll tape is cut using information from this relation, and is then carried to the bank, where it is processed sometime during the next two days to ensure that the amount gets credited on time. When it does get recorded in the database, the information is associated with a transaction time from tsg_3. In this case, March's paycheck was processed on March 30, and April's on April 29, resulting in the following relation.

accounts:	*name*	*credit*	*vt*	*tt*
	Eric	$2000	Apr 1	Mar 30
	Eric	$2600	May 1	Apr 29

Because the valid time will follow the transaction time by one to two days, the relation may be specialized to *early strongly predictively bounded by one to two days.*

For each update transaction, the administrative computer appends an item to the appropriate *monthly log*, depending on the valid time of the transaction. Company policy restricts transactions to no more than one month postactive, implying that only two logs are ever active: the current monthly log and the next month's log. When the retroactive salary increase (initially entered into LeeAnn's workstation on March 26) was processed by the administrative computer on April 2, a compensating transaction resulted in the following item being inserted into the next month's log (May).

May's monthly log:	*name*	*salary*	*vt*	*tt*
	Eric	$300	May 1	Apr 2

Subsequently, when Eric's payroll check for April was issued on April 28, the following item was inserted into the next month's log (May).

May's monthly log:	*name*	*salary*	*vt*	*tt*
	Eric	$2300	May 1	Apr 28

Because retroactive changes are possible as far back as one month (e.g., a transaction valid on April 1 being inserted into April's log on April 30), and postactive changes are possible as far into the future as two months (e.g., a transaction valid on April 30 being inserted into April's log on March 1), each of the monthly logs can be specialized to *strongly bounded between minus one month and plus two months.*

The state's accounting office uploads the log of a month shortly after that month ends. It then processes the items contained in the log, performing internal audits. Errors detected at that point may simply be corrected in the copy of the monthly log held in the accounting department's computer, or they may necessitate compensating transactions, depending on the specific error. Once the monthly log is cleaned up, it is applied to the financial database on the fifteenth of the month, a process termed "closing off the month" [62]. The financial relation will contain the following items after both April and May have been closed off.

Financial:	*name*	*salary*	*vt*	*tt*	tt_2
	Eric	$2000	Apr 1	May 15	Mar 29
	Eric	$300	May 1	Jun 15	Apr 2
	Eric	$2300	May 1	Jun 15	Apr 28

The transaction time, tt, when the entry is recorded in the financial relation is obtained from tsg_4, and the inherited transaction time, tt_2, is the transaction time of the original entry recorded in the monthly log, supplied by tsg_2.

Because updates to the financial relation occur only on the fifteenth of each month, this relation may be specialized to *transaction time regular over one month.* Also, tt always follows tt_2 by 16 days (e.g., tt = May 15 and tt_2 = April 30) to 76 days (e.g., tt = June 15 and tt_2 = April 1), and so the interrelation between the primary and the secondary transaction time may be characterized as *delayed strongly retroactively bounded with bound 16 to 76 days.* As discussed above in the context of the monthly logs, vt must be no more than one month prior to tt_2. Thus, the interrelation between vt and tt_2 is restricted, as before, to *strongly bounded between minus one month and plus two months.* These two relationships imply that the interrelation between vt and tt may be described as *delayed strongly retroactively bounded with bounded 16 to 46 days*, with the former bound being exemplified by vt = April 30 and tt = May 15, and with the latter bound being exemplified by vt = May 1 and tt = June 15. Finally, because each month is closed off during the next month, the relation is *globally nondecreasing* and *transaction time event regular with an interval of one month.*

The implications of the process of closing off on the temporal semantics of accounting databases were first examined by Thompson [62]. In his terminology, the tt of the payroll and financial relations is *physical time* (i.e., it is tied to a TSG and concerns the storage of data), vt of the payroll relation is *logical time* (i.e., it links the event with the value of a TSG present when the event occurred), vt of the financial relation is *accounting time* (i.e., it has been validated by the closeout process), and tt_2 of the financial relation and (equivalently) tt of the monthly log is *engineering time* (i.e., it is always up-to-date, but not necessarily consistent, because it has not yet been validated). This example illustrates how the application of specialization and generalization can accommodate Thompson's conceptual taxonomy of discrete clocks.

