3

Temporal Specialization and Generalization

Christian S. Jensen and Richard T. Snodgrass

A standard relation is two-dimensional with attributes and tuples as dimensions. 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.

While, 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 which 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 associated multiple transaction times telling when it was stored in previous temporal relations. 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 is 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.

1 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 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 temperatures during a chemical experiment, temperature measurements are recorded in the temporal relation *after* they are valid, due to 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 [33]. 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 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 was given a salary raise by a manager. This fact has an associated time when the raise was effective as well as the time when it was entered into the relation on the managers workstation. Later, this fact was copied into the centralized departmental personnel relation, and is associated with an additional time value, namely the time 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.

The paper extends a previously presented taxonomy on time in databases [51, 52]. 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 have one of four types. In a *snapshot* relation, a fact has neither a valid 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 the transaction time when it became current and the transaction time when it ceased to be current. Consequently, a rollback relation is ever-growing. While a rollback relation reflects the history of update activities, an *historical* relation models the part of reality modeled by the database. A fact in such a relation has a 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. Though we use relational terminology throughout this paper, most of the analysis applies analogously to other data models.

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 time-stamp per fact (e.g., [30, 62, 59]), 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 time-stamp 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. In many practical applications, valid and transaction times of facts exhibit interrelationships. Third, the taxonomy assumes that each fact has at most one transaction time and one valid time time-stamp (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. While we have focused primarily on comprehensiveness, we have not considered types of restricted interrelations that are of doubtful use. To address the third shortcoming, we provide the means for specifying the application system contexts of temporal relations.

We will not be concerned here with the semantics of time-varying attributes,

[1]From now on, we use the shorter, but not quite precise, terms ‘valid time-stamp’ and ‘transaction time-stamp’.

i.e., how to use time-stamp values and stored attribute values to derive the value of a time-varying attribute. For example, we will not address the issues of how to derive the temperature of a chemical reaction at an arbitrary point in time from time-stamped and stored temperature measurements. We are interested only in the semantics of the time stamps themselves.

The framework developed here allows researchers as well as database and system designers to precisely characterize, compare, and thus better understand specialized and generalized temporal relations and the application systems in which they are embedded. 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. To indicate that the framework is useful for database designers in understanding individual temporal relations in a particular design, we provide sample realistic situations in which each type of defined specialized relation may arise. These also serve as proof that the definitions are of practical as well as of academic interest. To demonstrate the relevance of the framework for researchers and system designers in understanding application systems with embedded temporal relations, we consider in detail how different types of temporal relations may coexist in sample application systems.

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. In this paper, we indicate how query processing/optimization techniques and secondary storage structures designed for one-dimensional, time-oriented data may be naturally extended to efficiently support specialized two-dimensional temporal data. As a result, 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 two-dimensional temporal data may also exploit the additional semantics discussed in this paper.

While we have not found directly related research beyond what has been mentioned already, the topics of the paper concern a multitude of previous research efforts. We will examine this previous research in detail in Sections 4 and 8.

The paper is structured as follows. In Section 2, we present a general definition and description of a temporal relation. In the following section, we examine the kinds of restrictions one might impose on temporal relations, considering in turn restrictions on isolated events, on collections of events, on isolated intervals, and on collections of intervals. Many previously proposed time-oriented data models do not support general temporal relations, and some support only a single time dimension. In Section 4, we use the framework to classify existing data models, and we show that some one-dimensional models do in fact support specialized temporal relations. In Section 5, we introduce generalized temporal relations. In Section 6, we present a comprehensive sample application system with embedded, generalized temporal relations. Queries on generalized relations may provide the same answers as queries on the underlying relations; in Section 7, we examine means for the database system to ensure that such queries always yield correct results. Section 8 contains a brief analysis of how existing approaches to efficiently store and retrieve one-dimensional time-varying data may be modified to support specialized temporal relations, thereby contributing to the lightly researched area of support for two-dimensional temporal data. The final section summarizes our work and points to future research.

2 A Conceptual Model of a Temporal Relation

We present a conceptual model of a temporal relation as a prelude to the extensions discussed in the remainder of the paper. Note that the adjective "temporal" (snapshot, rollback, and historical, as well) has most often been attributed to databases. We will take a more general approach and use it only for relations because a single database may consist of relations of several types.

A temporal relation 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 time-stamp
- valid time-stamp (interval or event)
- time-invariant attribute values
- time-varying attribute values
- 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 [11, 26]. We will 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 (c.f., a "life-line" [54] or a "time sequence" [55]). Thus, a relation (c.f., a "time sequence collection" [55]) 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 time-stamp generators (TSGs); thus each historical state has an associated unique transaction time. The granularity of transaction time-stamps 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. Therefore, 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 have been characterized as stepwise constant. We will 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, then immediately re-inserted, 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 of items for implementing the rollback operator [8, 54]. 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 date and time values.

Valid times are usually supplied by the user, but they may be system-generated. The valid time-stamp of an item records when the facts represented by the time-varying (and time-invariant) and user-defined time attribute values are true in reality. Valid times are always drawn from the domain of times and dates. The items of a relation may represent events, in which case the valid time-stamp 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 time-stamp of an item is an interval consisting of two valid time values. The valid time-stamps are used by the database system for implementing the time-slice operator [8, 32].

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* [48] 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 time-stamp granularity, or the database system may impose a fixed granularity on all relations managed by the database system. While 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.

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

When temporal relations are viewed as parts of larger application systems where items may flow between relations, generalizations arise. A temporal relation may inherit the transaction time attribute of another relation from which it receives items. This allows users of that relation to ask temporal queries on the relation itself and, in addition, on the other relation. With this generalization, we rename the transaction time attribute to the *primary transaction time attribute*, and we add an arbitrary number of *inherited transaction time attributes*. Each of the inherited transaction time attributes has an associated temporal relation in which it is the primary transaction time attribute.

In addition to associating a time value with an item when it is stored in a relation, times may be associated with items when more general events occur, e.g., when the item is placed into a buffer or when a particular processor receives the item. This generalization adds an arbitrary number of *TSG-generated time attributes* to temporal relations. Values of these attributes are system-supplied and are produced by non-decreasing TSGs.

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 time-stamp, the model makes no mention of whether tuple time-stamping or attribute-value time-stamping is employed. Neither do we assume a particular data model; items could be tuples in a relational database [10], records in a network database [13], or events in a time sequence collection [55]. 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 time-stamp and an interval transaction time-stamp [58] or with one or two valid time-stamps and three transaction time-stamps [8], as a backlog relation of insertion, modification, and deletion operations (tuples) with single transaction time-stamps [31] or with time warp attributes [69], and as tuples containing attributes time-stamped with one or more finite unions of intervals (termed *temporal elements* [19]).

3 Specialized Temporal Relations

In this section, we characterize temporal relations according to the interrelations of their time-stamps. In Sections 3.1 and 3.2, we consider singly stamped items (event stamped), and in Sections 3.3 and 3.4, we consider doubly stamped items (interval stamped). In Sections 3.1 and 3.3, we characterize relations considering the time-stamps of individual items in isolation, and in Sections 3.2 and 3.4, we characterize relations considering the interrelations of time-stamps of distinct items. In Section 3.5, we present a final, orthogonal specialization of temporal relations. Then, in Section 3.6 we relate the specializations of event and interval temporal relations. In Section 3.7, we relate the application of properties on a per relation basis to the corresponding properties applied 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 intensional definitions, i.e., for a relation schema to have a particular type, all its possible (non-empty) extensions must satisfy the definition of the type. The restrictions 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 time-stamps 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 time-stamp attributes. Specializations apply to any ordered pair of time-stamp attributes. Even though a generalized temporal relation has multiple time-stamp attributes, we choose for simplicity to apply specializations to only one ordered pair of time-stamp 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. While 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.

By its very nature, a taxonomy should be comprehensive. While striving towards achieving this, we have at the same time attempted to include only specializations that are of practical interest. We show that with some restrictions, the taxonomy based on isolated events is complete. The inter-event based taxonomy is restricted to cover the concepts of sequentiality and regularity, and the isolated interval based taxonomy covers only regularity. The inter-interval 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 number 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.

3.1 Taxonomy on Isolated Events

In this section we consider only *events* that take place at an instant of time in reality. Let R be a temporal relation, and let e be an 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 will mention only a single valid time vt_e and a single transaction time tt_e. In examples where we illustrate the definitions, we will assume that tt_e is $tt_e^{\vdash}$ (i.e., we consider insertion, not deletion or modification).

