Contents

1 Introduction

Time is an important aspect of all real-world phenomena. Events occur at specific points in time; objects and the relationships among objects exist over time. The ability to model this temporal dimension of the real wold is essential to many computer applications, such as econometrics, banking, inventory control, accounting, law, medical records, land and geographical information systems, and airline reservations.

Conventional databases represent the state of an enterprise at a single moment of time. Although the contents of the database continue to change as new information is added, these changes are viewed as modifications to the state, with the old, out-of-date data being deleted from the database. The current contents of the database may be viewed as a shapshot of the enterprise. In such systems the attributes involving time are manipulated solely by the application programs; the database management system (DBMS) interprets dates as values in the base data types. No conventional system interprets temporal domains when deriving new relations.

Application-independent DBMS support for time-varying information has been an active area of research for about 15 years, with approximately 400 papers generated thus far [Bolour et al. 1982, McKenzie 1986, Soo 1991, Stam & Snodgrass 1988]. This paper attempts to capture and summarize the major concepts, approaches, and implementation strategies that have been generated by that research.

We first examine the time domain: its structure, dimensionality (interestingly, there are several time dimensions) and temporal indeterminacy, followed by issues in representing values in this domain. We demonstrate that time is actually more complex than the spatial domain, as the former's dimensions are non-homogeneous.

Section 3 follows a similar organization in examining how facts may be associated with time. Data modeling issues are first examined, then representational alternatives are explored, with frequent comparisons with space. We briefly consider how facts may be simultaneously associated with both space and time, a common phenomena in land and geographic information systems.

We next consider languages for expressing temporal queries. We illustrate the various types of queries through examples in the temporal query language TQuel, and briefly appraise various standards efforts.

Temporal DBMS implementation is the topic of Sec. 5. We examine the impact to each of the components of a DBMS of adding temporal support, discussing query optimization and evaluation in some detail.

We conclude with a summary of the major accomplishments and disappointments of research into temporal databases.

We omit one major aspect, that of database design, due to lack of space.

2 The Time Domain

In this section we focus on time itself: how it is modeled and how it is represented. The next section will then combine time with facts, to model time-varying information.

2.1 Structure

We initially assume that there is one dimension of time. The distinctions we address here will apply to each of the several dimensions we consider in the next section.

Early work on *temporal logic* centered around two structural models of time, *linear* and *branching* [Van Benthem 1982]. In the linear model, time advances from the past to the future in a totally ordered fashion. In the branching model, also termed the *possible futures* model, time is linear from the past to now, where it then divides into several time lines, each representing a potential sequence of events [Worboys 1990]. Along any future path, additional branches may exist. The structure of branching time is a tree rooted at now. The most general model of time in a temporal logic represents time as an arbitrary set with a partial order imposed on it. Additional axioms introduce other, more refined models of time. For example, linear time can be specified by adding an axiom imposing a total order on this set. Recurrent processes may be associated with a *cyclic* model of time [Chomicki & Imelinski 1989, Lorentzos 1988, Lorentzos & Johnson 1988].

In spatial models, there is much less diversity, and a linear model is generally adequate.

Axioms may also be added to temporal logics to characterize the *density* of the time line [Van Benthem 1982]. Combined with the linear model, *discrete* models of time are isomorphic to the natural numbers, implying that each point in time has a single successor [Clifford & Tansel 1985]. *Dense* models of time are isomorphic to either

the rationals or the reals: between any two moments of time another moment exists. *Continuous* models of time are isomorphic to the reals, i.e., they are both dense and unlike the rationals, contain no "gaps."

In the continuous model, each real number corresponds to a "point" in time; in the discrete model, each natural number corresponds to a nondecomposable unit of time with an arbitrary duration. Such a nondecomposable unit of time is refered to as a *chronon* [Ariav 1986, Clifford & Rao 1987] (other, perhaps less desirable, terms include "time quantum" [Anderson 1982], "moment" [Allen & Hayes 1985], "instant" [Gadia 1986] and "time unit" [Navathe & Ahmed 1987, Tansel & Arkun 1986]). A chronon is the smallest duration of time that can be represented in this model. It is not a point, but a line segment on the time line.

Although time itself is generally perceived to be continuous, most proposals for adding a temporal dimension to the relational data model are based on the discrete time model. Several practical arguments are given in the literature for this preference for the discrete model over the continuous model. First, measures of time are inherently imprecise [Anderson 1982, Clifford & Tansel 1985]. Clocking instruments invariably report the occurrence of events in terms of chronons, not time "points." Hence, events, even so-called "instantaneous" events, can at best be measured as having occurred during a chronon. Secondly, most natural language references to time are compatible with the discrete time model. For example, when we say that an event occurred at 4:30 p.m., we usually don't mean that the event occurred at the "point" in time associated with 4:30 p.m., but at some time in the chronon (perhaps minute) associated with 4:30 p.m. [Anderson 1982, Clifford & Rao 1987, Dyreson & Snodgrass 1992A]. Thirdly, the concepts of chronon and interval allow us to naturally model events that are not instantaneous, but have duration [Anderson 1982]. Finally, any implementation of a data model with a temporal dimension will of necessity have to have some discrete encoding for time (Sec. 2.4).

Space may similarly be regarded as discrete, dense, or continuous. Note that, in all three of these alternatives, two separate space-filling objects cannot be located in the same point in space and time: they can be located in the same place at different times, or at the same time in different places.

Axioms can also be placed on the *boundedness* of time. Time can be bounded orthogonally in the past and in the future. The same applies to models of space.

Models of time may include the concept of *distance* (most temporal logics do not do so, however). Both time and space are *metrics*, in that they have a distance function satisfying four properties: (1) the distance is nonnegative, (2) the distance between any two non-identical elements is non-zero, (3) the distance from time α to time β is identical to the distance from β to α, and (4) the distance from α to γ is equal to or greater than the distance from α to β plus the distance from β to γ (the triangle inequality).

With distance and boundedness, restrictions on range can be applied. The scientific cosmology of the "Big Bang" posits that time begins with the Big Bang, 14 ± 4 billion years ago. There is much debate on when it will end, depending on whether the universe is *open* or *closed* (Hawking provides a readable introduction to this controversy [Hawking 1988]). If the universe is closed then time will have an end when the universe collapses back onto itself in what is called the "Big Crunch." If it is open then time will go on forever.

Similar considerations apply to space. In particular, an open universe implies unbounded space. However, many applications assume a bound as well as a range; geographical information systems don't need to contend with values greater than approximately 70 million meters.

Finally, one can differentiate *relative* time from *absolute* time (more precise terms are *unachored* and *anchored*). For example, "9A.M., January 1, 1992" is an absolute time, whereas "9 hours" is a relative time. This distinction, though, is not as crisp as one would hope, because absolute time is with respect to another time (in this example, midnight, January 1, A.D. 1). We will show in Sec. 2.4 how to exploit this interaction. Relative time differs from distance in that the former has a direction, e.g., one could envision a relative time of -9 hours, whereas a distance is unsigned.

One can also differentiate between relative and absolute space, with the same provisos.

2.2 Dimensionality

Space is multi-dimensional. Land information systems (LIS) manage units of land which are two-dimensional areas [Vrana 1989]. The more general geographical information systems (GIS) must deal with four types of data [Mark et al. 1989].

- *Type 0: point data* Here a two dimensional grid is applied to the surface, and point locations and associated attributes are reported to the nearest center of a grid cell. Note that this data type emphasizes a discrete

model, with the grid size being the space granularity.

- *Type 1: lines* Lines in two-space, such as region boundaries, are explicitly represented.
- *Type 2: coverage data* Coverage data, such as land use, soils or rock types, are represented as two-dimensional regions.
- *Type 3: surface data* Here, a surface in three-space is represented. Generally, the two-dimensional projection is decomposed into quadrants, and surface variation within the quadrant is represented by a polynomial surface. An especially simple representation of surface data records as point data the average elevation within each grid cell.

GIS's must more properly be considered to support at most $2\frac{1}{2}$ dimensions, since arbitrary three-dimensional structures cannot be represented even as surface data, Two very different applications require of a true third dimension. The extensive use of computers in oil and mining exploration has led to the availability, and the need for further processing, of digital geological information. This encompasses three dimensions [Jones 1989], the third dimension descends into the Earth. Secondly, interest in global weather patterns has generated the need for storing and analyzing data about the Earth's atmosphere and oceans [CES 1989], with the third dimension extending from the lithosphere to the troposphere. both up and down.

Time is also multi-dimensional [Snodgrass & Ahn 1986]. *Valid time* concerns the time a fact was true in reality. The valid time of an event is the wall clock time at which the event occurred in the real world, independent of the recording of that event in some database. Valid times can also be in the future, if it is known that some fact will be true at a specified time in the future. *Transaction time* concerns the time the fact was present in the database as stored data. The transaction time (an interval) of an event identifies the transactions that inserted the information about the event into the database and removed this information from the database. As with space, these two dimensions are orthogonal. A data model supporting neither is termed *snapshot*, as it captures only a single snapshot in time of both the database and the enterprise that the database models. A date model supporting only valid time is termed *historical*; one that supports only transaction time is termed *rollback*; and one that supports both valid and transaction time is termed *bitemporal* (*temporal* is a generic term implying some kind of time support).

Figure 1 illustrates a *single* bitemporal relation (i.e., table) composed of a sequence of historical states indexed by transaction time. It is the result of four transactions starting from an empty relation: (1) three tuples (i.e., rows) were added, (2) one tuple was added, (3) one tuple was added and an existing one terminated (logically deleted), and (4) the starting time of a previous tuple [the middle one added in transaction (1)] was changed to a somewhat later time (presumably the original starting time was incorrect) and a recently added tuple (the bottom one) was deleted (presumably it should not have been there in the first place.) Each update operation involves copying the historical state, then applying the update to the newly created state. Of course, less redundant representations than the one shown are possible. While we'll consider only linear time, branching transaction time provides a useful model for *versioning* in computer-aided design tasks [Dittrich & Lorie 1988] such as CAD [Ecklund et al. 1987, Katz et al. 1986] and CASE [Bernstein 1987, Hsieh 1989].

A different depiction that has proven useful is to time-stamp each fact with a *bitemporal element*[1], which is a set of *bitemporal chronons*. Each bitemporal chronon represents a tiny rectangle in valid-time/transaction-time space. Figure 2 shows the bitemporal element associated with the middle tuple of Fig. 1. Historical and rollback databases effectively record *historical chronons* and *rollback chronons*, respectively.

While valid time may be bounded or unbounded (as we saw, cosmologists feel that it is at least bounded in the past), transaction time is bounded on both ends. Specifically, transaction time starts when the database is created (before which time, nothing was stored), and doesn't extend past now (no facts are known to have been stored in the future). Changes to the database state are required to be stamped with the current transaction time. Hence, rollback and bitemporal relations are *append-only*, making them prime candidates for storage on write-once optical disks. As the database state evolves, transaction times grow monotonically. In contrast, successive transactions may mention widely varying valid times. For instance, the fourth transaction in Fig. 1 added information to the database that was transaction time-stamped with time 4, while changing a valid time of one of the tuples to 2.

[1] This term is a generalization of *temporal element*, previously used to denotes a set of single dimensional chronons [Gadia 1988]. An alternative, equally desirable term is *bitemporal lifespan* [Clifford & Croker 1987].