VII. Querying Generalized Temporal Relations

In previous sections, we explored how items may flow from one temporal relation to another in a system containing multiple temporal relations. In particular, we showed how it is possible for items to preserve primary transaction time attributes from predecessor relations. Appending a new transaction time attribute every time it is entered into a relation results in generalized temporal relations with multiple transaction time attributes. Preserving predecessor transaction timestamp attributes allows one to query the predecessor relation from the current relation.

The successor relation may be preferable to query for several reasons. Because of security controls, the predecessor relation may not be available, because it exists on a different processor for which direct access is not possible, or because it has since been deleted. Alternatively, the predecessor relation may be available, but the successor relation may have indexes

defined on it or may be clustered in such a manner that access to it for certain queries is more efficient.

For example, the (centralized) personnel relation of the application system discussed in the previous section inherits the transaction and valid time attributes of the employee relations local to the managers' workstations. Therefore, it is possible to query the employee relations from the personnel relation. Members of the data processing group, having access to only the administrative computer, can tell, say, when LeeAnn's secretary made a particular salary adjustment for an employee on the manager workstation.

Even though the primary transaction time attribute (and, naturally, the valid time attribute) from the predecessor temporal relation is present in the successor temporal relation, not all of the items present at a particular time in the predecessor relation may be present in the successor relation at the same time, for two reasons. First, there is likely to exist a transmission delay between the two relations. The delay from when an item is stored in the predecessor relation to when the item is stored in the successor relation may be significant. In the previous section, we saw that the delay between employee relations and the personnel relation may be up to eight days. Second, it may be that only a portion of the items entered into the predecessor relation are transmitted to the successor relation. For example, if LeeAnn has hired employees directly (as opposed to employees hired departmentally, such as Eric), the items recording salaries for those employees will never appear in the personnel relation.

In consequence, despite the fact that we have the capability of querying the predecessor relation remotely from the successor relation, the set of queries that can be answered correctly at the successor relation is a subset of the queries that can be answered correctly when querying the predecessor relation directly. For example, querying the personnel relation on March 28 regarding the current salary of Eric present in LeeAnn's employee relation will give the (incorrect) answer \$2000. The same query applied directly to LeeAnn's employee relation will give the correct answer of \$2300. However, the query of what was Eric's salary two weeks prior to March 28 will yield the correct result from either relation.

In addition to missing items, it may be that not all the time-varying attributes of an item present in an predecessor relation are included when items are transmitted to a successor relation. For example, the employee relation could contain a title attribute in addition to the salary attribute. For simplicity, we do not consider the possibility of partial transmittal further. We also do not consider here derived relations containing inherited transaction timestamps.

Below we discuss an approach that avoids the problem of the same query having differing (and thus inconsistent) answers depending on *where* it is asked. The fundamentally same approach was previously used to solve the similar problem of identical queries having different results depending on *when* they were asked [26]. This problem surfaces with temporal and rollback relations when flexible physical deletion is allowed.

In essence, the approach is to simply disallow all queries from a successor relation on a predecessor relation if the result of that query cannot be intensionally decided to be identical to the result of the corresponding query when asked locally at the predecessor relation.

For each ordered pair of a predecessor relation R_p and a successor relation R_s, where R_s has inherited the primary transaction time attribute from R_p, we define a *transmission filter*, $\mathcal{T}(R_p, R_s)$. This filter intensionally expresses which of the items currently present in R_p is currently present in R_s. For example, if only departmental employees appear in the employee relation, the filter between LeeAnn's employee relation and the personnel relation, stating that queries about the most recent eight days are disallowed, may be expressed as $\sigma_{tt \leq now - 8\,\text{days}}$. The transmission filters are defined by the designer of the overall application system.

When a query Q on R_p is asked at R_s, the following takes place.