We formally define a number of specialized temporal relations by restricting the allowed interrelations between valid and transaction time-stamp values of isolated items. Fifteen of the specialized relations are illustrated in Figure 1. 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 non-specialized temporal relation (termed *general*), there are no restrictions on the interrelations of the transaction and valid time-stamps of an item. The dots for the three last cases in the figure symbolize specific valid times computed in terms of corresponding transaction times.

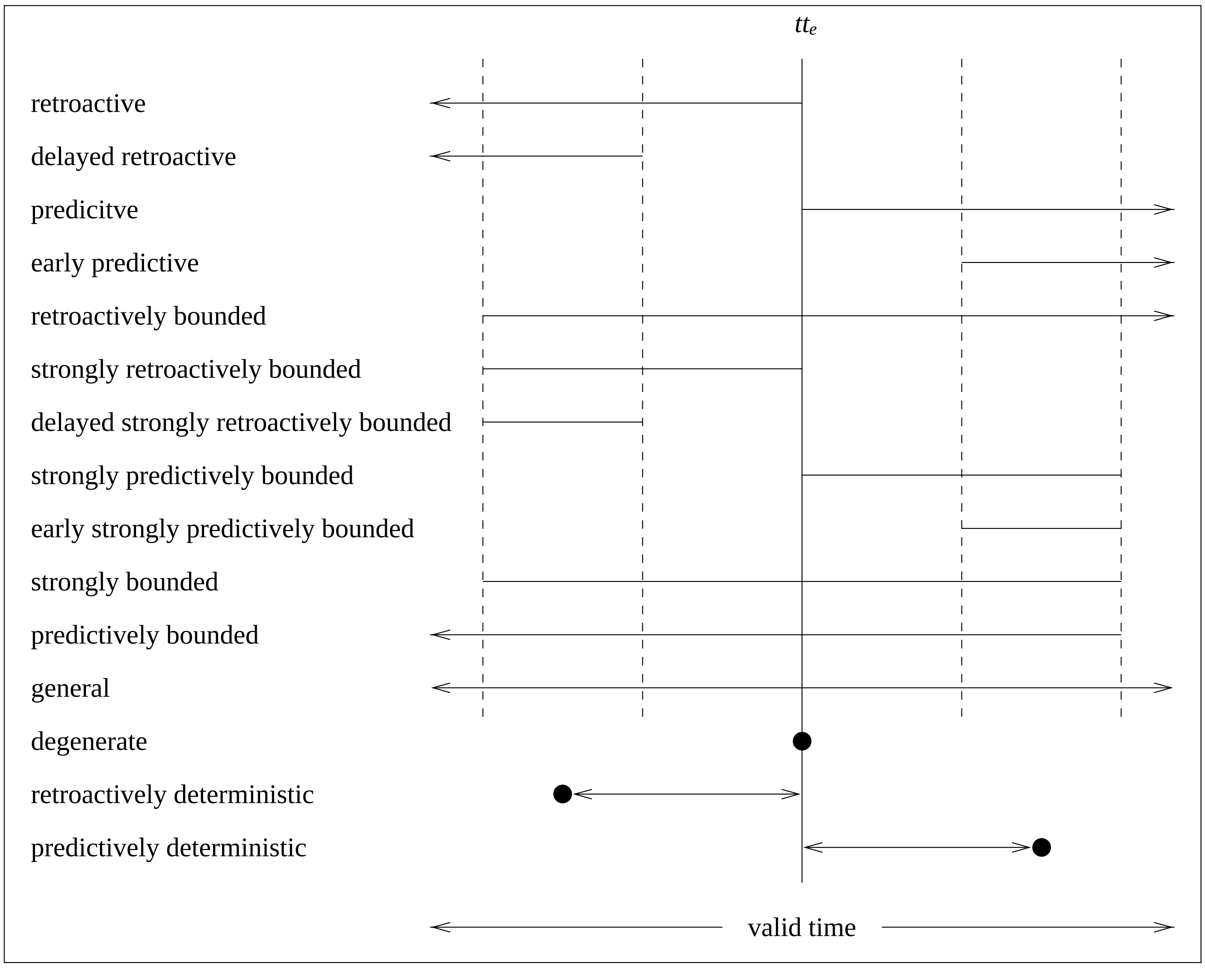
tt_e
retroactive
delayed retroactive
predicitve
early predictive
retroactively bounded
strongly retroactively bounded
delayed strongly retroactively bounded
strongly predictively bounded
early strongly predictively bounded
strongly bounded
predictively bounded
general
degenerate
retroactively deterministic
predictively deterministic
valid time

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

Definition 1 Temporal relation R is *retroactive* if

$$\forall e \in R \ (vt_e \leq tt_e) \qquad \square$$

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 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 of storage can be determined. For example, a particular set-up for the sampling of temperatures may result in delays that always exceed 30 seconds. This gives rise to a delayed retroactive relation.

Definition 2 Temporal relation R is *delayed retroactive with bound* $\Delta t \geq 0$ if

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

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 seconds, one day) or may be calendric-specific. An example of the latter is one month, where a month in the Gregorian calender contains 28 to 31 days, depending on the date to which the duration is added or subtracted.

Definition 3 Temporal relation R is *predictive* if

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

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 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 which specializes the retroactive temporal relation, the early predictive temporal relation is the specialization of the predictive temporal relation.

Definition 4 Temporal relation R is *early predictive with bound* $\Delta t \geq 0$ if

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

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 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 valid time-stamp never is less than the transaction time-stamp by more than a bounded time interval. In all bounded, delayed, and early relations, the bounds are fixed at schema definition time.

Definition 5 Temporal relation R is *retroactively bounded with bound* $\Delta t \geq 0$ if

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

Note that in a retroactively bounded relation, the valid time-stamp may exceed the transaction time-stamp. An example is a relation recording the project each employee is assigned to. While assignments 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.

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

Definition 6 Temporal relation R is *strongly retroactively bounded with bound* $\Delta t \geq 0$ if

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

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, the valid time-stamp is not only less than the transaction time-stamp within a lower bound—in addition, an upper bound (minimum delay) is also imposed.

Definition 7 Temporal relation R is *delayed strongly retroactively bounded with bounds* $\Delta t_1 \geq 0$ *and* $\Delta t_2 \geq 0$, where $\Delta t_1 \leq \Delta t_2$, if

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

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 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 time-stamp is in a bounded time interval after the transaction time-stamp, and the early specialization also adds a (positive) lower bound on the valid time-stamp.

Definition 8 Temporal relation R is *strongly predictively bounded with bound* $\Delta t \geq 0$ if

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

Definition 9 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

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

Direct deposit pay checks illustrate 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. In addition, the bank needs the tape at least three days in advance.

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

Definition 10 Temporal relation R is *strongly bounded with bounds* $\Delta t_1 \geq 0$ *and* $\Delta t_2 \geq 0$ if

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

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

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

Definition 11 Temporal relation R is *predictively bounded with bound* $\Delta t \geq 0$ if

$$\forall e \in R \; (vt_e \leq tt_e + \Delta t) \qquad \square$$

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

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

Definition 12 Temporal relation R is *degenerate* if

$$\forall e \in R \; (vt_e = tt_e) \qquad \square$$

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

At the implementation level, a degenerate temporal relation can be advantageously treated as a rollback relation due to the fact that relations are append-only and items are entered in time-stamp order—this will be discussed in more detail in Section 8. The process of recording degenerate relations is referred to as the *asynchronous method* [69].

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

Definition 13 Temporal relation R is *determined with mapping function m* if

$$\forall e \in R \ (vt_e = m(e)) \qquad \square$$

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, consider the determined versions of the retroactive and predictive temporal relations.

Definition 14 Temporal relation R is *retroactively determined with mapping function m* if

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

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 15 Temporal relation R is *predictively determined with mapping function m* if

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

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 the bounded version of the above two types of relations.

Definition 16 Temporal relation R is *strongly retroactively bounded determined with mapping function m and bound* $\Delta t \geq 0$ if

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

Definition 17 Temporal relation R is *strongly predictively bounded determined with mapping function m and bound* $\Delta t \geq 0$ if

$$\forall e \in R \ (vt_e = m(e) \wedge tt_e \leq m(e) \leq tt_e + \Delta t) \qquad \square$$

The examples given previously were in fact bounded.

The generalization/specialization structure of the specialized temporal relations defined above is presented in Figure 2. A relation type can be specialized into any of the successor relation types, 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 counterparts for all the undetermined specialized temporal relations, e.g., strongly bounded determined.