Figure 1: A bitemporal relation

Figure 2: A bitemporal element

The three dimensions in space are truly orthogonal and homogeneous, the one exception being the special treatment sometimes accorded elevation. In contrast, the two time dimensions are not homogeneous; transaction time has a different semantics than valid time. Valid and transaction time *are* orthogonal, though there are generally some application dependent correlations between the two times. As a simple example, consider the situation where a fact is recorded as soon as it becomes valid in reality. In such a *specialized* bitemporal database, termed *degenerate* [Jensen & Snodgrass 1993], valid and transaction time are identical. As another example, if a cloud cover measurement is recorded at most two days after it was valid in reality, and if it takes at least six hours from the measurement time to record the measurement, then such a relation is *delayed strongly retroactively bounded with bounds six hours and two days.*

Multiple transaction times may also be stored in the same relation, termed *temporal generalization* [Jensen & Snodgrass 1993]. These times may also be related to each other, or to the valid time, in various specialized ways. For example, a particular value for the reflectivity of a cloud over a point on the Earth may be recorded by an Earth Sensing Satellite at a particular time. Here, the valid time and transaction time are correlated, and the satellite's database may be considered to be a degenerate bitemporal database. Later, this data is sent to a ground station and stored; the transaction time of the stored data will be different from the valid time; this database may be classified as a *bounded retroactive* database. Later still, the data from several ground stations are merged into a central database, storing the original valid time, the transaction time of the recording into the central database, and the *inherited* transaction time when the data was stored in the ground station database. All three times may be needed, for instance, if data massaging was done with algorithms that were being improved over time. Such multiple transaction time dimensions do not have a spatial analogue.

2.3 Indeterminacy

Information that is *historically indeterminate* can be characterized as "don't know exactly when" information. This kind of information is prevalent; it arises in various situations, including the following.

- *Finer system granularity* — In perhaps most cases, the granularity of the database does not match the precision to which an event time is known. For example, an event time known to within one day and recorded on a system with time-stamps in the granularity of a millisecond happened sometime *during* that day, but during which millisecond is unknown.

- *Imperfect dating techniques* — Many dating techniques are inherently imprecise, such as radioactive and Carbon-14 dating. All clocks have an inherent imprecision [Dyreson & Snodgrass 1992A].

- *Uncertainty in planning* – Projected completion dates are often inexactly specified, e.g., the project will complete three to six months from now.

- *Unknown or imprecise event times* — In general, event times could be unknown or imprecise. For example, if we do now know when an individual was born, the individual's date of birth could be recorded in the database as either unknown (she was born between now and the beginning of time) or imprecise (she was born between now and 100 years ago).

There have been several proposals for adding historical indeterminacy to the time model [Gadia et al. 1992, Kahn & Gorry 1977], as well as more specific work on accommodating multiple time granularities [Ladkin 1987, Wiederhold et al. 1991]. The *possible chronons* model unifies treatment of both aspects [Dyreson & Snodgrass 1992A]. In this model, an event is *determinate* if it is know when (i.e., during which chronon) it occurred. A determinate event cannot overlap two chronons. If it is unknown when an event occurred, but known that it did occur, then the event is historically indeterminate. The indeterminacy refers to the *time* when the event occurred, not *whether* the event did or did not occur.

Two pieces of information completely describe an indeterminate event: a *set of possible chronons* and an *event probability distribution.* A single chronon from the set of possible chronons denotes when the indeterminate event actually occurred. However, it is unknown *which* possible chronon is the actual one. The event probability distribution gives the probability that the event occurred during each chronon in the set of possible chronons.

The implementation of the possible chronons model supports a fixed, minimal chronon size. Multiple granularities are handled by representing the indeterminacy explicitly. For example, if the underlying chronon is a

microsecond and an event is known to within a day, then this indeterminate event would be associated with a set of 86,400,000 possible chronons, and perhaps a uniform event probability distribution.

As a practical matter, events that occurred in the prehistoric past cannot be dated as precisely as events that occur in the present. There is an implicit "telescoping view" of time. Dating of recent events can often be done to the millisecond while events that occurred 400 million years ago can be dated to, perhaps at best the nearest 100,000 years. Dating future events is also problematic. It is impossible to say how many seconds will be between Midnight January 1, 1992 and Midnight January 1, 2300 because we don't know how many leap second will be added to correct for changes in the rotational clock. We can guess at the number of seconds, but "leap shifts" to the current clock are likely to invalidate our guess.

Historical indeterminacy occurs only in valid time. The granularity of a transaction time time-stamp is the smallest inter-transaction time. Transaction times are always determinate since the chronon during which a transaction takes place is always known.

Most of the above may be applied to space. Information that is *spatially indeterminate* can be characterized as "don't know exactly where" information. It is also prevalent, due to granularity concerns, measurement techniques, and unknown or imprecise location specifiers. One could envision an analogous "possible space quanta" model that could capture the variety of spatial indeterminacy. The telescoping view phenomenon also occurs in space, as distant locations are less precisely known.

As with time, a spatial *data granularity* coarser than the *database management system (DBMS) granularity* is often adopted. One of the more common models, Type 0 (Sec. 2.2), covers the two-dimensional space with a grid, with point locations and associated attributes reported to the nearest cell center. In this model, the data is a multiple of the underlying DBMS granularity. For example, the DBMS granularity might be a meter, with all location specifiers being expressed in this unit, while the grid cells may be 2 kilometers on a side.

2.4 Representation

Since time and space are metrics, a system of units is required to represent particular events or locations. A time-stamp or location specifier has a *physical realization* and an *interpretation*. The physical realization is a pattern of bits while the interpretation is the meaning of each bit pattern, that is, the time or location each pattern represents.

2.4.1 Interpretation.

For time, the central unit is the *second*. However, there are at least seven different definitions of this fundamental unit [Dyreson & Snodgrass 1992A].

Apparent solar — 1/86400 of the interval from noon to noon; varies from day to day.

Mean solar (UT0) — 1/86400 of a *mean solar day*, averaged over a year; varies from year to year.

Mean sidereal — 1/86400 of a *mean sidereal day*, measuring the rotation of the Earth with respect to a distant star; varies from year to year.

UT1 — UT0 corrected for polar wander.

UT2 — UT1 corrected for seasonal variations.

Ephemeris — mean solar second for the year 1900; does not vary. This was the standard definition from 1960 to 1967.

International System of Units (SI) — the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of cesium-133 atoms [Petley 1991].

When a range of less than 10,000 years is supported, the differences between these definitions are generally inconsequential, except for the apparent solar second, which varies by 1% over the course of a year. When ranges of several billion years are supported, however, all of these definitions differ significantly.

Different regions of the time line are used by different communities. For example, apparent solar time is important to historians, who care about whether something happened in the daylight or in darkness, as well as

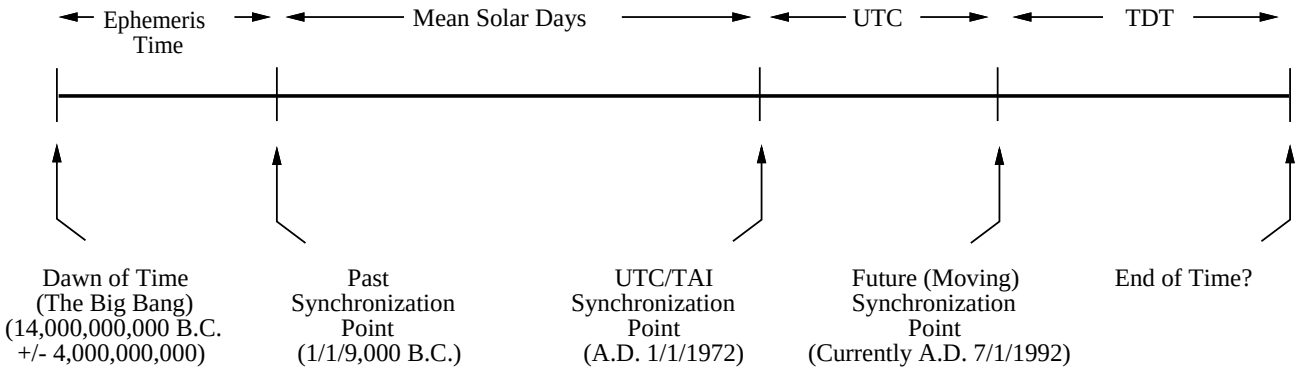

Figure 3: The base-line clock

to users of cadastral (real estate) databases, which utilize civil calendars [Hunter & Williamson 1990]. Ephemeris time is used by astronomers, while the SI second is the basis for radioactive dating used by geochronologists. Because of these different needs, as well as the telescoping view of time, we have proposed a specific temporal interpretation termed the *base-line clock* that constructs a time-line by using different well-defined clocks in different periods. This clock, shown in Fig. 3 (not to scale), partitions the time line into a set of contiguous *periods*. Each period runs on a different clock. A *synchronization point*, where two clocks are correlated, delimits a period boundary. The synchronization points occur at Midnight on the specified date.

From the Big Bang until Midnight January 1, 9000 B.C. the base-line clock runs on ephemeris time. This clock is preferable to the solar clock since ephemeris time is independent of the formation of the Earth and the Solar System. Also, we prefer using the ephemeris clock to the solar clock because an ephemeris year is a fixed duration, unlike the tropical year. For historic events, 9000 B.C. to January 1, 1972, the base-line clock follows the mean solar day clock. Historic events are usually dated with calendars. Calendar dates invariably count days and use an intercalation rule to relate the number of days to longer-term celestial clocks, e.g., the Gregorian calendar relates days to months and tropical years. At Midnight January 1, 1972 the base-line clock switches to Universal Coordinated Time (UTC). Midnight January 1, 1972 is when UTC was synchronized with the SI definition of second and the current system of leap seconds was adopted. The base-line clock runs on UTC until one second before Midnight, July 1, 1992. This is the next time at which a leap second may be added (a leap second will be added on this date according to the latest International Earth Rotation Service bulletin [USNO 1992]). After Midnight July 1, 1992, until the "Big Crunch" or the end of our base-line clock, the base-line clock follows Terrestrial Dynamic Time (TDT), an "idealized atomic time" [Guinot & Seidelmann 1988] based on the SI second, since both UTC and mean solar time are unknown and unpredictable.

The situation is much simpler for space. Here, the (SI) *meter* is the commonly accepted unit, with a single accepted definition, the length of the path traveled by light in vacuum during a time interval of 1/299,792,458 of a second [Petley 1991]. Distance is defined in terms of time, rather than the other way around, because time can be measured more accurately (1 part in 10^{10} over long intervals and 1 part in 10^{15} for between a minute and a day [Quinn 1991, Ramsey 1991]).

The base-line clock and its representation are independent of any calendar. We used Gregorian calendar dates in the above discussion only to provide an informal indication of when the synchronization points occurred. Many calendar systems are in use today; example calendars include academic (years consists of semesters), common fiscal (financial year beginning at the New Year), federal fiscal (financial year beginning the first of October) and time card (8 hour days and 5 day weeks, year-round). The usage of a calendar depends on the cultural, legal, and even business orientation of the user [Soo & Snodgrass 1992A]. A DBMS attempting to support time values must be capable of supporting all the multiple notions of time that are of interest to the user population.

Space also has multiple notions, though with less variability. The metric, U.S., and nautical unit systems are the most prevalent. Both time and space have precisely defined underlying semantics that may be mapped to multiple display formats. The spatial base-line is less complex than that for time; it consists of a single measure, the meter.

SYSTEM	*Size (bytes)*	*Range*	*Granularity Components*	*Number of Needed*	*Bytes Efficiency*	*Space*
OS (several)	4	≈ 136 years	second	1	4	100%
DB2 —`date`	4	10,000 years	day	3	2.9	71%
DB2 —`time`	3	24 hours	second	3	2.2	72%
DB2 —`timestamp`	10	10,000 years	microsecond	7	7.3	73%
SQL2 —`datetime`	20	10,000 years	second	5	4.8	24%
SQL2 —fractional `datetime`	27	10,000 years	microsecond	6	7.3	27%
Low Resolution	8	≈ 36 billion years	second	1	7.6	95%
High Resolution	8	≈ 17400 years	microsecond	3	7.5	93%
Extended Resolution	12	≈ 36 billion years	nanosecond	4	11.3	94%

Table 1: A comparison Of some physical layouts

2.4.2 Physical Realization.

The base-line clock defines the meaning of each time-stamp bit pattern in the physical realization of a time-stamp. The chronons of the base-line clock are the chronons in its constituent clocks. We assume that each chronon is one second in the underlying constituent clock. A chronon may be denoted by an integer, corresponding to a *single (DBMS) granularity*, or it may be denoted by a sequence of integers, corresponding to a *nested granularity*. For example, if we assume a granularity of a second relative to Midnight, January 1, 1980, then in a single granularity the integer 164,281,022 denotes 9:37:02AM March 15, 1985. If we assume a nested granularity of ⟨year, month, day, hour, minute, second⟩, then the sequence ⟨6,3,15,9,37,2⟩ denotes that same time.