1) The query Q is modified with the transmission filter expression, $\mathcal{T}(R_p, R_s)$, and the modified query expression $M_{\mathcal{T}(R_p,R_s)}(Q)$ is obtained.
2) Q and $M_{\mathcal{T}(R_p,R_s)}(Q)$ are tested for equivalence.
   a) If the test succeeds, then Q is processed.
   b) If the test fails, then the user is notified that the original query is disallowed and is presented with $M_{\mathcal{T}(R_p,R_s)}(Q)$. The user may submit this query for processing, modify the query and submit the result, or simply submit a completely new query. In the latter two cases, the new query will go through this cycle again.

For example, assume that the query $\pi_{\text{salary}}(\sigma_{\text{vt=Mar6} \wedge \text{tt=Mar28} \wedge \text{name=Eric}}(\text{employee}))$ is issued at the personnel relation on April 2. The fact that this query cannot be answered correctly from the personnel relation is discovered when the modified query, with the filter restriction added, is seen to be internally inconsistent, and thus is not equivalent to the original query. (With now = April 2, the expression in the transmission filter, $now - 8$ days evaluates to March 25, and because March 25 < March 28, the query is inconsistent.)

In considering each way of interconnecting temporal relations, we distinguish between three cases as outlined in Fig. 10, where processors have been omitted for simplicity. Only the interconnections between temporal relations where a successor inherits the primary transaction time attribute of the predecessor relation are of interest. In an application system, the three connection types may be applied together repeatedly, e.g., to specify a chain of relations, each inheriting data from the previous one.

The first case is the linear transmission of items from one relation (R_p) to another relation (R_s). Here all or just some of the items form R_p may be transmitted to R_s.

The second case involves the distribution of items. Here the items from one temporal relation may be distributed among an arbitrary number of relations. Note that the same item may be distributed to several relations and that the transmission filters, each between the predecessor and an individual successor relation, are thus independent. This kind of interconnection is absent from the application system in Section VI-B. If employees from several departments were

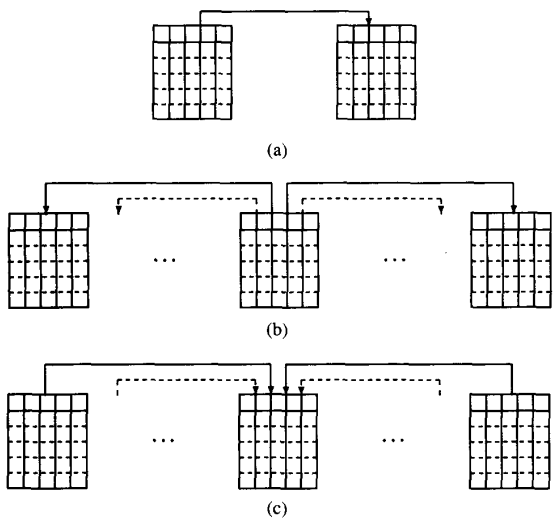


Fig. 10. Interconnections of temporal relations. (a) Linear transmission of items. (b) Distribution of items. (c) Collection of items.

managed by LeeAnn, items from her employee relation could be distributed among several personnel relations.

The third case is the collection of items where various items from multiple predecessor relations are transmitted to a single successor relation. With only the two previous cases, the predecessor relations that may be queried from some relations are all connected sequentially. When the third case is included, the relations that may be queried from some relation can be connected arbitrarily. In order to make the collection of items possible, we restrict all the immediate predecessor relations and successor relation to have the same schemes, with the exception that the successor relation has two additional attributes. First, the successor relation naturally has its own primary transaction time attribute. Second, it has an attribute associated with the transaction time attribute inherited from the set of predecessor relations that records from which predecessor items are received. The information of this second attribute, which partitions the successor relation with respect to the predecessor relations, is necessary in order to be able to query the predecessor relations from the successor relation. (This attribute may in fact be one of the attributes from the predecessor relations.) This kind of interconnection exists between the employee relations and the personnel relation. Here each manager records information about her own employees. Then the information recorded locally by each manager is collected in the central personnel relation.

Note that sources may take the role of originating temporal relations in the discussion above. Also note that buffers are irrelevant for the discussion above—buffers cannot be queried, and they only add additional delays and make the transmission filters between temporal relations more restrictive. Finally, note that in the above description, situations where a relation receives items that the relation itself transmitted are implicitly possible. For simplicity, we do not discuss such cycle.