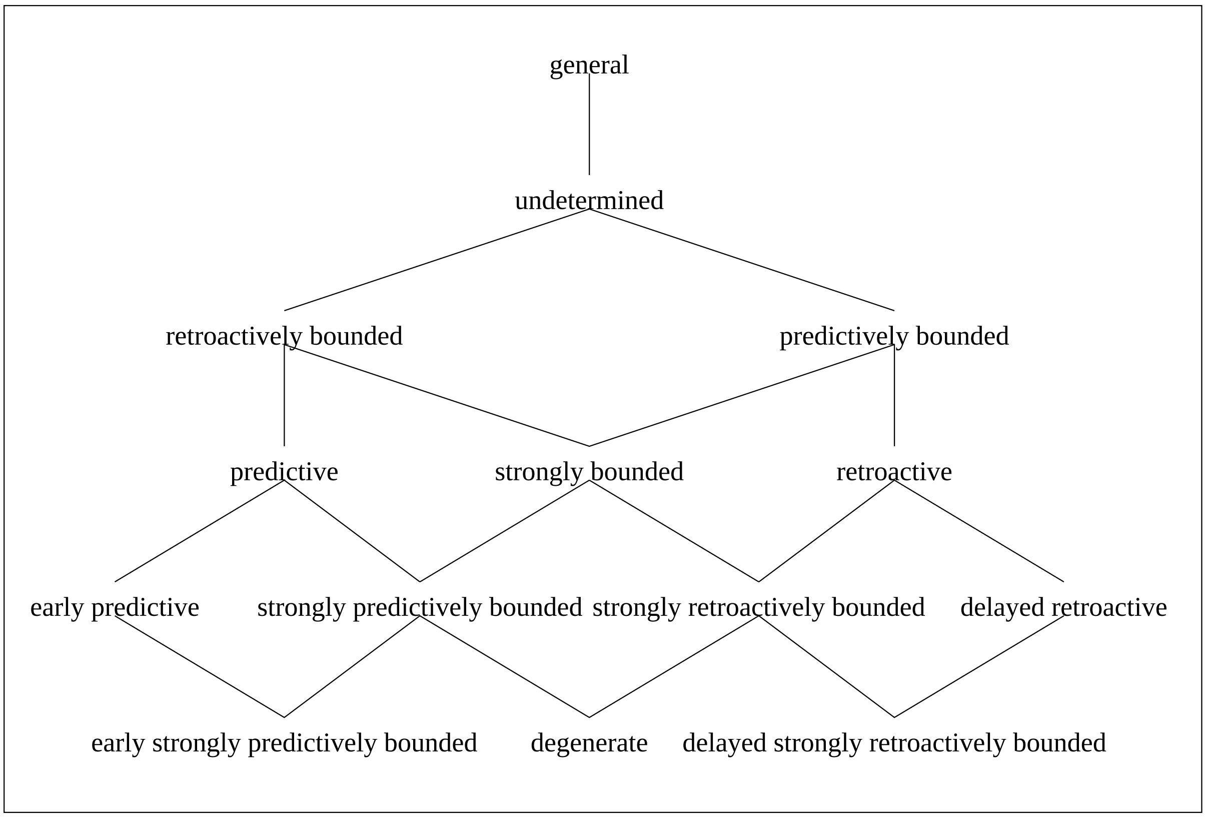

Figure 2: Generalization/Specialization Structure of the Event-based Taxonomy

The isolated event based taxonomy is complete with certain assumptions. To state these, note that the specializations in this section correspond to regions of the two-dimensional space spanned by transaction and valid time. There are five assumptions. First, we are interested only in undetermined relationships. Second, we are only interested 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 time stamps such as, e.g., 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 two-dimensional space, namely lines parallel to $tt_e = vt_e$ for which either (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 regions may be used as building blocks to form non-connected 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, the 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 eleven types of specialized temporal relations, each of which is included in the taxonomy.

3.2 Inter-event Based Taxonomy

The previous definitions were based on predicates on individual, event time-stamped 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 time-stamped 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 while 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 18 Temporal relation R is *globally sequential* if[2]

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

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.

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

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

Now we introduce the notion of a non-decreasing temporal relation. A relation is non-decreasing if items are entered in valid time-stamp order.

Definition 19 Temporal relation R is *globally non-decreasing* if

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

Sequentiality is generally a stronger property than non-decreasing. However, if the relation is degenerate then the two properties are identical. For completeness, we define also a non-increasing temporal relation where items are entered in non-increasing valid time-stamp order.

Definition 20 Temporal relation R is *globally non-increasing* if

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

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 21 Temporal relation R is *transaction time event regular with time unit* $\Delta t \geq 0$ if

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

Note that the transaction time-stamps 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* [69].

Definition 22 Temporal relation R is *valid time event regular with time unit* $\Delta t \geq 0$ if

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

The concept of *granularity* of valid time-stamps can be expressed in terms of this property. For example, if the valid time-stamp granularity is one second then, equivalently, the relation is valid time event regular with the time unit one second.

Definition 23 Temporal relation R is *temporal event regular with time unit* $\Delta t \geq 0$ if

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

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 24 Temporal relation R is *strict transaction time event regular with time unit* $\Delta t \geq 0$ if

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

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

Definition 25 Temporal relation R is *strict valid time event regular with time unit* $\Delta t \geq 0$ if

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

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

Definition 26 Temporal relation R is *strict temporal event regular with time unit* $\Delta t \geq 0$ if

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

While 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, the temporal time unit Δt_3 being some common divisor of Δt_1 and Δt_2. Thus, if $\Delta t_1 = 28$ seconds and $\Delta t_2 = 6$ seconds then $\Delta t_3 = 2$ seconds (largest common divisor). For the strict case, however, valid and transaction time event regularity does not imply temporal event regularity.

Analogous with the ordering properties, the above regularity properties can be defined in a global or per partition fashion. However, the non-strict 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 to 4p.m. during weekdays [42].

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 Figures 3 and 4. The two structures are orthogonal.

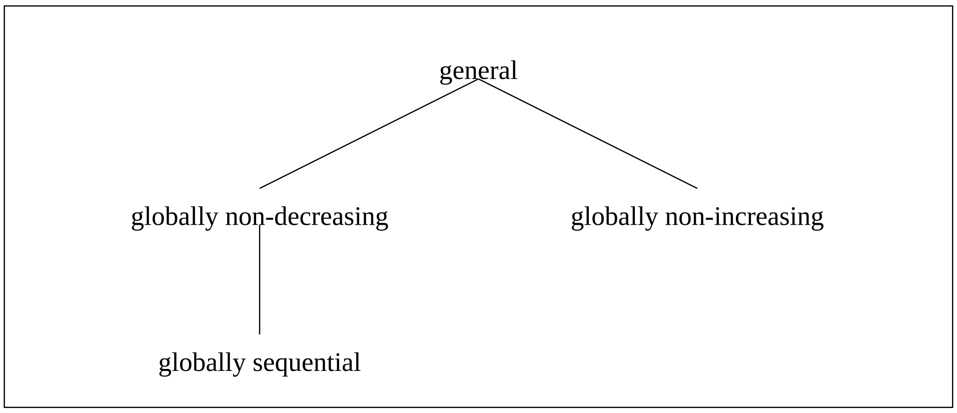

Figure 3: Generalization/Specialization Structure of the Inter-event Based Taxonomy (Part I—orderings)

3.3 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 3.2, k (possibly indexed) is an integer.

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

A temporal relation is transaction time regular, valid time regular, or temporally regular if the transaction time intervals, valid time intervals, or both transaction time and valid time intervals are regular, respectively. Note again that these properties concern durations rather than starting events, and that they can be calendric specific, e.g., one month.