Various time-stamps are in use in commercial database management systems and operating systems; a summary is provided in Table 1. This table compares formats from operating systems (specifically Unix, MSDOS, and the MacIntosh operating systems), the database systems DB2 [Date & White 1990], SQL2 [Date 1989A, Melton 1990], and several proposed formats to be discussed shortly. The SQL2 `datetime` time-stamp appears twice in the comparison, once with its optional fractional second precision field set to microseconds, and once without the optional field. The last three representations have recently been proposed, and will be discussed shortly.

Size is the number of bytes devoted to the representation, while *range* refers to the difference between the youngest and oldest time values that can be represented. The *granularity* of a time-stamp is the precision to which a time value can be represented. If a representation has more than one *component*, it is of a nested granularity. The extreme is the DB2 `timestamp` representation, in which the year, month, day, hour, minute, second, and number of microseconds are individually represented. *Space efficiency* is a measure of how much of the representation is actually needed. It is computed as a percentage of the number of bits needed to represent every chronon in the temporal interpretation (DB2 and SQL2 both use the Gregorian calendar temporal interpretation) versus the number of bits devoted to the physical realization. The minimum number of *bytes needed* to store the number of chronons dictated by a time-stamp's granularity and range is shown as a separate column. For instance, SQL2's `datetime` time-stamp uses 20 bytes, but only 4.8 bytes of space are needed to store a range of 10,000 years to the granularity of a second.

The evaluated time-stamps fall into two camps: OS-style time-stamps and database-style time-stamps. OS-style time-stamps have a limited range and granularity; these limitations are dictated by the size of the time-stamp. OS-style time-stamps are maximally space efficient, having a single granularity. The time-stamp itself is merely a count of the number of chronons that have elapsed since the origin in the temporal interpretation. But optimal space efficiency is attained at the expense of some *time efficiency*.

In contrast, database-style time-stamps, as exemplified by the DB2 `timestamp` format, are generally larger than OS-style time-stamps; they have a wider range and finer granularity. But, as a group, they also have poorer space utilization, having a nested granularity. The advantage of representing values separately is that they can be quickly accessed. Extracting the number of years from an OS-style time-stamp is more involved than performing a similar task on an DB2 `timestamp`.

These existing time-stamp representations suffer from inadequate range, too coarse a granularity, excessive space requirements, or a combination of these drawbacks. Finally, none of the time-stamps are able to represent *historical indeterminacy*. Consequently, we recently proposed new time-stamp formats, incorporating features from both the OS-style and database-style time-stamps. These formats combine high space efficiency with high

time efficiency for frequent time-stamp operations [Dyreson & Snodgrass 1992A].

There is a natural tradeoff between range and granularity in time-stamp development. Using the same number of bits, a time-stamp designer can make the granularity coarser to extend the range, or she can limit the range to support finer granularities. These observations imply that a format based on a single 32 bit word is inadequate for our purposes; there are simply not enough bits. Since we wanted to keep the time-stamp formats on 32 bit word boundaries, we allocated the next word increment, 64 bits, to our basic format. Using 64 bits, it is possible to represent all of time (that is, a range of 34 billion years) to the granularity of a second, and a range of historical time to granularities much finer than a second.

There are three basic types of time-stamps: events, spans, and intervals [Soo & Snodgrass 1992A]. We developed three new event time-stamp formats, with different *resolutions*, high, low, and extended. Resolution is a rough measure of a time-stamp precision. The low resolution format can represent times to the precision of a second. High resolution narrows the precision to a microsecond while extended resolution is even more precise; it can represent times to the precision of a nanosecond. High resolution has limited range but extended precision while low resolution has extended range but limited precision. Extended resolution handles those uncommon cases where the user wants both an extended range and an extended precision, at the cost of an extra word of storage.

Interval time-stamp formats are simply two event time-stamps, one for the starting event of the interval and one for the terminating event of the interval. We use this representation because all operations on intervals are actually operations on their delimiting events [Soo et al. 1992]. There are sixteen interval time-stamp formats in toto. The type fields in the delimiting event time-stamps distinguish each format.

Spans are relative times. There are two kinds of spans, fixed and variable [Soo & Snodgrass 1992A]. A *fixed span* is a count of chronons. It represents a fixed duration (in terms of chronons) on the base-line clock between two time values. The fixed span formats use exactly the same layouts as the standard event formats, with a different interpretation. The chronon count in the span representation is independent of the origin, instead of being interpreted as a count from the origin. The sign bit indicates whether the span is positive or negative rather than indicating the direction from the origin.

A *variable span*'s duration is dependent on an associated event. A common variable span is a *month*. The duration represented by a month depends on whether that month is associated with an event in June (30 days) or in July (31 days), or even in February (28 or 29 days). Variable spans use a specialized format requiring 64 bits.

To represent indeterminate events, we added nine formats, three of each resolution. There are three analogous formats for low resolution, and three for extended resolution. The most common, for high resolution with a uniform distribution, requires only 64 bits.

As we have seen in other areas, the considerations for space are similar, yet considerably simpler. A spatial representation of 32 bits (per dimension) for a range required to map the Earth results in a granularity of one decimeter, and of one centimeter for the third dimension (the atmosphere, the oceans, and the Earth's interior) or for restricted areas such as the United States or Europe. Moving up to 64 bits makes spatial indeterminacy representations feasible, and reduces the granularity to a nanometer, which should be adequate for quite a while.

3 Associating Facts with Time

The previous section explored models and representations for the time domain. We now turn to associating time with facts.

3.1 Underlying Data Model

Time has been added to many data models: the entity-relationship model [DeAntonellis et al. 1979, Klopprogge 1981], semantic data models [Hammer & McLeod 1981, Urban & Delcambre 1986], knowledge-based data models [Dayal & Smith 1986], deductive databases [Chomicki & Imelinski 1988, Chomicki & Imelinski 1989, Chomicki 1990, Kabanza et al. 1990], and object-oriented models [Dayal & Wuu 1992, Manola & Dayal 1986, Narasimhalu 1988, Rose & Segev 1991, Sciore 1991, Sciore 1991, Wuu & Dayal 1992]. However, by far the majority of work in temporal databases is based on the relational model. For this reason, we will assume this data model in subsequent discussion.

3.2 Attribute Variability

There are several basic ways in which an attribute associated with an object can interact with time and space. A *time-invariant* attribute [Navathe & Ahmed 1989] does not change over time. Some temporal data models require that the key of a relation be time-invariant; some others identify the object(s) participating in the relation with a time-invariant surrogate, a system-generated, unique identifier of an item that can be referenced and compared for equality, but not displayed to the user [Hall et al. 1976]. Secondly, the *value* of an attribute may be drawn from a temporal domain. An example is *date stamping*, where cadastral parcel records in a land information system contain fields that note the dates of registration of deeds, transfers of titles, and other pertinent historical information [Vrana 1989]. Such temporal domains are termed *user-defined time* [Snodgrass & Ahn 1986]; other than being able to be read in, displayed, and perhaps compared, no special semantics is associated with such domains. Interestingly, most such attributes are time-invariant. For example, the transfer date for a particular title transfer is valid over all time.

An analogous sitation exists for space. There are space-invariant attributes as well as attributes that are drawn from spatial domains, an example being an attribute recording the square feet of a residence, which is a relative spatial measure in two dimensions.

Time- and *space-varying* attributes are more interesting. There are five basic cases to consider.

- The value of an attribute associated with a *space-invariant* object may vary over time, termed *attribute temporality* [Vrana 1989]. An example is percentage of cloud cover over the Earth, which has a single value at each point in time, but varies over time. The value is uniquely specified by the temporal coordinate(s) (either valid time, or a combination of valid and transaction time).

- The value of an attribute associated with a region in space may be time invariant. An example is elevation: the value varies spatially but not temporally (assuming historical time; certainly the elevation varies over geologic time!). The value is unique given spatial coordinates.

- The value of an attribute associated with a region in space may vary over time. An example is the percentage of cloud cover over each 10-kilometer square grid element. Here, the object is identified spatially, and each object is associated with a time-varying sequence of values. Both the temporal and the spatial coordinates are required to uniquely identify a value.

- The boundary lines identifying a cadastral object, e.g., a particular land packet, may vary over time, termed *temporal topology* [Vrana 1989]. The orientation and interaction of spatial objects change over time; such objects could nevertheless have time-invariant attributes, such as initial purchase price. The temporal and spatial coordinates are required to uniquely identify a value, but this identification is indirect, via the topology.

- The final case is the most complex: the value of an attribute, which varies over time, is associated with a cartographic feature that also varies over time [Langran & Chrisman 1988]. An example is the appraised value of a land packet. The appraised value may change yearly, and the boundary of the land packet changes as it is reapportioned and parts sold to others. As with the previous two cases, both temporal and spatial coordinates are required to identify a value, but the temporal coordinate(s) are utilized twice, first to identify a cartographic feature and then to select a particular attribute value associated with that feature.

A further categorization is possible concerning which temporal domains are involved (only valid time, only transaction time, or both) and which spatial domains are involved (two dimensions, $2\frac{1}{2}$ dimensions, or a full three dimensions).

The first case is the traditional domain of temporal DBMS's. The second case is the domain of conventional LIS's and GIS's. While there has been some conceptual work on merging temporal and spatial support, as discussed throughout this paper, current implemented systems are fairly weak in this regard.

3.3 Representational Alternatives

Over two dozen extensions to the relational model to incorporate time have been proposed over the last 15 years. These models may be compared by asking four basic questions: how is valid time represented, how is transaction

Data Model	*Citation*	*Temporal Dimension(s)*	*Homogeneous*	*Identifier*
—	[Snodgrass & Ahn 1986]	both	yes	Ahn
Temporally Oriented Data Model	[Ariav 1986]	both	yes	Ariav
Time Relational Model	[Ben-Zvi 1982]	both	yes	Ben-Zvi
Historical Data Model	[Clifford & Warren 1983]	valid	yes	Clifford-1
Historical Relational Data Model	[Clifford & Croker 1987]	valid	no	Clifford-2
Homogeneous Relational Model	[Gadia 1988]	valid	yes	Gadia-1
Heterogeneous Relational Model	[Gadia & Yeung 1988]	valid	no	Gadia-2
TempSQL	[Gadia 1992]	both	yes	Gadia-3
DM/T	[Jensen et al. 1991]	transaction	N/A	Jensen
LEGOL 2.0	[Jones et al. 1979]	valid	yes	Jones
DATA	[Kimball 1978]	transaction	N/A	Kimball
—	[Lomet & Salzberg 1989]	transaction	N/A	Lomet
Temporal Relational Model	[Lorentzos 1988]	valid	no	Lorentzos
—	[Lum et al. 1984]	transaction	yes	Lum
—	[McKenzie & Snodgrass 1991B]	both	no	McKenzie
Temporal Relational Model	[Navathe & Ahmed 1989]	valid	yes	Navathe
HQL	[Sadeghi 1987]	valid	yes	Sadeghi
HSQL	[Sarda 1990A]	valid	yes	Sarda
Temporal Data Model	[Segev & Shoshani 1987]	valid	yes	Shoshani
TQuel	[Snodgrass 1987]	both	yes	Snodgrass
Postgres	[Stonebraker 1987]	transaction	no	Stonebraker
HQuel	[Tansel 1986]	valid	no	Tansel
Accounting Data Model	[Thompson 1991]	both	yes	Thompson
Time Oriented Databank Model	[Wiederhold et al. 1975]	valid	yes	Wiederhold

Table 2: Temporal Data Models

time represented, how are attribute values represented, and is the model *homogeneous*, i.e., are all attributes restricted to be defined over the same valid time(s) [Gadia 1988].