In summary, three connection types are employed in specifying the system topology. For each inherited timestamp attribute, a transmission filter is specified that allows the database system to ensure that queries always yield correct results.

VIII. Implications for Query Optimization and Execution

In this section, we consider the performance implications for processing queries over specialized temporal relations. Specifically, we indicate how query processing algorithms and indexing techniques designed for 1-D time-varying data may be naturally extended to apply to specialized temporal (i.e., 2-D, valid and transaction time) relations as well. This represents a simple but significant contribution to the largely unexplored topic of efficient support of temporal data.

We proceed in two steps. First, we describe the general idea of applying 1-D approaches to 2-D 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 1-D 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. Instead, we discuss query optimization only to show that the taxonomy may be used to take known techniques that were heretofore limited to either rollback or historical databases (i.e., one time dimension) and apply them to specialized relations containing multiple time dimensions. New research efforts aimed directly toward the efficient support of temporal relations may also be designed to exploit the semantics of specialized temporal relations, with resulting performance gains.

A. Exploiting Specializations

The general idea can be stated as follows. In order to apply existing techniques previously used to improve the performance of queries on 1-D data, we use the specific interrelation between valid and transaction timestamps, guaranteed by the type of a specialized temporal relation, to simply disregard one time dimension and use only 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 items resulting from update activity arrive, by definition, in transaction timestamp order at temporal relations, we find it natural to use the transaction time dimension and ignore the valid time dimension.

This approach applies, with some variations, to all specialized temporal relations. The application of the approach to specialized temporal relations toward the bottom of a specialization/generalization structure (see Figs. 2 to 6), being closer to degenerate relations that never require more than one timestamp, will be more successful than the application to relations higher in the specialization/generalization structure. 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 items of a relation are physically clustered on transaction time on a per-relation basis (e.g., [29]) or on a per-object surrogate basis (e.g., [20], [44]). These are

straightforward representations, particularly if write-once storage media are used. Assuming physical clustering and strongly retroactively bounded temporal data, the following important property holds: All items with valid timestamps equal to some value, t_x, may be found within a limited number of items after the item with transaction time value equal to t_x, if it exists, and otherwise after the item with the largest transaction timestamp less than t_x. The limit on the number of items depends on the particular retroactive bound and the intensity of update activity for the relation.

The 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 1-D data (valid or transaction time) may prove applicable. Order preserving physical organizations for 1-D data seem especially promising, because order preservation in the transaction time dimension carries over the valid time dimension and results in items that are nearly clustered with respect to valid time.

B. Application to Previous Proposals

The issue of efficient temporal query processing is largely unexplored. Although much research is still needed, the efficient support of queries on data with a single time dimension of various kinds has been addressed to some extent. As indicated above, 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 consider two approaches to efficiently support various joints on 1-D temporal data. Leung and Muntz have proposed a stream processing approach to temporal (semi-) joins [35]. In this approach, the input to, and the output from, stream processors consists of a set of streams of items. A processor has a local state, and it is allowed to see only a single item 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 effects 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 semijoins). A stream join processor that assumes a time-ordered sequence of items may be converted into a processor 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 prestream 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.

A more traditional approach to the processing of 1-D temporal joins is chosen in most other work in temporal query optimization [19]–[21]. Most notably, a temporal event-join consisting of three time-oriented joins is considered [20]. 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 bookkeeping is necessary to process nearly ordered data, as opposed to the currently 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 1-D 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 attempt to ensure that storing previously current/valid data as well as current data should not adversely affect to a significant degree the performance of queries accessing only current data. The time-split B-tree [38], [44] 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 also has timestamps 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, Kolovson and Stonebraker generalize R-trees to span both magnetic and optical disk media, thus providing new intermediates between R-trees residing on only a single medium [33]. This is relevant when the bulk of temporal data do not fit on magnetic disk and must be migrated to optical disk [43]. They also introduce tactics aimed at improving observed deficiencies of existing indexing techniques for historical data (e.g., R-trees) [34]. This research may likely be extended to deal successfully with specialized temporal data.