Definition 27 Temporal relation R is *transaction time interval regular with time unit* $\Delta t \geq 0$ if

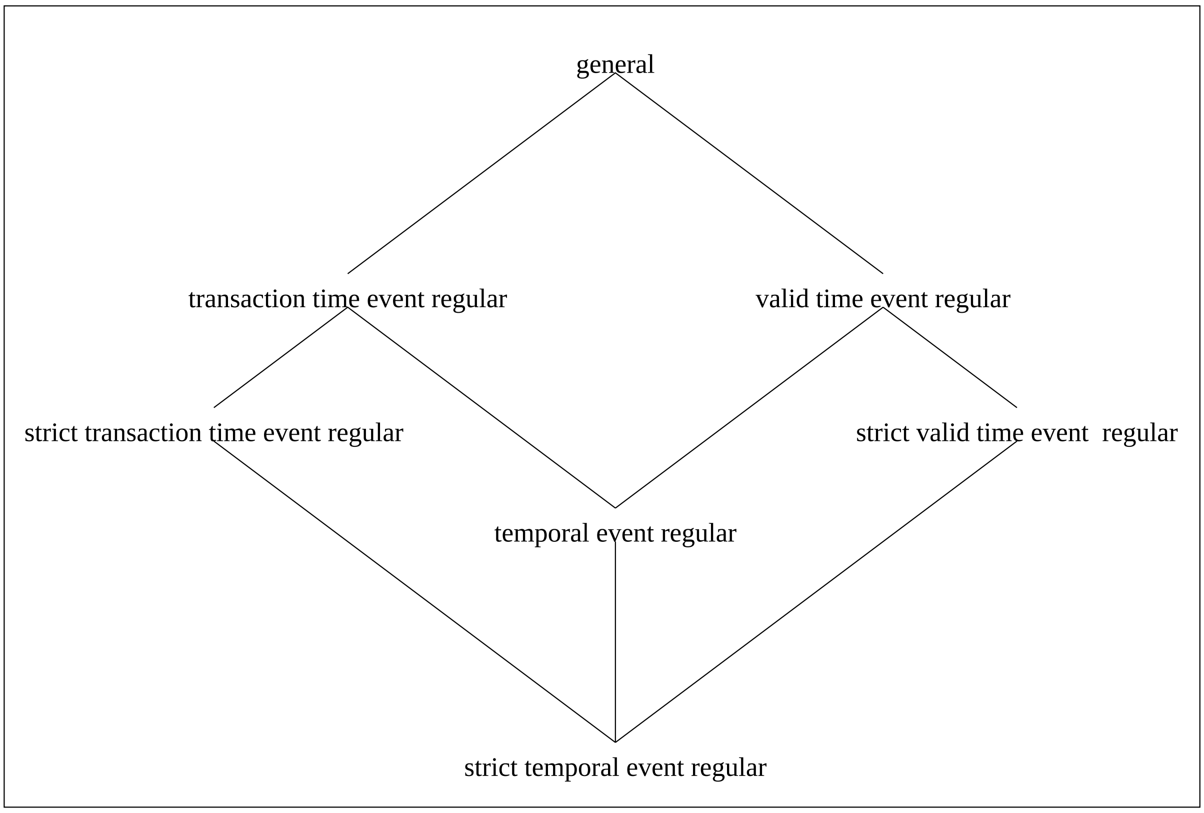
general
transaction time event regular
valid time event regular
strict transaction time event regular
strict valid time event regular
temporal event regular
strict temporal event regular

Figure 4: Generalization/Specialization Structure of the Inter-event Based Taxonomy (Part II—regularity)

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

Definition 28 Temporal relation R is *valid time interval regular with time unit* $\Delta t \geq 0$ if

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

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 29 Temporal relation R is *temporal interval regular with time unit* $\Delta t \geq 0$ if

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

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. These special cases, we term *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 Figure 4, with the exception that "event" is replaced by "interval."

3.4 Inter-interval Based Taxonomy

As with events, we distinguish restrictions that are applied individually to all intervals and restrictions on the interrelationship between multiple intervals in a relation. The restrictions listed below apply to relations, but they may be applied on a per partition basis as well. Many of these same terms also apply to event relations, and were defined in Section 3.2; context should differentiate these uses.

Definition 30 Temporal relation R is *globally sequential* if

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

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

A relation is non-decreasing if items are entered in valid time-stamp order, and it is non-increasing if items are entered in reverse valid time-stamp order.

Definition 31 Temporal relation R is *globally non-decreasing* if

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

Concerning the example just discussed, let us now record each Thursday the next week's assignment. In this case the transaction time (i.e., Thursday) of the next week's assignment (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 non-decreasing.

As with events, sequentiality is a stronger property than non-decreasing.

Definition 32 Temporal relation R is *globally non-increasing* if

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

Definition 33 Temporal relation R is *globally contiguous* if

$$\begin{aligned}\forall e \in R\ (\ \exists e' \in R - \{e\}\ (\ & vt_e^{\dashv} = vt_{e'}^{\vdash} \wedge tt_e < tt_{e'} \\ & \wedge \neg\exists e'' \in R - \{e, e'\}(tt_e < tt_{e''} < tt_{e'})) \\ \vee \forall e' \in R - \{e\}\ & (vt_e^{\vdash} \geq vt_{e'}^{\dashv})) \end{aligned} \qquad \square$$

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 will be given in Section 3.6.

Allen has demonstrated that there exist a total of thirteen possible relationships between two intervals [5]. 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 requires 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 34 Temporal relation R is *successive transaction time* **X** if

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

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

Figure 5 illustrates the specialization/generalization structure for the properties discussed above. In this figure, *successive transaction time* is abbreviated ‘st-’, and *successive transaction time inverse* is abbreviated ‘sti-’.

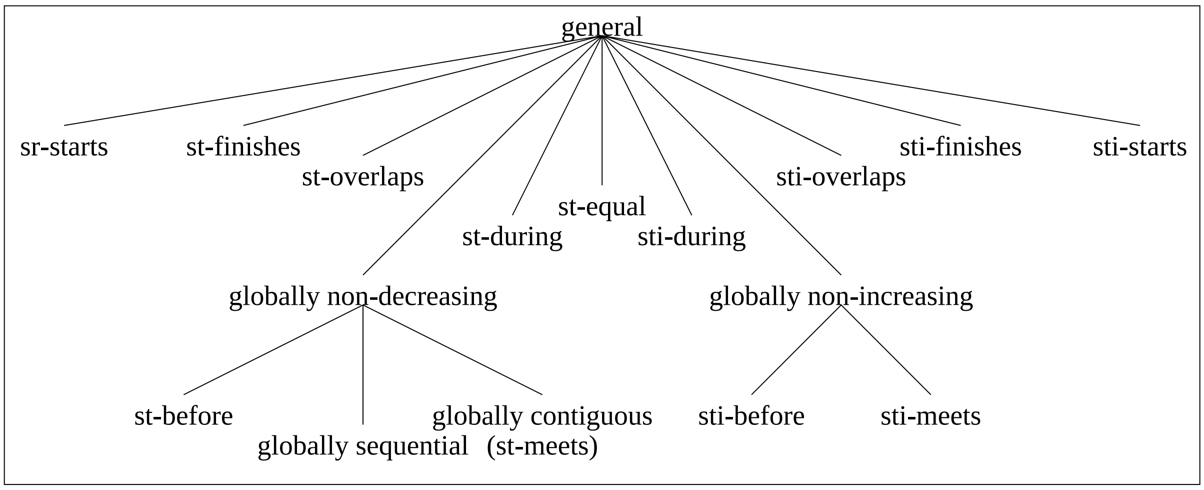

Figure 5: Generalization/Specialization Structure of the Inter-interval Based Taxonomy

3.5 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 *n*th 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 Figure 6 where the dashed lines indicate intermediate, incomplete relations.

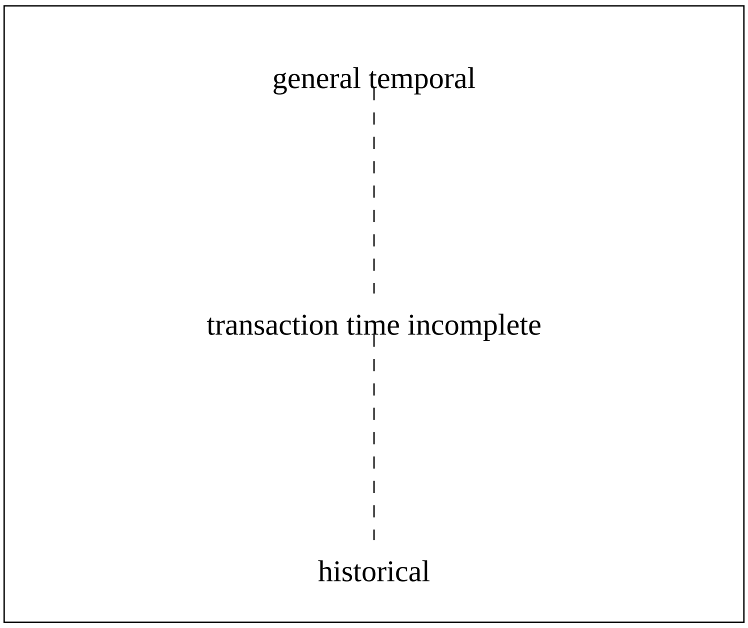

Figure 6: Generalization/Specialization Structure of Transaction Time Incomplete Temporal Relations

3.6 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 [27, 34] and derive 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.