3.3.1 Data Models.

Table 2 lists most of the temporal data models that have been proposed to date. If the model is not given a name, we appropriate the name given the associated query language, where available. Many models are described in several papers; the one referenced is the initial journal paper in which the model was defined. Some models are defined only over valid time or transaction time; others are defined over both. Whether the model is homogeneous is indicated in the next column. Tuple-timestamped data models, to be identified in the next section, and data models that use single chronons as time-stamps are of necessity homogeneous. The issue of homogeneity is not relevant for those data models supporting only transaction time. The last column indicates a short identifier which denotes the model; the table is sorted on this column.

We omit a few intermediate data models, specifically Gadia's multihomogeneous model [Gadia 1986], which was a precursor to his heterogeneous model (Gadia-2), and Gadia's two-dimensional temporal relational database model [Bhargava & Gadia 1989], which is a precursor to Gadia-3. We also do not include the data model used as the basis for defining temporal relational completeness [Tuzhilin and Clifford 1990], because it is a generic data model that does not force decisions on most of the aspects to be discussed here.

More detail on these data models, including a comprehensive comparison, may be found elsewhere [McKenzie & Snodgrass 1991A, Snodgrass 1987].

	Single chronon	Interval (pair of chronons)	Historical element (set of chronons)
Time-stamped attribute values	Lorentzos Thompson	Gadia-2 McKenzie Tansel	Clifford-2 Gadia-1 Gadia-3
Time-stamped tuples	Ariav Clifford-1 Lum Sadeghi Shoshani Wiederhold	Ahn Ben-Zvi Jones Navathe Sarda Snodgrass	

Table 3: Representation of Valid Time

3.3.2 Valid Time.

Two fairly orthogonal aspects are involved in representing valid time. First, is valid time represented with single chronon identifiers (i.e., event time-stamps, Sec. 2.4), with intervals (i.e., as interval time-stamps, Sec. 2.4.2), or as historical elements (i.e., as a set of chronon identifiers, or equivalently as a finite set of intervals)? Second, is valid time associated with entire tuples or with individual attribute values? A third alternative, associating valid time with sets of tuples, i.e., relations, has not been incorporated into any of the proposed data models, primarily because it lends itself to high data redundancy. The data models are elevated on these two aspects in Table 3. Interestingly, only one quadrant, time-stamping tuples with an historical element, has not been considered (but see Sec. 3.3.5)

3.3.3 Transaction Time.

The same general issues are involved in transaction time, but there are about twice as many alternatives. Transaction time may be associated with

- a single chronon, which implies that tuples inserted on each transaction signify the termination (logical deletion) of previously current tuples with identical keys, with the time-stamps of these previously recorded tuples not requiring change.
- an interval. A newly inserted tuple would be associated with the interval starting at now and ending at the special value U.C., *until-changed*.
- three chronons. Ben-Zvi's model records (1) the transaction time when the valid start time was recorded, (2) the transaction time when the valid stop time was recorded, and (3) the transaction time when the tuple was logically deleted.
- a transaction-time element, which is a set of not-necessarily-contigous chronons.

Another issue concerns whether transaction time is associated with individual attribute values, with tuples, or with sets of tuples.

The choices made in the various data models are characterized in Table 4. Gadia-3 is the only data model to time-stamp attribute values; it is difficult to efficiently implement this alternative directly. Gadia-3 also is the only data model that uses transaction-time elements (but see Sec. 3.3.5). Ben-Zvi is the only one to use three transaction-time chronons. All of the rows and columns are represented by at least one data model.

3.3.4 Attribute Value Structure.

The final major decision to be made in designing a temporal data model is how to represent attribute values. There are six basic alternatives. In some models, the time-stamp appears as an explicit attribute; we do not consider such attributes in this analysis.

	Single chronon	Interval (pair of chronons)	Three Chronons	Transaction-time element (set of chronons)
Time-stamped attribute values				Gadia-3
Time-stamped tuples	Ariav Jensen Kimball Lomet	Snodgrass Stonebraker	Ben-Zvi	
Time-stamped sets of tuples	Ahn Thompson	McKenzie		

Table 4: Representation of Transaction Time

- *Atomic valued*—values do not have any internal structure. Ariav, Ben-Zvi, Clifford-1, Jensen, Jones, Kimball, Lomet, Lorentzos, Lum, Navathe, Sadeghi, Sarda, Shoshani, Snodgrass, Stonebraker and Thompson all adopt this approach. Tansel allows atomic values, as well as others, listed below.
- *Set valued*—values are sets of atomic values. Tansel supports this representation.
- *Functional, atomic valued*—values are functions from the (generally valid) time domain to the attribute domain. Clifford-2, Gadia-1, Gadia-2, and Gadia-3 adopt this approach.
- *Ordered pairs*—values are an ordered pair of a value and a (historical element) time-stamp. McKenzie adopts this approach.
- *Triplet valued*—values are a triple of attribute value, valid from time, and value to time. This is similar to the ordered pairs representation, except that only one interval may be represented. Tansel supports this representation.
- *Set-triplet valued*—values are a set of triplets. This is more general than ordered pairs, in that more than one value can be represented, and more general than functional valued, since more than one attribute value can exist at a single valid time [Tansel 1986]. Tansel supports this representation.

In the conventional relational model, if attributes are atomic-valued, they are considered to be in *first normal form* [Codd 1972A]. Hence, only the data models placed in the first category may be considered to be strictly in first normal form. However, in several of the other models, the non-atomicity of attribute values comes about because time is added. It turns out that the property of "all snapshots are in first normal form" is closely associated with homogeneity (Sec. 3.3.1).

3.3.5 Separating Semantics from Representation.

It is our contention that focusing on data *presentation* (how temporal data is displayed to the user), on data *storage*, with its requisite demands of regular structure, and on efficient *query evaluation* has complicated the central task of capturing the time-varying semantics of data. The result has been, as we have seen, a plethora of incompatible data models, with many query languages (Sec. 4.1), and a corresponding surfeit of database design and implementation strategies that may be employed across these models.

We advocate instead a very simple *conceptual temporal data model* that captures the essential semantics of time-varying relations, but has no illusions of being suitable for presentation, storage, or query evaluation. Existing data model(s) may be used for these latter tasks. This conceptual model time-stamps tuples with bitemporal elements, sets of *bitemporal chronons*, which are rectangles in the two-dimensional space spanned by valid time and transaction time (see Fig. 2). Because no two tuples with mutually identical explicit attribute values (termed *value-equivalent* tuples) are allowed in a bitemporal relation instance, the full time history of a fact is contained in a single tuple.

In Table 3, the conceptual temporal data model occupies the unfilled entry corresponding to time-stamping tuples with historical elements, and occupies the entry in Table 4 corresponding to time-stamping tuples with transaction-time elements. An important property of the conceptual model, shared with the conventional relational model but not held by the representational models, is that relation instances are semantically unique; distinct instances model different realities and thus have distinct semantics.

It is possible to demonstrate equivalence mappings between the conceptual model and several *representational* models [Jensen et al. 1992]. Mappings have already been demonstrated for three data models: Gadia-3 (attribute time-stamping), Jensen (tuple time-stamping with a single transaction chronon), and Snodgrass (tuple time-stamping, with interval valid and transaction times). This equivalence is based on *snapshot equivalence*, which says that two relation instances are equivalent if all their snapshots, taken at all times (valid and transaction), are identical. Snapshot equivalence provides a natural means of comparing rather disparate representations. An extension to the conventional relational algebraic operators may be defined in the conceptual data model, and can be mapped to analogous operators in the representational models. Finally, we feel that the conceptual data model is the appropriate location for database design and query optimization.

In essence, we advocate moving the distinction between the various existing temporal data models from a semantic basis to a physical, performance-relevant basis, utilizing the proposed conceptual data model to capture the time-varying semantics. Data presentation, storage representation, and time-varying semantics should be considered in isolation, utilizing different data models. Semantics, specifically as determined by logical database design, should be expressed in the conceptual model. Multiple presentation formats should be available, as different applications require different ways of viewing the data. The storage and processing of bitemporal relations should be done in a data model that emphasizes efficiency.

4 Querying

A data model consists of a set of objects with some structure, a set of constraints on those objects, and a set of operations on those objects [Tsichritzis & Lochovsky 1982]. In the two previous sectins we have investigated in detail the structure of and constraints on the objects of temporal relational databases, the temporal relation. In this section, we complete the picture by discussing the operations, specifically temporal query languages.

Many temporal query languages have been proposed. In fact, it seems obligatory for each researcher to define their own data model and query language (we return to this issue at the end of this section). We first summarize the twenty-odd query languages that have been proposed thus far. We then briefly discuss the various activities that should be supported by a temporal query language, using a specific language in the examples. Finally, we touch on work being done in the standards arena that is attempting to bring highly needed order to this confusing collection of languages.

We do not consider the related topic of *temporal reasoning* (also termed *inferencing* or *rule-based search*) [Chomicki 1990, Kahn & Gorry 1977, Karlsson 1986, Lee et al. 1985, Sheng 1984, Sripada 1988] that uses artificial intelligence techniques to perform more sophisticated analyses of temporal relationships, generally with much lower query processing efficiency.

4.1 Language Proposals

Table 5 lists the major temporal query language proposals to date. While many of these languages each have several associated papers, we have indicated the most comprehensive or most readily available reference. The underlying data model is a reference to Table 2. The next column lists the conventional query language the temporal proposal is based on, from the following.

SQL Structured Query Language [Date 1989B], a tuple calculus-based language; the lingua franca of conventional relational databases.

Quel The tuple calculus based query language [Held et al. 1975] originally defined for the Ingres relational DBMS [Stonebraker et al. 1976].

QBE Query-by-Example [Zloof 1975], a domain calculus based query language.

Name	*Citation*	*Underlying Data Model*	*Based On*	*Formal Semantics*	*Equivalent Algebra*
HQL	[Sadeghi et al. 1987]	Sadeghi	DEAL	partial	[Sadeghi 1987]
HQuel	[Tansel 1986]	Tansel	Quel	yes	[Tansel 1986]
HSQL	[Sarda 1990A]	Sarda	SQL	no	[Sarda 1990B]
HTQuel	[Gadia 1988]	Gadia-1	Quel	yes	[Gadia 1988]
Legol 2.0	[Jones et al. 1979]	Jones	relational algebra	no	N/A
Postquel	[Stonebraker et al. 1990]	Stonebraker	Quel	no	none
TDM	[Segev & Shoshani 1987]	Shoshani	SQL	no	none
Temporal Relational Algebra	[Lorentzos 1988]	Lorentzos	relational algebra	yes	N/A
TempSQL	[Gadia 1992]	Gadia-3	SQL	partial	none
Time-By-Example	[Tansel et al. 1989]	Tansel	QBE	yes	[Tansel 1986]
TOSQL	[Ariav 1986]	Ariav	SQL	no	none
TQuel	[Snodgrass 1987]	Snodgrass	Quel	yes	[McKenzie & Snodgrass 1991B]
TSQL	[Navathe & Ahmed 1989]	Navathe	SQL	no	none
—	[Ben-Zvi 1982]	Ben-Zvi	SQL	yes	[Ben-Zvi 1982]
—	[Clifford & Warren 1983]	Clifford-1	IL_s	yes	N/A
—	[Clifford & Croker 1987]	Clifford-2	relational algebra	yes	N/A
—	[Gadia 1986]	Gadia-2	Quel	no	none
—	[Jensen & Mark 1991]	Jensen	relational algebra	yes	N/A
—	[McKenzie & Snodgrass 1991B]	McKenzie	relational algebra	yes	N/A
—	[Thompson 1991]	Thompson	relational algebra	yes	N/A
—	[Tuzhilin and Clifford 1990]	*several*	relational algebra	yes	N/A

Table 5: Temporal query languages

IL_s An intensional logic formulated in the context of computational linguistics [Montague 1973].

relational algebra A procedural language with relations as objects [Codd 1972B].

DEAL An extension of the relational algebra incorporating functions, recursion, and deduction [Deen 1985].

Most of the query languages have a formal definition. Some of the calculus-based query languages have an associated algebra that provides a means of evaluating queries.