The Time Index is an indexing technique based on the B-tree [14], [16]. It uses endpoints of intervals of validity for the indexing of items. How to extend this technique to cover specialized temporal data is an interesting topic.

Transaction time data may also be stored in backlogs clustered on the time dimension [28], [29]. On top of the backlogs, indexed and selectively cached views, together with differential (incremental and decremental) computation techniques, may be employed, together with standard query processing technique. The specialized temporal relations with close valid and transaction times may be easily integrated into, and efficiently supported by, this query processing and optimization framework.

In summary, it appears that many implementation techniques originally proposed for rollback or historical database and supporting only one kind of time may be adapted to also apply to specialized form of temporal relations supporting both kinds of time.

IX. Conclusion and Future Research

A temporal relation has two database system-interpreted time attributes, transaction time and valid time. A transaction timestamp is a simple value indicating when a fact is stored in the temporal relation. A valid timestamp records the validity of a fact, and it may be a simple value (event relation) or an ordered pair of simple values (interval relation). In general, these timestamps are independent, meaning that facts may be associated with a point or a pair of points in an unrestricted 2-D space. In many situations, however, the time points of facts are restricted to limited regions of this space, resulting in specialized temporal relations. Examples include process monitoring, satellite surveillance of crops or weather, accounting applications, and real-time databases. The restricted interrelations of timestamps constitute important semantics of temporal relation schemas.

In this paper, we considered the specialized semantics of the time attributes in generalized temporal relations. These include the standard temporal relation dimensions of valid time and (primary) transaction time, inherited transaction timestamps, and TSG-generated timestamps.

We presented an extensive taxonomy of temporal specializations, some restricting the stamps of individual facts, and others restricting the stamps of an interfact basis. The taxonomy provides a better understanding of the nature of individual temporal relations and of how various temporal data models compare. Additionally, a database system may be extended to exploit such time-related semantics of temporal relations if they are recorded in the schema. In particular, we showed that storage and indexing structures for 1-D temporal data may be naturally extended to efficiently support specialized temporal relations. The additional semantics may be used also for query optimization purposes, resulting in more efficient query processing.

We extended to 2-D space associated with facts to having n dimensions, resulting in generalized temporal relations. This natural extension resulted from considering temporal relations as parts of larger application systems, where facts were allowed to flow from relation to relation and thus accumulate timestamps. We presented a set of components that may be used to specify the topology of application systems, and we discussed the ability to query a predecessor relation via a successor relation. By means of examples, we illustrated how application systems are described and how specialization may be applied to any of the time dimensions in generalized temporal relations.

Further work is indicated in two areas. As we have shown (Section VIII), specialized temporal relations present an opportunity to optimize temporal queries; more work is needed to exploit specializations 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.

An overall approach to designing temporal databases is still needed. This paper has considered only half of the problem of designing temporal relations: determining the characteristics of the timestamp attributes that concern entire items. Just as important are the characteristics of the individual time-varying attributes. A fully articulated design methodology for temporal relations must address both timestamp attributes and time-varying attributes.

Acknowledgment

The topic of temporal specialization was introduced during discussions between the second author and A. Segev, whose comments encouraged further exploration of what was then termed "coupled relations." M. Stonebraker's use of valid time examples to illustrate rollback relations in POSTGRES (which record only transaction time) implied that this system supports temporal relations; the research reported here into this implication shows that in a real sense it is true. P. Thompson's dissertation work inspired investigation of the system topology and inherited transaction times, which led us to temporal generalization. Comments on earlier drafts of this paper by L. Bækgaard, C. Dyreson, S. Hsu, L. Mark, and M. Soo helped improve the presentation.

References

[1] R. K. Abbott and H. Garcia-Molina, "Scheduling real-time transactions: A performance evaluation," *ACM Trans. Database Syst.*, vol. 17, no. 3, pp. 513–560, Sept. 1992.

[2] M. Adiba, N. B. Quang, and J. Palazzo de Oliveira, "Time concept in generalized data base," in *ACM 13th Ann. Comput. Sci. Conf.*, 1985, pp. 214–223.

[3] M. Adiba and N. B. Quang, "Historical multimedia databases," in *Proc. 12th Int. Conf. Very Large Data Bases*, 1986, pp. 63–70.