3.7 Interrelations between Per Relation and Per Partition Specializations

We now consider the interrelations of specializations when applied on a per relation basis, on 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 to hold on a *per relation* basis, the set of all items in the relation must satisfy the specialization. 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 Figure 7. A specialization is *per item* if it applies to individual items in isolation (see Sections 3.1 and 3.3); otherwise, it is *inter-item* (see Sections 3.2 and 3.4). 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.

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

Figure 7: Four Types of Specializations on Temporal Relations

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 which 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 $x_1, x_2, \ldots, x_k$ 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 Δt_1, Δt_2, ..., Δt_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 $x_1, x_2, \ldots, x_k$ 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 inter-item specializations instead of per item specializations, no general statements may be made.

3.8 Summary

We have presented an extensive taxonomy of specialized properties of temporal relations. The practical relevance of the definitions are emphasized by examples. The properties apply to either event or interval temporal relations. A relation may have specialized per item properties (Sections 3.1 and 3.3) as well as specialized inter-item properties (Sections 3.2 and 3.4). A relation may also be transaction time incomplete (Section 3.5). 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.

4 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 the taxonomy to perform such a characterization. We proceed by successively applying greater temporal specialization.

4.1 General Temporal Relations

General temporal relations are found in only a few data models [8, 58].

The snapshot mechanism [4] may be extended to support general temporal relations. A *snapshot* of a relation is an independent copy of the current state of that relation at the time of the snapshot. Thus, snapshots are derived from base relations, but they do not change when the underlying base relations change [2, 41]. The snapshot mechanism may be applied to a relation in three ways [1, 7, 6]. First, there is the manual snapshot where a `generate-version` command is used to create a shapshot (termed a "manual album"). Second, there is the periodic snapshot (termed an "automatic album") where, for example, the user may specify, "Keep snapshots for the end of the month for a window of 12 months." Third, there is the successive snapshot where the system creates a new snapshot every time the underlying relation is updated (termed a "movie").

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

4.2 Retroactive Temporal Relations

Gadia presents a multi-dimensional temporal data model which is in turn restricted to a two-dimensional data model with valid and transaction time as the dimensions [23]. In this model, however, only data valid in the past may be stored. For example, it is impossible to store on May 11, 1991 the fact that "As of now, Dr. Jones is hired as an assistant professor from September 1, 1991 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 time-stamps are used for the modeling of real-world activity [19].

Sarda proposes another specialized temporal data model in which current facts may be appended and where so-called retrospective updates (changes to information about the past) are possible [53]. Hence, the transaction time is always equal to or after the valid time, and, like the previous model, this model supports retroactive temporal relations.

4.3 Strongly Retroactively Bounded Relations

In real-time databases, transactions have hard real-time deadlines [3]. If the deadline passes before the transaction is executed, the transaction is unscheduled. 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.

4.4 Degenerate Temporal Relations

Relations representing time sequences and time sequence collections of the TSC model [50, 62, 63, 55] 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 [64] is a per surrogate contiguous relation that is also per surrogate sequential.

The Postgres data model [49, 65] 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 [60] 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. Thus, all updates are physically append-only. Only event time-stamps are possible, and they are unique, increasing, and system-supplied. Additionally, the assumption is made that time-varying attributes have stepwise constant semantics [27, 29, 30]. 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 time-stamps reflecting real time, it may be used as a temporal data model when the transaction and valid times of items coincide, and hence it is also a degenerate temporal model. Similarly, successive snapshots of a conventional relation (a movie) produce a degenerate temporal relation.

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

Adiba introduced an append-only relation which may be modified using special error-correcting operations [6]. 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 [14]. Some are degenerate, some are transaction time event regular, and most are globally ordered.

4.5 Transaction Time Incomplete Temporal Relations

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

4.6 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.

5 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 time-stamps of a temporal relation, thereby coupling the two time-stamps in some fashion. Now, we change perspective and consider temporal relations as parts of larger application systems where items move between multiple temporal relations. We investigate temporal generalization, which involves decoupling time-stamps.

The general 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 [16, 25, 61].

Temporal specialization and generalization are also duals. As we have seen, specialization contracts the space of possible interrelations of time-stamps. Temporal generalization appears in two guises, each of which expands the space of possible interrelations of time-stamps. 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 3, thereby contracting the (two-dimensional) space of possible interrelations; generalization involves moving up these lattices, expanding the space of possible interrelations.

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

Specialization may be applied to any two time-dimensions. Consequently, standard two-dimensional temporal relations may be perceived as multi-dimensional 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.

In this section, we first give an example of an inherited time-stamp generated by a sensor. Next, we define the components that may be used for describing application systems at a suitable level of abstraction. Most notably, we define several so-called time-stamp generators.

5.1 Example: Temperature Monitoring

This section discusses a very simple application with a generalized temporal relation containing one inherited transaction time-stamp. The system, illustrated in Figure 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 time-stamped with the current time when they arrive at a sensor; the time-stamps are obtained from the time-stamp generators *tsg₀* and *tsg₁*. The valid time-stamps of the measurements are assumed to be identical to these sensor time-stamps. 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 time-stamped (using time-stamps obtained from tsg_P) and entered into the relation. The relation thus contains three time-stamps, the valid time-stamp, the (primary) transaction time-stamp (from tsg_P), and the sensor time-stamp (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 time-stamps, which are identical; indeed only one needs to be stored. It is generalized because two transaction time-stamps are recorded in the relation.

Due to 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 time-stamp and the transaction time-stamp, which is why the sensor time-stamp must be included in the relation. The buffer allows us to characterize the relation as globally non-decreasing and as delayed retroactive, with a bound computable from the various delays in the system.

We will examine a more comprehensive example in Section 6, after presenting the types of components used for describing systems with generalized temporal relations.

5.2 System Topology

Transaction time-stamps added to a relation by temporal generalization are generated elsewhere in the system by time-stamp generators. During the design of a generalized temporal relation, the sources of these additional time-stamps must be described; 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