More comprehensive comparisons may be found elsewhere [McKenzie & Snodgrass 1991A, Snodgrass 1987].

4.2 Types of Temporal Queries

We now examine the types of temporal queries that each of the above-listed query languages support to varying degrees. We'll use TQuel [Snodgrass et al. 1993] in the examples, as it is the most completely defined temporal language [Snodgrass 1987].

4.2.1 Schema Definition.

We will use one relation in these examples.

Example. *Define the* `Cities` *relation.*

```
create persistent interval Cities(Name is char, State is char,
        Population is I4, IncorporationDate is event,
        Size is area)
```

`Cities` has five explicit attributes: two strings (denoted by `char`), a 4-byte integer (denoted by `I4`), a user-defined event, and a user-defined area. The *`persistent`* and *`interval`* keywords specify a bitemporal relation, with four implicit time-stamp attributes: a valid start time, a valid end time, a transaction start time, and a transaction end time. The valid time-stamps define the interval when the attribute values were true in reality, and the transaction time-stamps specify the interval when the information was current in the database. □

4.2.2 Quel Retrieval Statements.

Since TQuel is a strict superset of Quel, all Quel queries are also TQuel queries [Snodgrass 1987]. Here we give one such query, as a review of Quel.

The query uses a *`range`* statement to specify the *tuple variable* `C`, which will remain active for use in subsequent queries.

Example. *What is the current population of the cities in Arizona?*

```
range of C is Cities

retrieve (C.Name, C.Population)
where C.State = "Arizona"
```

The *target list* specifies which attributes of the qualifying tuples are to be retained in the result, and the where clause specifies which underlying tuples from the underlying relation(s) qualify to participate in the query. Because the defaults have been defined appropriately, each TQuel query yields the same result as its Quel counterpart. □

4.2.3 Rollback (Transaction-time Slice).

The *`as of`* clause rolls back a transaction-time relation (consisting of a sequence of snapshot relation states) or a bitemporal relation (consisting of a sequence of valid-time relation states) to the state that was current at the specified transaction time. It can be considered to be a transaction time analogue of the where clause, restricting the underlying tuples that participate in the query.

Example. *What was the population of Arizona's cities as best known in 1980?*

```
retrieve (C.Name, C.Population)
where C.State = "Arizona"
as of begin of |January 1, 1980|
```

This query uses an event temporal constant, delimited with vertical bars, "$|\cdots|$". TQuel supports multiple calendars and calendric systems [Soo & Snodgrass 1992A, Soo & Snodgrass 1992B, Soo et al. 1992]. In this case, the default is the Gregorian calendar with English month names. □

4.2.4 Valid-time Selection.

The *`when`* clause is the valid-time analogue of the where clause: it specifies a predicate on the event or interval time-stamps of the underlying tuples that must be satisfied for those tuples to participate in the remainder of the processing of the query.

Example. *What was the population of the cities in Arizona in 1980 (as best known right now)?*

```
retrieve (C.Name, C.Population)
where C.State = "Arizona"
when C overlap |January 1, 1980|
as of present
```

A careful examination of the prose statement of this and the previous query illustrates the fundamental difference between valid time and transaction time. The *`as of`* clause selects a particular transaction time, and thus

rolls back the relation to its state stored at the specified time. Corrections stored after that time will not be incorporated into the retrieved result. The particular **when** statement given here selects the facts *valid in reality* at the specified time. All corrections stored up to the time the query was issued are incorporated into the result. In this case, all corrections made after 1980 to the census of 1980 will be included in the resulting relation. □

Example. *What was the population of the cities in Arizona in 1980, as best known at that time?*

```
retrieve (C.Name, C.Population)
where C.State = "Arizona"
when C overlap |January 1, 1980|
as of |January 1, 1980|
```

The result of this query, executed any time after January 1, 1980, will be identical to the result of the first query specified, "*What is the current population of the cities in Arizona?*", executed exactly on midnight of that date. □

4.2.5 Valid Time Projection.

The **valid** clause serves the same purpose as the target list; it specifies some aspect of the derived tuples, in this case, the valid time of the derived tuple.

Example. *For what date is the most recent information on Arizona's cities valid?*

```
retrieve (C.All)
valid at begin of C
where C.State = "Arizona"
```

This query extracts relevant events from an interval relation. □

4.2.6 Aggregates.

As TQuel is a superset of Quel, all Quel aggregates are still available [Snodgrass et al. 1993].

Example. *What is the current population of Arizona?*

```
retrieve (sum(C.Population where C.State = "Arizona"))
```

Note that this query only counts city residents. □

This query applied to a bitemporal relation yields the same result as its conventional analogues, that is, a single value. With just a little more work, we can extract its time-varying behavior.

Example. *How has the population of Arizona flucuated over time?*

```
retrieve (sum(C.Population where C.State = "Arizona"))
when true
```
□

New, temporally-oriented aggregates are also available in TQuel. One of the most useful computes the interval when the argument was rising in value. This aggregate may be used wherever an interval expression is expected.

Example. *For each growing city, when did it start growing?*

```
retrieve (C.Name)
valid at begin of rising(C.Population by C.Name
                         where C.State = "Arizona")
```
□

4.2.7 Historical Indeterminacy.

Indeterminacy aspects can hold for individual tuples, or for all the tuples in a relation.

Example. *The information in the* `Cities` *relation is known only to within thirty days.*

```
modify cities to indeterminate span = %30 days%
```

%30 days% is a *span*, an unanchored length of time [Soo & Snodgrass 1992B]. Spans can be created by taking the difference of two events; spans can also be added to an event to obtain a new event. □

Example. *What cities in Arizona definitely had a population over 500,000 at the beginning of 1980?*

```
retrieve (C.Name)
where C.State = "Arizona" and C.Population > 500000
when C overlap |January 1, 1980|
```

The default is to only retrieve tuples that fully satisfy the predicate. This is consistent with the Quel semantics. □

Historical indeterminacy enters queries at two places, specifying the *credibility* of the underlying information to be utilized in the query, and specifying the *plausibility* of temporal relationships expressed in the when and valid clauses. We'll only illustrate plausibility here.

Example. *What cities in Arizona had a population over 500,000 probably at the beginning of 1980?*

```
retrieve (C.Name)
where C.State = "Arizona" and C.Population > 500000
when C overlap |January 1, 1980| probably
```

Here, "probably" is syntactic sugar for "*with plausibility* 70". □

4.2.8 Schema Evolution.

Often the database schema needs to be modified to accommodate a changing set of applications. The *modify* statement has several variants, allowing any previous decision to be later changed or undone. Schema evolution involves transaction time, as it concerns how the data is stored in the database [McKenzie & Snodgrass 1990]. As an example, changing the type of a relation from a bitemporal relation to an historical relation will cause future intermediate states to not be recorded; states stored when the relation was a temporal relation will still be available.

Example. *The* Cities *relation should no longer record all errors.*

```
modify Stocks to not persistent
```

□

4.3 Standards

Support for time in conventional data base systems (e.g., [Oracle 1987, Tandem 1983]) is entirely at the level of user-defined time (i.e., attribute values drawn from a temporal domain). These implementations are limited in scope and are, in general, unsystematic in their design [Date & White 1990, Date 1988]. The standards bodies (e.g., ANSI) are somewhat behind the curve, in that SQL includes no time support. Date and time support very similar to that in DB2 is currently being proposed for SQL2 [Melton 1990]. SQL2 corrects some of the inconsistencies in the time support provided by DB2 but inherits its basic design limitations [Soo & Snodgrass 1992B].

An effort is currently underway within the research community to consolidate approaches to temporal data models and calculus-based query languages, to achieve a consensus extension to SQL and an associated data model upon which future research can be based. This extension is termed the *Temporal Structured Query Language*, or TSQL (not to be confused with an existing language proposal of the same name).

5 System Architecture

The three previous sections in concert sketched the boundaries of a temporal data model, by examining the temporal domain, how facts may be associated with time, and how temporal information may be queried. We now turn to the implementation of the temporal data model, as encapsulated in a temporal DBMS.

Figure 4: Components of a data base management system

Adding temporal support to a DBMS impacts virtually all of its components. Figure 4 provides a simplified architecture for a conventional DBMS. The *database administrator (DBA)* and her staff design the database, producing a physical schema specified in a *data definition language (DDL)*, which is processed by the *DDL Compiler* and stored, generally as system relations, in the *System Catalog*. Users prepare queries, either ad hoc or embedded in procedural code, which is submitted to the *Query Processor*. The query is first lexically and syntactically analyzed, using information from the system catalog, then optimized for efficient execution. A query evaluation plan is sent to the *Query Evaluator*. For ad hoc queries, this occurs immediately after processing; for embedded queries, this occurs when the cursor associated with a particular query is opened. The query evaluator is usually an interpreter for a form of the relational algebra annotated with access methods and operator strategies. While evaluating the query, this component accesses the database via a *Stored Data Manager*, which implements concurrency control, transaction management, recovery, buffering, and the available data access methods.

In the following, we visit each of these components in turn, reviewing what changes need to be made to add temporal support.

5.1 DDL Statements

Relational query languages such as Quel and SQL actually do much more than simply specify queries; they also serve as data definition languages (e.g., through Quel's `create` statement, c.f., Sec. 4.2.1) and as data manipulation languages (e.g., through SQL's `INSERT`, `DELETE` and `UPDATE` statements). The changes to support time involve adding temporal domains, such as event, interval, and span [Soo & Snodgrass 1992B] and adding constructs to specify support for transaction and valid time, such as the TQuel keywords `persistent` and `interval`.

5.2 System Catalog

The big change here is that the system catalog must consist of transaction-time relations. Schema evolution concerns only the recording of the data, and hence does not involve valid time. The attributes and their domains, the indexes, even the names of the relations all vary over transaction time.

5.3 Query Processing

There are two aspects here, one easily extended (language analysis) and one for which adding temporal support is much more complex (query optimization).

Language analysis needs to consider multiple calendars, which extend the language with calendar-specific functions. An example is `monthof`, which only makes sense in calendars for which there are months. The changes to language processing, primarily involving modifications to semantic analysis (name resolution and type checking), have been worked out in some detail [Soo et al. 1992].

Optimization of temporal queries is substantially more involved than that for conventional queries, for several reasons. First, optimization of temporal queries is more critical, and thus easier to justify expending effort on, than conventional optimization. The relations that temporal queries are defined over are larger, and are growing monotonically, with the result that unoptimized queries take longer and longer to execute. This justifies trying harder to optimize the queries, and spending more execution time to perform the optimization.

Second, the predicates used in temporal queries are harder to optimize [Leung & Muntz 1990, Leung & Muntz 1991A]. In traditional database applications, predicates are usually equality predicates (hence the prevalence of equi-joins and natural joins); if a less-than join is involved, it is rarely in combination with other less-than predicates. On the other hand, in temporal queries, less-than joins appear more frequently, as a conjunction of several inequality predicates. As an example, the TQuel ***`overlap`*** operator is translated into two less-than predicates on the underlying time-stamps. Optimization techniques in conventional databases focus on equality predicates, and often implement inequality joins as cartesian products, with their associated inefficiency.

And third, there is greater opportunity for query optimization when time is present [Leung & Muntz 1991A]. Time advances in one direction: the time domain is continuous expanding, and the most recent time point is the largest value in the domain. This implies that a natural clustering or sort order will manifest itself, which can be exploited during query optimization and evaluation. The integrity constraint $beginof(t) < endof(t)$ holds for every time-interval tuple t. Also, for many relations it is the case that the intervals associated with a key are contiguous in time, with one interval starting exactly when the previous interval ended. An example is salary data, where the intervals associated with the salaries for each employee are contiguous. *Semantic query optimization* can exploit these integrity constraints, as well as additional ones that can be inferred [Shenoy & Özsoyoğlu 1989].

In this section, we first examine local query optimization, of a single query, then consider global query optimization, of several queries simultaneously. Both involve the generation of a *query evaluation plan*, which consists of an algebraic expression annotated with access methods.