[4] ——, "Dynamic database snapshots, albums, and movies," in *Proc. Conf. Temporal Aspects in Inform. Syst.*, 1987, pp. 207–225.

[5] M. Adiba, "Derived relations: A unified mechanism for views, snapshots, and distributed data," in *Proc. 7th Int. Conf. Very Large Data Bases*, 1981, pp. 293–305.

[6] M. E. Adiba and B. G. Lindsay, "Database snapshots," in *Proc. 6th Int. Conf. Very Large Databases*, 1980, pp. 86–91.

[7] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, Nov. 1983.

[8] J. Ben-Zvi, "The time relational model," Ph.D. dissertation, Comput. Sci. Dept., UCLA, USA, 1982.

[9] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, June 1970.

[10] C. J. Data, *An Introduction to Database Systems, Volume II.* Reading, MA: Addison-Wesley, 1985.

[11] ——, *An Introduction to Database Systems, Volume I*, 5th ed. Reading, MA: Addison-Wesley, 1990.

[12] B. Dubrovsky, "Universal data access for time series analysis," *PIXEL*, vol. 2, pp. 42–44, Mar./Apr. 1991.

[13] C. E. Dyreson and R. T. Snodgrass, "Timestamp semantics and representation," *Inform. Syst.*, vol. 18, no. 3, pp. 143–166, 1993.

[14] R. Elmasri, Y.-J. Kim, and G. T. J. Wuu, "Efficient implementation techniques for the time index," in *Proc. 7th Int. Conf. Data Eng.*, 1991, pp. 102–111.

[15] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems.* Redwood City, CA: Benjamin/Cummings, 1989.

[16] R. Elmasri, G. T. J. Wuu, and Y.-J. Kim, "The time index: An access structure for temporal data," in *Proc. 16th Int. Conf. Very Large Data Bases*, 1990, pp. 1–12.

[17] S. K. Gadia, "A homogeneous relational model and query languages for temporal databases," *ACM Trans. Database Syst.*, vol. 13, no. 4, pp. 418–448, Dec. 1988.

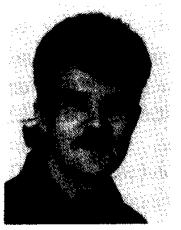
[18] S. K. Gadia and C.-S. Yeung, "A generalized model for a relational temporal database," in *Proc. ACM SIGMOD Conf.*, 1988, pp. 251–259.

[19] H. Gunadhi and A. Segev, "A framework for query optimization in temporal databases," in *5th Int. Conf. Statistical and Scientific Database Mgmt. Syst.*, 1989.

[20] ——, "Event-joint optimization in temporal relational databases," in *Proc. 15th Int. Conf. Very Large Data Bases*, 1989, pp. 205–215.

[21] A. Segev, H. Gunadhi, R. Chandra, and J. G. Shantikumar, "Selectivity estimation in temporal manipulation," *Inform. Sci.*, vol. 74, no. 1–2, Oct. 1993.