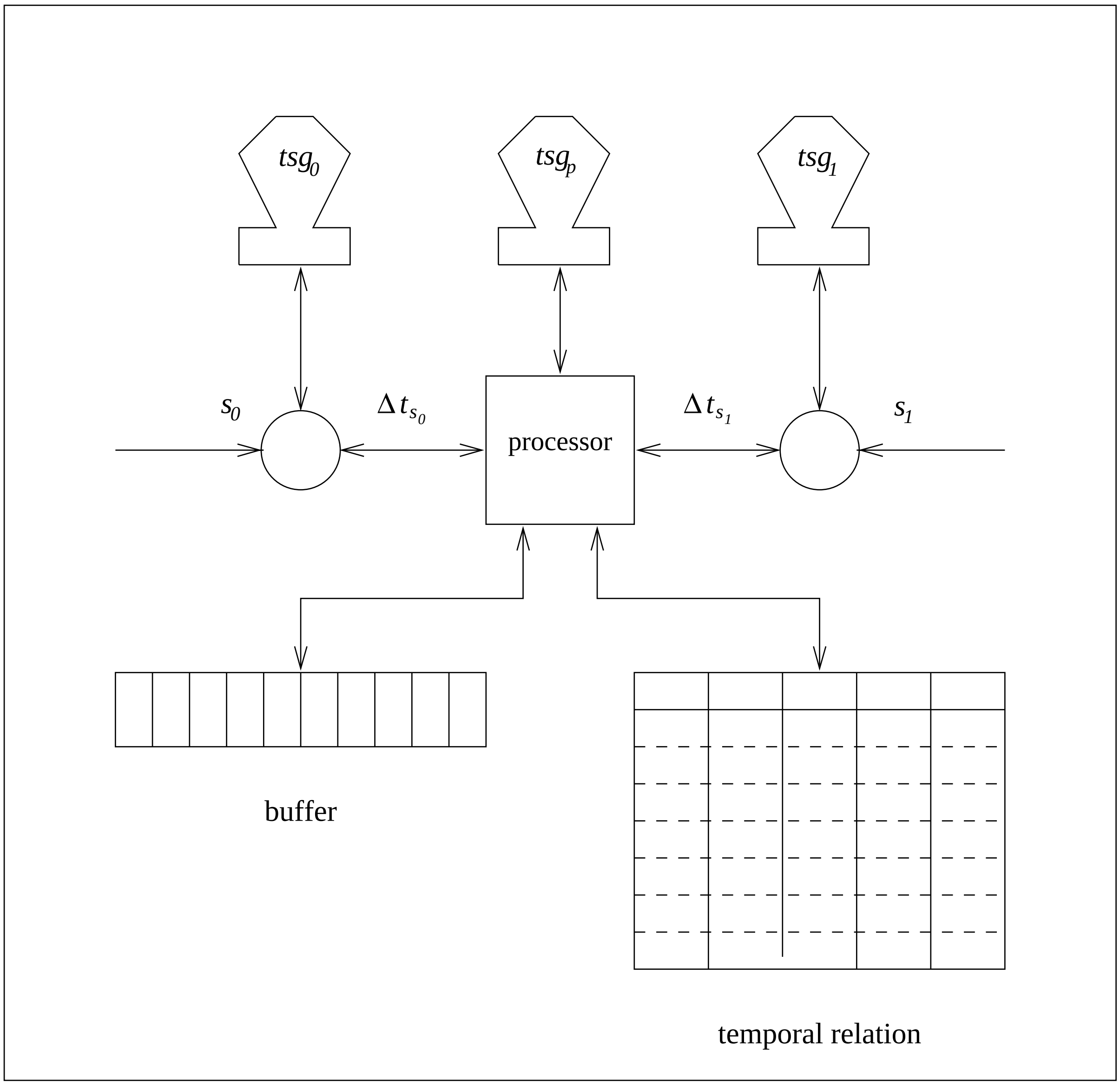

Figure 8: Buffering of Temporal Data

which 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 is recorded by sensors which observe a portion of the real-world being modeled. In a manual scenario, data is received by a processor, either as a result of data input by a user or data retrieved from an on-line source. Data may be time-stamped. A *time-stamp generator* (TSG) is a mechanism that returns time-stamp values on demand. Manually supplied data may or may not contain a valid time-stamp. If not, the receiving processor must append a valid time-stamp, obtained upon request from an available TSG. A processor responsible for entering data into a temporal relation also utilizes a TSG for transaction time-stamping.

A sensor, s_i, forms events (observations). To do so, it sends requests, r_{ik}, to a TSG, γ_j, and receives time-stamps, 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 time-stamp from a TSG whenever it performs an action. For example, a processor may request a stamp when it stores data in a buffer or when it retrieves data from a buffer, and the time-stamp may be made part of the data that is stored. These are TSG-generated time dimensions. A time-stamp added by a processor when it stores an item in a relation is a transaction time-stamp.

Finally, in a system, components of the types mentioned above may be connected via *data channels*. 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 time-stamps from the TSG (see, for example, Figure 8). The system topology does not specify the order in which data is sent along data channels, though specializations of relations may imply a certain data ordering. In Section 7, we discuss the possible ways of interconnecting temporal relations.

We emphasize that the system topology is utilized in this paper only for specifying the source of inherited time-stamp attributes. We do not differentiate between logically centralized, distributed, heterogeneous, federated or multi-databases [44, 57, 70]; this system description should be applicable in varying degrees to all of these systems.

5.3 Summary

In review, a generalized temporal relations has—in addition to user-defined times, the valid time, and a primary transaction time—a number of inherited transaction and TSG-generated times. Transaction times are generated by monotonically increasing TSGs, and other times are generated by non-decreasing TSGs. In addition, user-defined and valid times may be, and often are, user-supplied.

We have identified two kinds of temporal generalization. The first kind is simply the removal of one of more of the restrictions discussed in Section 3. The second kind is the addition of one or more times, either inherited transaction times or TSG-generated times. The system topology identifies the TSG that supplies the time-stamp. An inherited transaction time is distinct from the primary transaction time which is generated when the item was recorded in the temporal relation. TSG-generated times differ from inherited transaction times in that they record not the action of inserting an item into a separate temporal (or rollback) relation, but rather record some internal action by a processor or sensor, such as the placement of an item into a buffer or the removal of an item from a buffer, or the sensing of a value.

Both kinds of generalization expand the space of allowable time-stamp values; specialization, which can be applied to a single time-stamp attribute (e.g., a particular time-stamp attribute may be event regular) or to a pair of time-stamp attributes (e.g., degeneracy specifies that both values are identical), contracts the space of allowable time-stamp values. In this sense, generalization and specialization are duals of each other.

6 An Application System with Multiple 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 Figure 9. An *employee relation* is maintained on the workstation of each manager 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 when relevant.

Eric was hired by LeeAnn with a salary of $2000 per month. Due to 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 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. As both 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.

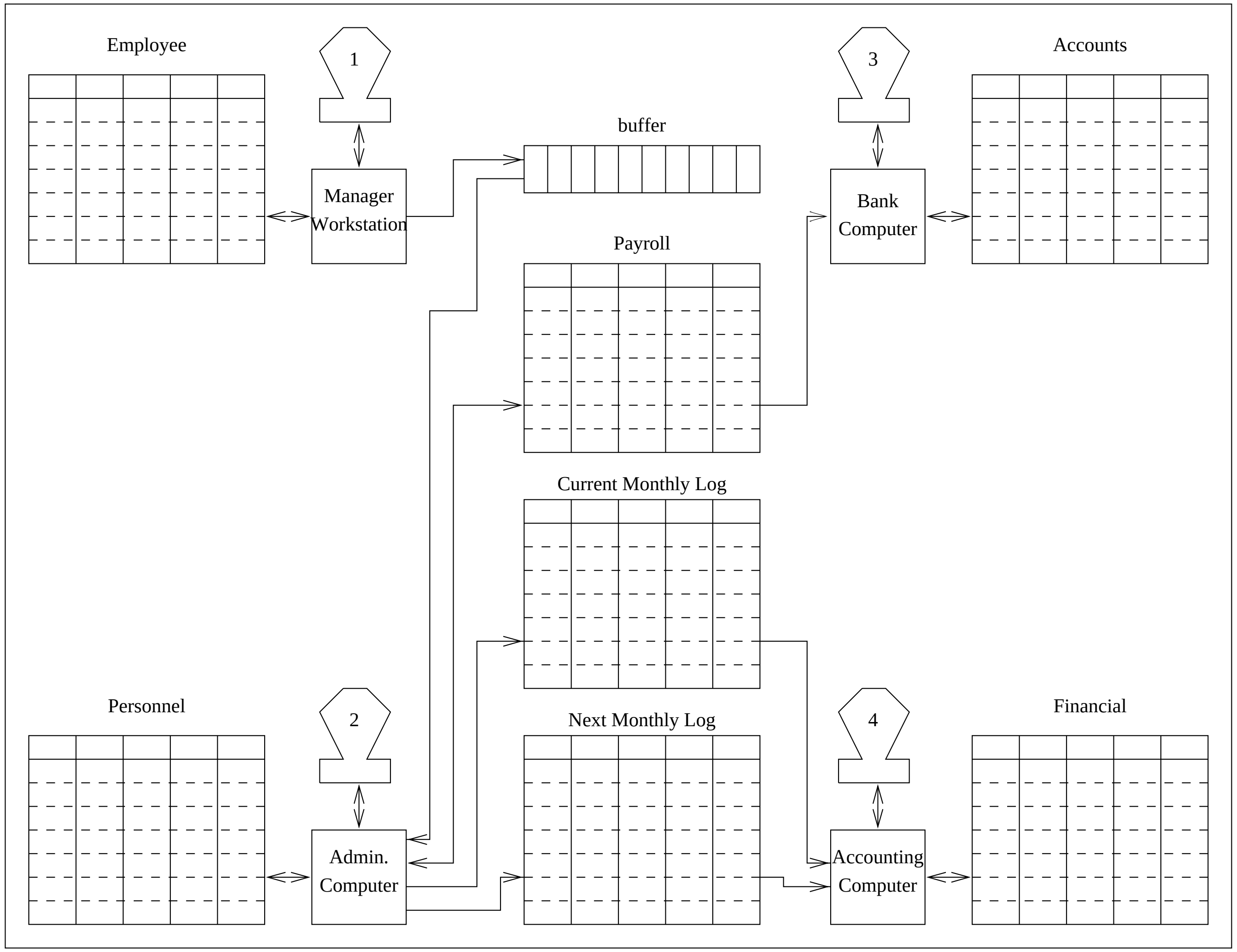

Figure 9: System Topology for the Extendend Example

Personnel:

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

The transaction time-stamp, tt, records when the batch job executed the transaction recording this item; it is supplied by tsg_2. The inherited transaction time-stamp, 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 update 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 pay checks. 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 pay checks 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 the check is to be issued, April 1, is the valid time. Note that Eric's March salary is actually $2300. However, this fact didn't 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'll 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 time-stamps and the valid time-stamps 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 active ever: 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" [69]. 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.