5.3.1 Local Query Optimization.

A single query can be optimizing by replacing the algebraic expression with an equivalent one that is more efficient, by changing an access method associated with a particular operator, or by adopting a particular implementation of an operator. The first alternative requires a definition of equivalence, in the form of a set of tautologies. Tautologies have been identified for the conventional relational algebra [Enderton 1977, Smith & Chang 1975, Ullman 1988], as well as for many of the algebras listed in Table 5. Some of these temporal algebras support the standard tautologies, enabling existing query optimizers to be used.

To determine which access method is best for each algebraic operator, *meta-data*, that is, statistics on the stored temporal data, and *cost models*, that is, predictors of the execution cost for each operator implementation/access method combination, are needed. Temporal data requires additional meta-data, such as *lifespan* of a relation (the time interval over which the relation is defined), the lifespans of the tuples, the surrogate and tuple arrival distributions, the distributions of the time-varying attributes, regularity and granularity of temporal data, and the frequency of null values, which are sometimes introduced when attributes within a tuple aren't synchronized [Gunadhi et al. 1989]. Such statistical data may be updated by random sampling or by a scan through the entire relation.

There has been some work in developing cost models for temporal operators. An extensive analytical model has been developed and validated for TQuel queries [Ahn & Snodgrass 1988, Ahn & Snodgrass 1989], and *selectivity estimates* on the size of the results of various temporal joins have been derived [Gunadhi & Segev 1989, Gunadhi et al. 1989].

5.3.2 Global Query Optimization.

In global query optimization, a collection of queries is simultaneously optimized, the goal being to produce a single query evaluation plan that is more efficient than the collection of individual plans [Satoh et al. 1985, Sellis 1986]. A state transition network appears to be the best way to organize this complex task [Jensen et al. 1993]. *Materialized views* [Blakeley et al. 1986, Blakeley & Martin 1990, Roussopoulos 1982, Roussopoulos 1991] are expected to play an important role in achieving high performance in the face of temporal databases of monotonically increasing size. For an algebra to utilize this approach, incremental forms of the operators are required (c.f., [Jensen et al. 1991, McKenzie 1988]).

5.4 Query Evaluation

Achieving adequate efficiency in query evaluation is very important. We first examine operations on time-stamps, some of which are critical to high performance. We then review a study that showed that a straightforward implementation would not result in reasonable performance. Since joins are the most expensive, yet very common, operations, they have been the focus of a significant amount of research. Finally, we will examine the many temporal indexes that have been proposed.

5.4.1 Domain Operations.

In Sec. 2 we outlined the domain of time-stamps. Query evaluation performs input, comparison, arithmetic, and output operations on values of this domain. Ordered by contribution to execution efficiency, they are comparison (which is often in the "inner loop" of join processing), arithmetic (which is most often performed during creation of the resulting tuple), output (which is only done when transferring results to the screen or to paper, a much slower process than execution or even disk I/O), and finally input (which is done exactly once per value). However, the SQL2 format, with its five components (see Table 1) is optimized for the relatively infrequent operations of (Gregorian) input and output, and is rather slow at comparison and addition. The proposed formats instead optimize comparison at the expense of input and output. For a sequence of operations that inputs two relations and computes and outputs the overlap (favoring input and output more than expected), the high resolution format is more efficient, with only 50 tuples, than the SQL2 format, even though the high resolution format has much greater range and smaller granularity [Dyreson & Snodgrass 1992B].

Performing these operations efficiently in the presence of historical indeterminacy is more challenging. For the default range credibility and ordering plausibility, and for comparing events whose sets of possible chronons do not overlap, there is little overhead even when historical indeterminacy is supported [Dyreson & Snodgrass 1992A]. The average *worse case* for comparison, over all plausibilities, when the sets of possible chronons overlap signficantly, is less than 100 microseconds on a Sun-4, or about the time to transfer a 100-byte tuple from disk.

5.4.2 A Straightforward Implementation.

The importance of efficient query optimization and evaluation for temporal databases was underscored by an initial study that analyzed the performance of a brute-force approach to adding time support to a conventional DBMS. In this study, the university Ingres DBMS was extended in a minimal fashion to support TQuel querying [Ahn & Snodgrass 1986]. The results were very discouraging for those who might have been considering such an approach. Sequential scans, as well as access methods such as hashing and ISAM, suffered from rapid performance degradation due to ever-growing overflow chains. Because adding time creates multiple tuple versions with the same key, reorganization does not help to shorten overflow chains. The objective of work in temporal query evaluation then is to avoid looking at all of the data, because the alternative implies that queries will continue to slow down as the database accumulates facts.

There were four basic responses to this challenge. The first was a proposal to separate the *historical* data, which grew monotonically, from the *current* data, whose size was fairly stable and whose accesses were more frequent [Lum et al. 1984]. This separation, termed *temporal partitioning*, was shown to significantly improve performance of some queries [Ahn & Snodgrass 1988], and was later generalized to allow multiple cached states, which further improves performance [Jensen et al. 1993]. Second, new query optimization strategies were proposed (Sec. 5.3).

Third, new join algorithms, to be discussed next, were proposed. And finally, new temporal indexes, also to be discussed, were proposed.

5.4.3 Joins.

Three kinds of temporal joins have been studied: binary joins, multiway joins, and joins executed on multiprocessors.

A wide variety of binary joins have been considered, including *time-join*, *time-equijoin* (TE-join) [Clifford & Croker 1987], *event-join*, *TE-outerjoin* [Gunadhi & Segev 1991], *contain-join*, *contain-semijoin*, *intersect-join* [Leung & Muntz 1991A], and *contain-semijoin* [Leung & Muntz 1992]. The various algorithms proposed for these joins have generally been extensions to nested loop or merge joins that exploit sort orders or local workspace.

Leung argues that a checkpoint index (Sec. 5.4.4) is useful when stream processing is employed to evaluate both two-way and multi-way joins [Leung & Muntz 1992].

Finally, Leung has explored in depth partitioning strategies and temporal query processing on multiprocessors [Leung & Muntz 1991B].

5.4.4 Temporal Indexes.

Conventional indexes have long been proposed to reduce the need to scan an entire relation to access a subset of its tuples. Indices are even more important in temporal relations that grow monotonically in size. In table 6 we summarize the temporal index structures that have been proposed to date. Most of the indexes are based on B^+-Trees [Comer 1979], which index on values of a single key; the remainder are based on R-Trees [Guttman 1984], which index on ranges (intervals) of multiple keys. There has been considerable discussion concerning the applicability of point-based schemes for indexing interval data. Some argue that structures that explicitly accommodate intervals, such as R-Trees and their variants, are preferable; others argue that mapping intervals to their endpoints is efficient for spatial search [Lomet 1991].

If the structure requires that exactly one record with each key value exist at any time, or if the data records themselves are stored in the index, then it is designated a *primary* storage structure; otherwise, it can be used either as a primary storage structure or as a secondary index. The checkpoint index is associated with a particular indexing condition, making it suitable for use during the processing of queries consistent with that condition.

A majority of the indexes are tailored to transaction time, exploiting the append-only nature of such information. Most utilize as a key the valid-time or transaction-time interval (or possibly both, in the case of the Mixed Media R-Tree). Lum's index doesn't include time at all; rather it is a means of accessing the history, represented as a linked list of tuples, of a key value. The Append-only Tree indexes the transaction-start time of the data, and the Lop-Sized B^+-Tree is most suited for indexing events such as bank transactions. About half the indexes utilize only the time-stamp as a key; some include a single non-temporal attribute; and the two based on R-Trees can exploit its multi-dimensionality to support an arbitrary number of non-temporal attributes. Of the indexes supporting non-temporal keys, most treat such keys as a true separate dimension, the exceptions being the indexes discussed by Ahn, which support a single composite key with the interval as a component.

While preliminary performance studies have been carried out for each of these indexes in isolation, there has been little effort to compare them concerning their space and time efficiency. Such a comparison would have to consider the differing abilities of each (those supporting no non-temporal keys would be useful for doing temporal cartesian products, but perhaps less useful for doing temporal joins that involved equality predicates on non-temporal attributes) as well as various underlying distributions of time and non-temporal keys (the indexes presume various non-uniform distributions to achieve their performance gains over conventional indexes, which generally assume a uniform key distribution).

5.5 Stored Data Manager

We examine three topics, storage structures (including page layout), concurrency control, and recovery. Page layout for temporal relations is more complicated than conventional relations if non-first normal form (i.e., non-atomic attribute values) are adopted, as is proposed in many of the temporal data models listed in Sec. 3.3.1. Often such attributes are stored as linked lists, for example representing a valid-time element (set of valid-time

Name	*Citation*	*Based On*	*Primary/ Secondary*	*Temporal Dimension(s)*	*Temporal Key(s)*	*Non-Temporal Key(s)*
Append-only Tree	[Gunadhi & Segev 1993]	B^+-Tree	primary	transaction	event	0
Checkpoint Index	[Leung & Muntz 1992]	B^+-Tree	secondary	transaction	event	0
Lop-Sided B^+-Tree	[Kolovson 1990]	B^+-Tree	both	transaction	event	0
Monotonic B^+-Tree	[Elmasri et al. 1992]	Time Index	both	transaction	interval	0
—	[Lum et al. 1984]	B^+-Tree or Hashing	primary	transaction	none	1
Time-Split B-Tree	[Lomet & Salzberg 1990]	B^+-Tree	primary	transaction	interval	1
Mixed Media R-Tree	[Kolovson & Stonebraker 1989]	R-Tree	both	transaction, trans+valid	interval, pairs of intervals	k ranges, $k \geq 1$
Time Index	[Elmasri et al. 1990]	B^+-Tree	both	both	interval	0
Two-level Combined Attribute/Time Index	[Elmasri et al. 1991]	B^+-Tree +Time Index	both	both	interval	1
—	[Ahn & Snodgrass 1988]	B^+-Tree, Hashing	*various*	*various*	interval	1
SR-Tree	[Kolovson & Stonebraker 1990]	Segment Index + R-Tree	both	both	interval, pairs of intervals	k ranges, $k \geq 1$

Table 6: Temporal Indexes

chronons) as a linked list of intervals. Hsu has developed an analytical model to determine the optimal block size for such linked lists [Hsu & Snodgrass 1991].

Many structures have been proposed, including *reverse chaining* (all history versions for a key are linked in reverse order) [Ben-Zvi 1982, Dadam et al. 1984, Lum et al. 1984], *accession lists* (a block of time values and associated tuple id's between the current store and the history store), *clustering* (storing history versions together on a set of blocks), *stacking* (storing a fixed number of history versions), and *cellular chaining* (linking blocks of clustered history versions), with analytical performance modeling [Ahn 1986] being used to compare their space and time efficiency [Ahn & Snodgrass 1988].

Several researchers have investigated adapting existing concurrency control and transaction management techniques to support transaction time. The subtle issues involved in choosing whether to time-stamp at the beginning of a transaction (which restricts the concurrency control method that can be used) or at the end of the transaction (which may require data earlier written by the transaction to be read again to record the transaction) have been resolved in favor of the latter through some implementation tricks [Dadam et al. 1984, Lomet 1990, Stonebraker 1987]. The Postgres system is an impressive prototype DBMS that supports transaction time [Stonebraker et al. 1990]. Time-stamping in a distributed setting has also been considered [Lomet 1990]. Integrating temporal indexes with concurrency control to increase the available concurrency has been studied [Lomet & Salzberg 1991].

Finally, since a transaction-time database contains all past versions of the database, it can be used to recover from media failures that cause a portion or all of the current version to be lost [Lomet 1991].

6 Conclusion

We conclude with a list of accomplishments, a list of disappointments, and a pointer to future work.

There have been many significant accomplishments over the past fifteen years of temporal database research.

- The semantics of the time domain, including its structure, dimensionality, and indeterminacy, is well-understood.
- Representational issues of time-stamps have recently been resolved.
- Operations on time-stamps are now well-understood, and efficient implementations exist.
- A great amount of research has been expended on temporal data models, addressing this extraordinarily complex and subtle design problem.
- Many temporal query languages have been proposed. The numerous types of temporal queries are fairly well-understood. Half of the proposed temporal query languages have a strong formal basis.
- Temporal joins are well-understood, and a multitude of implementations exist.
- Approximately a dozen temporal index structures have been proposed.
- The interaction between transaction time support and concurrency control and transaction management has been studied to some depth.
- Several prototype temporal DBMS implementations have been developed.