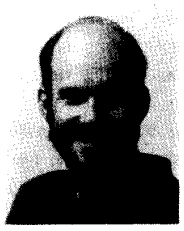
[22] P. Hall, J. Owlett, and S. J. P. Todd, "Relations and entities," in G. M. Nijssen, Ed., *Modeling in Data Base Management Systems*. Amsterdam, Netherlands: North-Holland, 1976, pp. 201–220.
[23] M. Hammer and D. McLeod, "Database Description with SDM: A semantics Database Model," *ACM Trans. Database Syst.*, vol. 6, no. 3, pp. 351–386, Sept. 1981.
[24] J. P. Held and J. V. Carlis, "The applicative data model," *Inform. Sci.*, 1989, pp. 249–283.
[25] C. S. Jensen, "Toward the realization of transaction time database systems," Ph.D. dissertation, CS-TR-2568, UMIACS-TR-90-144, Dept. Comput. Sci., University of Maryland, College Park, MD, USA, 1990.
[26] C. S. Jensen and L. Mark, "A framework for vacuuming temporal databases," Tech. Rep. CS-TR-2516, UMIACS-TR-90-105, Dept. Comput. Sci., Univ. of Maryland, College Park, MD, USA, 1990.
[27] ——, "Queries on change in an extended relational model," *IEEE Trans. Knowl. Data Eng.*, vol. 4, no. 2, pp. 192–200, Dec. 1991.
[28] C. S. Jensen, L. Mark, and N. Roussopoulos, "Incremental implementation model for relational databases with transaction time," *IEEE Trans. Knowl. Data Eng.*, vol. 3, no. 3, pp. 461–473, Sept. 1991.
[29] C. S. Jensen, L. Mark, N. Roussopoulos, and T. Sellis, "Using caching, cache indexing, and differential techniques to efficiently support transaction time," *VLDB J.*, vol. 1, no. 2, pp. 75–111, Jan. 1992.
[30] C. S. Jensen and R. T. Snodgrass, "Temporal specialization," in F. Golshani, Ed., *Proc. Int. Conf. Data Eng.*, 1992, pp. 594–603.
[31] S. Jones, P. Mason, and R. Stamper, "Legol 2.0: A relational specification language for complex rules," *Inform. Syst.*, vol. 4, no. 4, pp. 293–305, Nov. 1979.
[32] K. A. Kimball, "The DATA system," M.S. thesis, Univ. of Pennsylvania, USA, 1978.
[33] ——, "Indexing techniques for historical databases," in *Proc. 5th Int. Conf. Data Eng.*, 1989, pp. 127–137.
[34] C. P. Kolovson and M. Stonebraker, "Segment indexes: Dynamic indexing techniques for multi-dimensional interval data," in *Proc. ACM SIGMOD Conf.*, 1991, pp. 138–147.
[35] T. Y. C. Leung and R. R. Muntz, "Query processing for temporal databases," in *Proc. 6th Int. Conf. Data Eng.*, 1990, pp. 200–208.
[36] B. Lindsay, L. Hass, C. Mohan, H. Pirahesh, and P. Wilms, "A snapshot differential refresh algorithm," in *Proc. ACM SIGMOD Conf.*, 1986, pp. 53–60.
[37] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of multiple autonomous databases," *ACM Computing Surv.*, vol. 22, no. 3, pp. 267–293, Sept. 1990.
[38] D. Lomet and B. J. Salzberg, "The performance of a multiversion access method," in *Proc. ACM SIGMOD Conf.*, 1990, pp. 353–363.
[39] N. Lorentzos and R. Johnson, "Extending relational algebra to manipulate temporal data," *Inform. Syst.*, vol. 13, no. 1, pp. 289–296, 1988.
[40] S. B. Navathe and R. Ahmed, "A temporal relational model and a query language," *Inform. Sci.*, vol. 49, pp. 147–175, Oct. 1989.
[41] K. Ramamritham, "Real-time databases," *Int. J. Distrib. Parallel Databases*, vol. 1, no. 2, pp. 199–226, 1993.
[42] D. Rotem and A. Segev, "Physical organization of temporal data," in *Proc. 3rd Int. Conf. Data Eng.*, 1987, pp. 547–553.
[43] L. A. Rowe and M. R. Stonebraker, Eds., "The POSTGRES papers," Tech. Rep. UCB/ERL M86/85, Electron. Res. Lab., College of Eng., Univ. of California, Berkeley, CA, USA, June 1987.
[44] B. J. Salzberg and D. Lomet, "Access methods for multiversion data," in *Proc. ACM SIGMOD Conf.*, 1989, pp. 315–324.
[45] N. L. Sarda, "Extensions to SQL for historical databases," *IEEE Trans. Knowl. Data Eng.*, vol. 2, pp. 220–230, June 1990.
[46] B.-M. Schueler, "Update reconsidered," in *Proc. IFIP Working Conference on Modeling in Data Base Mgmt. Syst.*, 1977, pp. 149–164.
[47] A. Segev and A Shoshani, "Modeling temporal semantics," in M. Leonard, C. Rolland, F. Bodart, Eds., *Temporal Aspects in Information Systems*. Amsterdam, Netherlands: North-Holland, 1988, pp. 47–58.
[48] ——, "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.
[49] ——, "Logical modeling of temporal data," in *Proc. ACM SIGMOD Conf.*, 1987, pp. 454–466.
[50] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surv.*, vol. 22, no. 3, pp. 183–236, Sept. 1990.
[51] A. Shoshani and K. Kawagoe, "Temporal data management," in *Proc. 12th Int. Conf. Very Large Data Bases*, 1986, pp. 79–88.
[52] J. M. Smith and D. C. P. Smith, "Database abstractions: Aggregation and generalization," *ACM Trans. Database Syst.*, vol. 2, no. 2, pp. 105–133, June 1977.
[53] R. T. Snodgrass, "The Temporal Query Language TQuel," *ACM Trans. Database Syst.*, vol. 12, no. 2, pp. 247–298, June 1987.
[54] R. T. Snodgrass and I. Ahn, "A taxonomy of time in databases," in *Proc. ACM SIGMOD Conf.*, 1985, pp. 236–246.
[55] ——, "Temporal databases," *IEEE Comput.*, vol. 19, no. 9, pp. 35–42, Sept. 1986.
[56] M. Soo and R. T. Snodgrass, "Multiple calendar support for conventional database management systems," Tech. Rep. 92-7, Comput. Sci. Dept., Univ. of Arizona, Feb. 1992.
[57] M. Soo, R. T. Snodgrass, C. Dyreson, C. S. Jensen, and N. Kline, "Architectural extensions to support multiple calendars," TempIS Tech. Rep. 32, Comput. Sci. Dept., Univ. of Arizona, USA, May 1992.
[58] M. Stonebraker and L. A. Rowe, "The design of POSTGRES," in C. Zaniolo, Ed., *Proc. ACM SIGMOD Conf.*, 1986, pp. 340–355.
[59] M. Stonebraker, L. A. Rowe, and M. Harohama, "The implementation of POSTGRES," *IEEE Trans. Knowl. Data Eng.*, vol. 2, pp. 125–142, Mar. 1990.
[60] M. Stonebraker, "The design of the POSTGRES storage system," in *Proc. 13th Int. Conf. Very Large Data Bases*, 1987, pp. 289–300.
[61] G. Thomas, G. R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman, "Heterogeneous distributed database systems for production use," *ACM Computing Surv.*, vol. 22, no. 3, pp. 237–266, Sept. 1990.
[62] P. M. Thompson, "A temporal data model based on accounting principles," Ph.D. dissertation, Dept. Comput. Sci., Univ. of Calgary, AB, Canada, Mar. 1991.