As 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 to, as before, *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 bound 16 to 46 days,* the former bound exemplified by vt = April 30 and tt = May 15, the latter bound exemplified by vt = May 1 and tt = June 15. Finally, because each month is closed off during the next month, the relation is *globally non-decreasing* 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 [69]. This 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, as 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.

7 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 a predecessor transaction time-stamp attribute allows one to query the predecessor relation from the current relation. In this section, we explore this capability.

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 can tell, say, when LeeAnn's secretary made a particular salary adjustment for an employee.

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 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, i.e., 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 as to 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 will not consider the possibility of partial transmittal further.

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 [28]. 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 an 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 an 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 are 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 $\sigma_{tt \leq NOW-8\ days}$.

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

$$\pi_{salary}\ (\sigma_{vt=\text{Mar6} \wedge tt=\text{Mar28} \wedge name=\text{Eric}}(employee))$$

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 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.)

The transmission filters are defined by the designer of the overall application system.

In considering each way of interconnecting temporal relations, we distinguish between three cases as outlined in Figure 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.

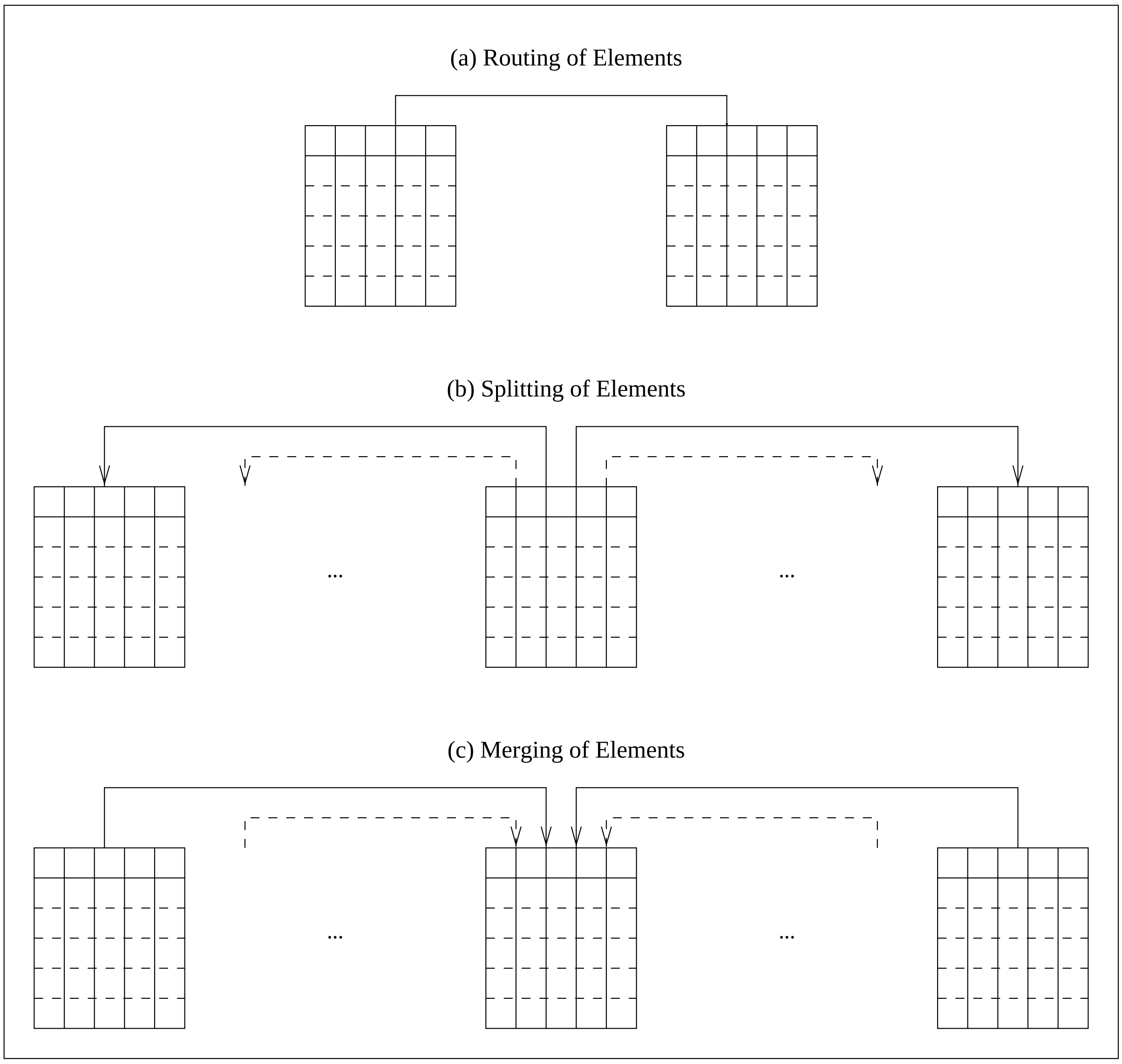

Figure 10: Interconnections of Temporal Relations

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 from 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 6. If employees from several departments were 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 the successor relation to have the same schemas, 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 informations about her own employees. Then the informations recorded locally by each manager is collected in the central personnel relation.

Note that sources may take the roles 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, making the transmission filters between temporal relations more restrictive. Finally note that in the above description, situations where a relation receives items that it itself transmitted are implicitly possible. For simplicity we do not discuss such cycles.

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

8 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 one-dimensional time-varying data may be naturally extended to apply to specialized temporal (i.e., two-dimensional, 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 one-dimensional approaches to two-dimensional data; second, we briefly review related research and show how the general idea applies. We do not attempt to give a detailed analysis of the application of one-dimensional approaches to specialized temporal relations; that would require us to select specific stored representations, indexes, and processing strategies for temporal data, which is beyond the scope of this paper. 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 directed directly towards the efficient support of temporal relations may also be designed to exploit the semantics of specialized temporal relations with resulting performance gains.

8.1 Exploiting Identified Specializations

The general idea can be stated as follows. In order to apply existing techniques previously used to improve the performance of queries on one-dimensional data, we utilize the specific interrelation between valid and transaction time-stamps, guaranteed by the type of a specialized temporal relation, to simply disregard one time dimension and only use 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 time-stamp order at temporal relations, we find it natural to utilize the transaction time dimension and ignore the valid time dimension.

This approach applies, with some variations, to all specialized temporal relations. The application of the approach to specialized temporal relations towards the bottom of a specialization/generalization structure (see Figures 2 to 6), being closer to degenerate relations which never require more than one time-stamp, 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., [31]) or on a per object surrogate basis (e.g., [56, 21]). These are straight forward representations, particularly if write-once storage media are utilized. Assuming physical clustering and strongly retroactively bounded temporal data, the following important property holds: All items with valid time-stamps 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 time-stamp less than t_x. The limit in the number of items depends on the particular retroactive bound and the intensity of update activity for the relation.

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

8.2 Application to Previous Proposals

The issue of efficient temporal query processing is largely unexplored. While 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 stated, it appears that this research may be extended naturally to include the efficient support of specialized temporal data. Below, we briefly review some of this research.

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

Leung and Muntz have proposed a stream processing approach to temporal (semi-) joins [43]. In this approach, the input to, and the output from, stream processors consist of sets 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 effect of different sort orderings on the efficiency and the size of the local state were considered for one temporal join (and as special cases, two semi-joins). A stream join processor that assumes a time-ordered sequence of 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 pre-stream processors that each use a buffer to convert nearly ordered data into totally ordered data (an integration into a single processor may improve performance). The buffer sizes correspond to the sizes of the regions mentioned in the general discussion above.

A more traditional approach to the processing of one-dimensional temporal joins is chosen in most other work in temporal query optimization [20, 22, 21]. Most notably, a temporal event-join consisting of three time-oriented joins is considered [21]. 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 current totally ordered data. In particular, results obtained for append-only databases are highly relevant for specialized temporal relations.

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

A number of research contributions aimed at supporting time-varying data 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 [45, 56] 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 time-stamps that are smaller than the split time. We believe that the time-split mechanism may be modified to make this indexing technique suitable for some types of specialized temporal data.