There have also been some disappointments.

- The user-defined time support in the SQL2 standard is poorly designed. The representation specified in that standard suffers from inadequate range, excessive space requirements, and inefficient operations.
- There has been almost no work done in comparing the two dozen temporal data models, to identify common features and define a consensus data model upon which future research and commercialization may be based. It is our feeling that expecting a data model to simultaneously express time-varying semantics while optimizing data presentation, data storage, and query evaluation is unrealistic. We advocate a two-tiered data model, with a conceptual data model expressing the semantics, and with several representational data models serving these other objectives.

- There is also a need to consolidate approaches to temporal query languages, identify the best features of each of the proposed languages, and incorporate these features into a consensus query language that could serve as the basis for future research into query optimization and evaluation. Also, more work is needed on adding time to so-called fourth generation languages that are revolutionizing user interfaces for commercially available DBMS's.
- It has been demonstrated that a straightforward implementation of a temporal DBMS will exhibit poor performance.
- More empirical studies are needed to compare join algorithms, and to possibly suggest even more efficient variants.
- While there are a host of individual approaches to isolated portions of a DBMS, no coherent architecture has arisen. While the analysis given in Sec. 5 may be viewed as a starting point, much more work is needed to integrate these approaches into a cohesive structure.
- There has been little effort to compare the relative performance of temporal indexes, making selection in specific situations difficult or impossible.
- Temporal database design is still in its infancy, hindered by the plethora of temporal data models.
- There are as yet no prominent commercial temporal DBMS's, despite the obvious need in the marketplace.

Obviously these disappointments should be addressed. In addition, future work is also needed on adding time to the newer data models that are gaining recognition, including object-oriented data models and deductive data models. Finally, there is a great need for integration of spatial and temporal data models, query languages, and implementation techniques.

7 Acknowledgements

This work was supported in part by NSF grant ISI-8902707. James Clifford was helpful in understanding structural aspects of models of time. Curtis Dyreson, Christian S. Jensen, Nick Kline and Michael Soo provided useful comments on a previous draft.

Bibliography

[Ahn 1986] Ahn, I. *Performance Modeling and Access Methods for Temporal Database Management Systems.* PhD. Diss. Computer Science Department, University of North Carolina at Chapel Hill, July 1986.

[Ahn & Snodgrass 1986] Ahn, I. and R. Snodgrass. *Performance Evaluation of a Temporal Database Management System*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. C. Zaniolo. Association for Computing Machinery. Washington, DC: May 1986, pp. 96–107.

[Ahn & Snodgrass 1988] Ahn, I. and R. Snodgrass. *Partitioned Storage for Temporal Databases. Information Systems*, 13, No. 4 (1988), pp. 369–391.

[Ahn & Snodgrass 1989] Ahn, I. and R. Snodgrass. *Performance Analysis of Temporal Queries. Information Sciences*, 49 (1989), pp. 103–146.

[Allen & Hayes 1985] Allen, J. F. and P. J. Hayes. *A Common-Sense Theory of Time*, in *Proceedings of the International Joint Conference on Artificial Intelligence.* Los Angeles, CA: Aug. 1985, pp. 528–531.

[Anderson 1982] Anderson, T. L. *Modeling Time at the Conceptual Level*, in *Proceedings of the International Conference on Databases: Improving Usability and Responsiveness.* Ed. P. Scheuermann. Jerusalem, Israel: Academic Press, June 1982, pp. 273–297.

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

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

[Bernstein 1987] Bernstein, P. A. *Database System Support for Software Engineering- An Extended Abstract*, in *Ninth International Conference on Software Engineering.* IEEE, ACM. Monterey, CA: Computer Society Press, Mar. 1987, pp. 166–178.

[Bhargava & Gadia 1989] Bhargava, G. and S. K. Gadia. *A 2-dimensional Temporal Relational Database Model for Querying Errors and Updates, and for Achieving Zero Information-loss.* Technical Report TR#89-24. Department of Computer Science, Iowa State University. Dec. 1989.

[Blakeley et al. 1986] Blakeley, J.A., P.-A. Larson and F.W. Tompa. *Efficiently Updating Materialized Views*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. C. Zaniolo. Association for Computing Machinery. Washington, DC: May 1986, pp. 61–71.

[Blakeley & Martin 1990] Blakeley, Jose A. and Nancy L. Martin. *Join Index, Materialized View, and Hybrid-Hash Join: A Performance Analysis*, in *Proceedings of the Sixth International Conference on Data Engineering.* February 1990, pp. 256–263.

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

[Chomicki & Imelinski 1988] Chomicki, J. and T. Imelinski. *Temporal Deductive Databases and Infinite Objects*, in *Proceedings of the Seventh ACM SIGAct-SIGMod-SIGArt Symposium on Principles of Database Systems.* Austin, Texas: Association for Computing Machinery, Mar. 1988, pp. 61–73.

[Chomicki & Imelinski 1989] Chomicki, J. and T. Imelinski. *Relational Specifications of Infinite Query Answers*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* May 1989, pp. 174–183.

[Chomicki 1990] Chomicki, J. *Polynomial Time Query Processing in Temporal Deductive Databases*, in *9th Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* Nashville, TN: Apr. 1990.

[Clifford & Warren 1983] Clifford, J. and D. S. Warren. *Formal Semantics for Time in Databases. ACM Transactions on Database Systems*, 8, No. 2, June 1983, pp. 214–254.

[Clifford & Tansel 1985] Clifford, J. and A. U. Tansel. *On an Algebra for Historical Relational Databases: Two Views*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247–265.

[Clifford & Croker 1987] Clifford, J. and A. Croker. *The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans*, in *Proceedings of the International Conference on Data Engineering.* IEEE Computer

Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1987, pp. 528–537.

[Clifford & Rao 1987] Clifford, J. and A. Rao. *A Simple, General Structure for Temporal Domains*, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* AFCET. France: May 1987, pp. 23–30.

[Codd 1972A] Codd, E. F. *Further Normalization of the Data Base Relational Model*, in Data Base Systems. Vol. 6 of Courant Computer Symposia Series. Englewood Cliffs, N.J.: Prentice Hall, 1972.

[Codd 1972B] Codd, E. F. *Relational Completeness of Data Base Sublanguages*, in Data Base Systems. Vol. 6 of Courant Computer Symposia Series. Englewood Cliffs, N.J.: Prentice Hall, 1972. pp. 65–98.

[Comer 1979] Comer, D. *The Ubiquitous B-tree. Computing Surveys*, 11, No. 2 (1979), pp. 121–138.

[Dadam et al. 1984] Dadam, P., V. Lum and H.-D. Werner. *Integration of Time Versions into a Relational Database System*, in *Proceedings of the Conference on Very Large Databases.* Ed. U. Dayal, G. Schlageter and L.H. Seng. Singapore: Aug. 1984, pp. 509–522.

[Date 1989A] Date, C. J. *An Overview of SQL2. Info. Database*, 4, No. 1, Spring 1989, pp. 8–12.

[Date 1989B] Date, C. J. *A Guide to the SQL Standard (Second Edition).* Addison-Wesley, 1989.

[Date & White 1990] Date, C. J. and C. J. White. *A Guide to DB2.* Reading, MA: Addison-Wesley, 1990. Vol. 1, 3rd edition.

[Date 1988] Date, C.J. *A Proposal for Adding Date and Time Support to SQL. SIGMOD Record*, 17, No. 2, June 1988, pp. 53–76.

[Dayal & Smith 1986] Dayal, U. and J. M. Smith. *PROBE: A Knowledge-Oriented Database Management System*, in On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies. Springer-Verlag, 1986.

[Dayal & Wuu 1992] Dayal, U. and G. Wuu. *A Uniform Approach to Processing Temporal Queries.* Technical Report. Bellcore. 1992.

[DeAntonellis et al. 1979] DeAntonellis, V., A. Degli, G. Mauri and B. Zonta. *Extending the Entity-Relationship Approach to Take Into Account Historical Aspects of Systems*, in *Proceedings of the International Conference on the E-R Approach to Systems Analysis and Design.* Ed. P. Chen. North Holland, 1979.

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

[Dittrich & Lorie 1988] Dittrich, Klaus R. and Raymond A. Lorie. *Version Support for Engineering Database Systems. IEEE Transactions on Software Engineering*, 14, No. 4, April 1988, pp. 429–437.

[Dyreson & Snodgrass 1992A] Dyreson, C. E. and R. T. Snodgrass. *Time-stamp Semantics and Representation.* Technical Report TR 92-16a. Computer Science Department, University of Arizona. July 1992.

[Dyreson & Snodgrass 1992B] Dyreson, C. E. and R. T. Snodgrass. *Time-stamp Semantics and Representation.* TempIS Technical Report 33. Computer Science Department, University of Arizona. Revised May 1992.

[Ecklund et al. 1987] Ecklund, D. J., E. F. Ecklund, R. O. Eifrig and F. M. Tonge. *DVSS: A Distributed Version Storage Server for CAD Applications*, in *Proceedings of the Conference on Very Large Databases.* Brighton, England: 1987, pp. 443–454.

[Elmasri et al. 1990] Elmasri, R., G. Wuu and Y. Kim. *The Time Index - An Access Structure for Temporal Data*, in *Proceedings of the Conference on Very Large Databases.* Brisbane, Australia: Aug. 1990.

[Elmasri et al. 1991] Elmasri, R., 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, pp. 102–111.

[Elmasri et al. 1992] Elmasri, R., M. Jaseemuddin and V. Kouramajian. *Partitioning of Time Index for Optical Disks*, in *Proceedings of the International Conference on Data Engineering.* Ed. F. Golshani. IEEE. Phoenix, AZ: Feb. 1992, pp. 574–583.

[Enderton 1977] Enderton, H.B. *Elements of Set Theory.* New York, N.Y.: Academic Press, Inc., 1977.

[Gadia 1986] Gadia, S. K. *Toward a Multihomogeneous Model for a Temporal Database*, in *Proceedings of the International Conference on Data Engineering.* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 390–397.

[Gadia 1988] Gadia, S. K. *A Homogeneous Relational Model and Query Languages for Temporal Databases. ACM Transactions on Database Systems*, 13, No. 4, Dec. 1988, pp. 418–448.

[Gadia & Yeung 1988] Gadia, S. K. and C. S. Yeung. *A Generalized Model for a Relational Temporal Database*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery. Chicago, IL: June 1988, pp. 251–259.

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

[Gadia et al. 1992] Gadia, S. K., S. Nair and Y.-C. Poon. *Incomplete Information in Relational Temporal Databases*, in *Proceedings of the Conference on Very Large Databases.* Vancouver, Canada: Aug. 1992.

[Guinot & Seidelmann 1988] Guinot, B. and P. K. Seidelmann. *Time scales: their history, definition and interpretation. Astronomy & Astrophysics*, 194 (1988), pp. 304–308.

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

[Gunadhi et al. 1989] Gunadhi, H., A. Segev and G. Shantikumar. *Selectivity Estimation in Temporal Databases.* Technical Report LBL-27435. Lawrence Berkeley Laboratories. 1989.

[Gunadhi & Segev 1991] Gunadhi, H. and A. Segev. *Query Processing Algorithms for Temporal Intersection Joins*, in *Proceedings of the 7th International Conference on Data Engineering.* Kobe, Japan: 1991.

[Gunadhi & Segev 1993] Gunadhi, H. and A. Segev. *Efficient Indexing Methods for Temporal Relations. IEEE Transactions on Knowledge and Data Engineering*, 5, No. 3, June 1993, pp. 496–509.

[Guttman 1984] Guttman, A. *R-Trees: A Dynamic Index Structure For Spatial Searching*, in *Proceedings of*

ACM SIGMOD International Conference on Management of Data. Ed. B. Yormack. Association for Computing Machinery. Boston, MA: June 1984, pp. 47–57.