Christian S. Jensen (S'90–M'91) received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science from Aalborg University, Denmark, in 1985, 1988, and 1991, respectively.

For 2 1/2 years, starting Fall 1988, he was part of the Database Systems Group in the Department of Computer Science at the University of Maryland, College Park. In December 1990, he joined the faculty of the Department of Mathematics and Computer Science at Aalborg University as an Assistant Professor. In February 1994, he became an Associate Professor, also at Aalborg University. During each of 1991, 1992, and 1994, he visited the Department of Computer Science at the University of Arizona for six months. His research interests include database systems architecture, query processing and optimization, temporal data models and query languages, object orientation, database design, data modeling, and incomplete information.

Dr. Christian has served on the program committees for a number of conferences, including VLDB, ACM SIGMOD, and IEEE Data Engineering and serves regularly as a reviewer for major database journals. He is the general chair of the 1995 International Workshop on Temporal Databases.



Richard T. Snodgrass (S'79–M'81–SM'87) received the Ph.D. from Carnegie Mellon University in 1982, and joined the University of Arizona in 1989.

His research interests include temporal databases and programming environments.

Dr. Snodgrass directed the design and implementation of the Scorpion Meta Software Development Environment, described in his book, *The Interface Description Language: Definition and Use* (Computer Science Press). He is an Associate Editor of the *ACM Transactions on Database Systems* and is on the editorial board of the *International Journal of Computer and Software Engineering*. He chaired the program committee for the 1994 SIGMOD Conference. He is co-editing a special section of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING on Temporal and Real-Time Databases.