Also based on the above-mentioned philosophy, Kolovson and Stonebraker generalize R-trees to span both magnetical and optical disk media, thus providing new intermediates between R-trees residing on only a single medium [38]. This is relevant when the bulk of temporal data does not fit on magnetic disk and must be migrated to optical disk [49]. They also introduce tactics aimed at improving observed deficiencies of existing indexing techniques for historical data (e.g., R-trees) [39]. 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 [15, 17]. 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 [30, 31]. 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 techniques. 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 databases and supporting only one kind of time, may be adapted to also apply to specialized forms of temporal relations supporting both kinds of time.

9 Conclusion and Future Research

A temporal relation has two database system-interpreted time attributes, transaction time and valid time. A transaction time-stamp is a simple value, indicating when a fact is stored in the temporal relation. A valid time-stamp 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 time-stamps are independent, meaning that facts may be associated with a point or a pair of points in an unrestricted two-dimensional 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 time-stamps 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 time-stamps, and TSG-generated time-stamps.

We presented an extensive taxonomy of temporal specializations, some restricting the stamps of individual facts, others restricting the stamps on an inter-fact 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 one-dimensional 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. Other potential uses for the semantics include integrity checking and display.

We extended the two-dimensional 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 time-stamps. 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.

Future work is indicated in two areas. As we have shown (Section 8), 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 time-stamp 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 time-stamp attributes and time-varying attributes.

Acknowledgements

The topic of temporal specialization was introduced during discussions between the second author and Arie Segev, whose comments encouraged further exploration of what was then termed “coupled relations.” Mike Stonebraker’s use of valid time examples to illustrate rollback relations in Postgres (which only record transaction time) implied that Postgres supports temporal relations; the research reported here into this implication shows that in a real sense it is true. Peter 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 Lars Bækgaard, Curtis Dyreson, Suchen Hsu, Leo Mark and Michael Soo helped improve the presentation.

This work was conducted while the first author visited the University of Arizona, and it was supported in part by NSF grant ISI-8902707. In addition, the first author was supported by Danish NSF (Statens Naturvidenskabelige Forskningsråd) grant 11-8696-1 SE.

References

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

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

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

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

[5] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

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

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

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

[9] P. P-S. Chen. The Entity-Relationship Model—Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

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

[11] C. J. Date. *An Introduction to Database Systems—Volume II*. The Systems Programming Series. Addison Wesley Publishing Company, First edition, July 1985.

[12] C. J. Date. *Relational Database—Selected Writings*. Addison Wesley Publishing Company, 1986.

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

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

[15] R. Elmasri, Y.-J. Kim, and G. T. J. Wuu. Efficient Implementation Techniques for the Time Index. In *Proceedings of the Seventh International Conference on Data Engineering*, 1991.

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

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

[18] S. Ferg. Modeling the Time Dimension in an Entity-Relationship Diagram. In *Proceedings of the Fourth International Conference on the Entity-Relationship Approach*, pages 280–286, Silver Spring, MD, 1985. IEEE, Computer Society Press.

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

[20] H. Gunadhi and A. Segev. A Framework For Query Optimization In Temporal Databases. In *Fifth International Conference on Statistical and Scientific Database Management Systems*, 1989.

[21] H. Gunadhi and A. Segev. Event-join optimization in temporal relational databases. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, pages 205–215, August 1989.

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

[23] S. K. Gadia and C.-S. Yeung. A Generalized Model for a Relational Temporal Database. In *Proceedings of the ACM SIGMOD Conference*, pages 251–259, 1988.

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

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

[26] P. Hall, J. Owlett, and S. J. P. Todd. Relations and Entities. In G. M. Nijssen, editor, *Modelling in Data Base Management Systems*, pages 201–220. North-Holland, 1976.

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

[28] C. S. Jensen and L. Mark. A Framework for Vacuuming Temporal Databases. Technical report, CS-TR-2516, UMIACS-TR-90-105, Department of Computer Science. University of Maryland, College Park, MD 20742, August 1990.

[29] C. S. Jensen and L. Mark. Queries on Change in an Extended Relational Model. *IEEE Transactions on Knowledge and Data Engineering*, 4(1):to appear, March 1992.

[30] C. S. Jensen, L. Mark, and N. Roussopoulos. Incremental Implementation Model for Relational Databases with Transaction Time. *IEEE Transactions on Knowledge and Data Engineering*, 3(4):to appear, December 1991.

[31] C. S. Jensen, L. Mark, N. Roussopoulos, and T. Sellis. Using Caching, Cache Indexing, and Differential Techniques to Efficiently Support Transaction Time. Technical report, CS-TR-2413, UMIACS-TR-90-25, Department of Computer Science. University of Maryland, College Park, MD 20742, February 1990.

[32] S. Jones, P. Mason, and R. Stamper. LEGOL 2.0: A Relational Specification Language for Complex Rules. *Information Systems*, 4(4):293–305, November 1979.

[33] C. S. Jensen and R. T. Snodgrass. Temporal Specialization. In *Proceedings of the IEEE International Conference on Data Engineering*, February, 1992.

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

[35] M. R. Klopprogge and P. C. Lockemann. Modelling Information Preserving Databases: Consequences of the Concept of Time. In M. Schkolnick and C. Thanos, editors, *Proceedings of the Nineth International Conference on Very Large Data Bases*, pages 399–416, Florence, Italy, 1983.

[36] M. R. Klopprogge. TERM: An Approach to Include the Time Dimension in the Entity-Relationship Model. In *Proceedings of the Second International Conference on the Entity Relationship Approach*, pages 477–512, Washington, DC, October 1981.

[37] M. R. Klopprogge. *Entity and Relationship Histories: A Concept for Describing and Managing Time Variant Information in Databases*. PhD Dissertation, Universitat Karlsruhe, Karlsruhe, FRG, 1983.

[38] C. P. Kolovson and M. R. Stonebraker. Indexing Techniques for Historical Databases. In *Proceedings of the Fifth International Conference on Data Engineering*, pages 127–137, February 1989.

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

[40] H. F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill Advanced Computer Science Series. McGraw-Hill Book Company, second edition, 1991.

[41] B. Lindsay, L. Hass, C. Mohan, H. Pirahesh, and P. Wilms. A Snapshot Differential Refresh Algorithm. In *Proceedings of the ACM SIGMOD Conference*, pages 53–60, May 1986.

[42] N. Lorentzos and R. Johnson. Extending Relational Algebra to Manipulate Temporal Data. *Information Systems*, 13(3): 288–296, 1988.

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

[44] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

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

[46] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 11 Taft Court, Rockville, Maryland 20850, 1983.

[47] K. Marzullo. Tolerating Failures of Continuous-Valued Sensors. *Transactions on Computer Systems*, 8(4):284–304, November 1990.

[48] S. B. Navathe and R. Ahmed. A Temporal Relational Model and a Query Language. *Information Sciences*, 49(1):147–175, October 1989.

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

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

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

[52] R. T. Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, 19(9):35–42, September 1986.

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

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

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

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

[57] A. P. Sheth and J. A. Larson. Federated Database Systems for managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

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

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

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

[61] J. M. Smith and D. C. P. Smith. Database Abstractions: Aggregation and Generalization. *ACM Transactions on Database Systems*, 2(2):105–133, June 1977.

[62] A. Segev and A. Shoshani. Logical Modeling of Temporal Data. In *Proceedings of the ACM SIGMOD Conference*, pages 454–466, May 1987.

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

[64] A. Segev and A. Shoshani. The representation of a temporal data model in the relational environment. In *Proceeding of the 4th International Conference on Statistical and Scientific Database Management*, 1988.

[65] M. R. Stonebraker. The Design of the Postgres Storage System. In *Proceedings of the Thirteenth International Conference on Very Large Data Bases*, pages 289–300, September 1987.

[66] R. Studer. A Conceptual Model for Time. In *Proceedings of the Sixth International Conference on the Entity-Relationship Approach*, New York, NY, November 1987. The ER Institute.

[67] D. Tasker. An Entity-Relationship View of Time. In *Proceedings of the Sixth International Conference on the Entity-Relationship Approach*, New York, NY, November 1987. The ER Institute.

[68] T. J. Teorey and J. P. Fry. *Design of Database Structures*. Prentice-Hall Software Series. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[69] P. M. Thompson. *A Temporal Data Model Based on Accounting Principles*. PhD Dissertation, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, March 1991.

[70] 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 Surveys*, 22(3):237–266, September 1990.

[71] J. D. Ullman. *Database and Knowledge–Base Systems*, Volume I of *Principles of Computer Science*. Computer Science Press, 1803 Research Boulevard, Rockville, MD 20850, 1988.

[72] G. Wiederhold. *Database Design*. McGraw Hill, 1977.