[Hall et al. 1976] Hall, P., J. Owlett and S. J. P. Todd. *Relations and Entities*, in *Modelling in Data Base Management Systems.* Ed. G. M. Nijssen. North-Holland, 1976. pp. 201–220.

[Hammer & McLeod 1981] Hammer, M. and D. McLeod. *Database Description with SDM: A Semantic Database Model. ACM Transactions on Database Systems*, 6, No. 3, Sep. 1981, pp. 351–386.

[Hawking 1988] Hawking, S. *A Brief History of Time.* New York: Bantam Books, 1988.

[Held et al. 1975] Held, G.D., M. Stonebraker and E. Wong. *INGRES–A Relational Data Base Management System*, in *Proceedings of the AFIPS National Computer Conference.* Anaheim, CA: AFIPS Press, May 1975, pp. 409–416.

[Hsieh 1989] Hsieh, D. *Generic Computer Aided Software Engineering (CASE) Databases Requirements*, in *Proceedings of the Fifth International Conference on Data Engineering.* Los Angeles, CA: Feb. 1989, pp. 422–423.

[Hsu & Snodgrass 1991] Hsu, S. H. and R. T. Snodgrass. *Optimal Block Size for Repeating Attributes.* TempIS Technical Report No. 28. Department of Computer Science, University of Arizona. Dec. 1991.

[Hunter & Williamson 1990] Hunter, G. and I. Williamson. *The Development of a Historical Digital Cadastral Database. International Journal of Geographical Information Systems*, 4, No. 2 (1990), pp. 169–179.

[Jensen & Mark 1991] Jensen, C. S. and L. Mark. *Queries on Change in an Extended Relational Model. IEEE Transactions on Knowledge and Data Engineering*, (to appear) (1991).

[Jensen et al. 1991] Jensen, C. S., L. Mark and N. Roussopoulos. *Incremental Implementation Model for Relational Databases with Transaction Time. IEEE Transactions on Knowledge and Data Engineering*, 3, No. 4, Dec. 1991, pp. 461–473.

[Jensen et al. 1992] Jensen, C. S., M. D. Soo and R. T. Snodgrass. *Unification of Temporal Relations.* Technical Report 92-15. Computer Science Department, University of Arizona. July 1992.

[Jensen & Snodgrass 1993] Jensen, C. S. and R. Snodgrass. *Temporal Specialization and Generalization. IEEE Transactions on Knowledge and Data Engineering*, (to appear) (1993).

[Jensen et al. 1993] Jensen, C. S., L. Mark, N. Roussopoulos and T. Sellis. *Using Differential Techniques to Efficiently Support Transaction Time. The VLDB Journal*, 2, No. 1, Jan. 1993, pp. 75–111.

[Jones 1989] Jones, C. B. *Data structures for three-dimensional spatial information systems in geology. International Journal of Geographical Information Systems*, 3, No. 1 (1989), pp. 15–31.

[Jones et al. 1979] Jones, S., P. Mason and R. Stamper. *LEGOL 2.0: A Relational Specification Language for Complex Rules. Information Systems*, 4, No. 4, Nov. 1979, pp. 293–305.

[Kabanza et al. 1990] Kabanza, F., J-M Stevenne and P. Wolper. *Handling Infinite Temporal Data*, in *9th Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* Nashville, TN: Apr.

1990.

[Kahn & Gorry 1977] Kahn, K. and G. A. Gorry. *Mechanizing Temporal Knowledge. Artificial Intelligence*, (1977), pp. 87–108.

[Karlsson 1986] Karlsson, T. *Representation and Reasoning about Temporal Knowledge.* SYSLAB Working Paper Nr 105. The Systems Development and Artificial Intelligence Laboratory, University of Stockholm. 1986.

[Katz et al. 1986] Katz, R. H., E. Chang and R. Bhateja. *Version Modeling Concepts for Computer-Aided Design Databases*, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* Ed. C. Zaniolo. Association for Computing Machinery. Washington, DC: May 1986, pp. 379–386.

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

[Klopprogge 1981] Klopprogge, M. R. *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.* Washington, DC: Oct. 1981, pp. 477–512.

[Kolovson & Stonebraker 1989] Kolovson, C. and M. Stonebraker. *Indexing Techniques for Historical Databases*, in *Proceedings of the Fifth International Conference on Data Engineering.* Los Angeles, CA: Feb. 1989, pp. 127–137.

[Kolovson 1990] Kolovson, C. *Indexing Techniques for Multi-Dimensional Spatial Data and Historical Data in Database Management Systems.* PhD. Diss. University of California, Berkeley, Nov. 1990.

[Kolovson & Stonebraker 1990] Kolovson, C. and M. Stonebraker. *S-Trees: Database Indexing Techniques for Multi-Dimensional Interval Data.* Technical Report UCB/ERL M90/35. University of California. Apr. 1990.

[Ladkin 1987] Ladkin, P. *The Logic of Time Representation.* PhD. Diss. University of California, Berkeley, Nov. 1987.

[Langran & Chrisman 1988] Langran, G. and N. Chrisman. *A Framework for Temporal Geographic Information. Cartographica*, 25, No. 3 (1988), pp. 1–14.

[Lee et al. 1985] Lee, R. M., H. Coelho and J. C. Cotta. *Temporal Inferencing on Administrative Databases. Information Systems*, 10, No. 2 (1985), pp. 197–206.

[Leung & Muntz 1990] Leung, T. Y. and R. Muntz. *Query Processing for Temporal Databases*, in *Proceedings of the 6th International Conference on Data Engineering.* Los Angeles, California: Feb. 1990.

[Leung & Muntz 1991A] Leung, T. Y. and R. Muntz. *Stream Processing: Temporal Query Processing and Optimization.* Technical Report CSD-910079. University of California, Los Angeles. Dec. 1991.

[Leung & Muntz 1991B] Leung, T. Y. and R. Muntz. *Temporal Query Processing and Optimization in Multiprocessor Database Machines.* Technical Report CSD-910077. Computer Science Department, UCLA. Nov. 1991.

[Leung & Muntz 1992] Leung, T. Y. and R. Muntz. *Generalized Data Stream Indexing and Temporal Query*

Processing, in *Second International Workshop on Research Issues in Data Engineering: Transaction and Query Processing*. Feb. 1992.

[Lomet & Salzberg 1989] Lomet, D. and B. Salzberg. *Access Methods for Multiversion Data*, in *Proceedings of ACM SIGMOD International Conference on Management of Data*. June 1989, pp. 315–324.

[Lomet 1990] Lomet, D. *Consistent Timestamping for Transactions in Distributed Systems*. Technical Report CRL90/3. Digital Equipment Corporation. Sep. 1990.

[Lomet & Salzberg 1990] Lomet, D. and B. Salzberg. *The Performance of a Multiversion Access Method*, in *Proceedings of ACM SIGMOD International Conference on Management of Data*. Atlantic City: May 1990, pp. 353–363.

[Lomet 1991] Lomet, D. *Grow and Post Index Trees: Role, Techniques and Future Potential*, in *Proc. of the Second Symposium on Large Spatial Databases*. 1991.

[Lomet & Salzberg 1991] Lomet, D. and B. Salzberg. *Concurrency and Recovery for Index Trees*. Technical Report CRL 91/8. Digital Equipment Corporation. Aug. 1991.

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

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

[Lum et al. 1984] Lum, V., P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner and J. Woodfill. *Designing DBMS Support for the Temporal Dimension*, in *Proceedings of ACM SIGMOD International Conference on Management of Data*. Ed. B Yormark. Association for Computing Machinery. Boston, MA: June 1984, pp. 115–130.

[Manola & Dayal 1986] Manola, F. and U. Dayal. *PDM: An Object-Oriented Data Model*, in *Proceedings of the International Workshop on Object-Oriented Database Systems*. 1986.

[Mark et al. 1989] Mark, D. M., J. P. Lauzon and J. A. Cebrian. *A review of quadtree-based strategies for interfacing coverage data with digital elevation models in grid form. International Journal of Geographical Information Systems*, 3, No. 1 (1989), pp. 3–14.

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

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

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

[McKenzie & Snodgrass 1991A] McKenzie, E. and R. Snodgrass. *An Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. ACM Computing Surveys*, 23, No. 4, Dec. 1991, pp. 501–543.

[McKenzie & Snodgrass 1991B] McKenzie, L. and R. T. Snodgrass. *Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions.* Technical Report TR–91–15. Department of Computer Science, University of Arizona. Aug. 1991.

[Melton 1990] Melton, J. (ed.) *Solicitation of Comments: Database Language SQL2.* American National Standards Institute, Washington, DC, 1990.

[Montague 1973] Montague, R. *The proper treatment of quantification in ordinary English*, in Approaches to Natural Language. Dordrecht, Holland: D. Reidel Publishing Co., 1973.

[Narasimhalu 1988] Narasimhalu, A. *A Data Model for Object-Oriented Databases with Temporal Attributes and Relationships.* Technical Report. National University of Singapore. 1988.

[Navathe & Ahmed 1987] Navathe, S. B. and R. Ahmed. *TSQL-A Language Interface for History Databases*, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* AFCET. France: May 1987, pp. 113–128.

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

[USNO 1992] Observatory, U.S. Naval *Time Service Announcement.* Series 14. Washington, D.C.. Feb. 1992.

[Oracle 1987] Oracle Computer, Inc. *ORACLE Terminal User's Guide.* Oracle Corporation, 1987.

[Petley 1991] Petley, B. W. *Time and Frequency in Fundamental Metrology. Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 1070–1077.

[Quinn 1991] Quinn, T. J. *The BIPM and the Accurate Measurement of Time. Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 894–906.

[Ramsey 1991] Ramsey, N. F. *The Past, Present, and Future of Atomic Time and Frequency. Proceedings of the IEEE*, 79, No. 9, July 1991, pp. 936–943.

[Rose & Segev 1991] Rose, E. and A. Segev. *TOODM - A Temporal Object-Oriented Data Model with Temporal Constraints*, in *Proceedings of the 10th International Conference on the Entity Relationship Approach.* Oct. 1991.

[Roussopoulos 1982] Roussopoulos, N. *View Indexing in Relational Databases. ACM Transactions on Database Systems*, 7, No. 2, June 1982, pp. 258–290.

[Roussopoulos 1991] Roussopoulos, N. *An Incremental Access Method for ViewCache: Concept, Algorithms, and Cost Analysis. ACM Transactions on Database Systems*, 16, No. 3, september 1991, pp. 535-563.

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

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

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

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

[Satoh et al. 1985] Satoh, K., M. Tsuchida, F. Nakamura and K. Oomachi. *Local and Global Query Optimization Mechanisms for Relational Databases*, in *Proceedings of the Conference on Very Large Databases*. Ed. A. Pirotte and Y. Vassiliou. Stockholm, Sweden: Aug. 1985, pp. 405–417.

[CES 1989] Sciences, Committee on Earth *Our Changing Planet: A U.S. strategy for global change research*. 1989. (Unpublished paper.)

[Sciore 1991] Sciore, E. *Versioning and Configuration management in an Object-Oriented Data Model*. Technical Report BCCS 91-12 revised. Computer Science Department, Boston College. 1991.

[Sciore 1991] Sciore, E. *Multidimensional Versioning for Object-Oriented databases. Proc. Second International Conf. on Deductive and Object-Oriented Databases*, , Dec. 1991.

[Segev & Shoshani 1987] Segev, A. and A. Shoshani. *Logical Modeling of Temporal Data*, in *Proceedings of the ACM SIGMOD Annual Conference on Management of Data*. Ed. U. Dayal and I. Traiger. Association for Computing Machinery. San Francisco, CA: ACM Press, May 1987, pp. 454–466.

[Sellis 1986] Sellis, T.K. *Global Query Optimization*, in *Proceedings of ACM SIGMOD International Conference on Management of Data*. Ed. C. Zaniolo. Association for Computing Machinery. Washington, DC: May 1986, pp. 191–205.

[Sheng 1984] Sheng, R. L. *A Linguistic Approach to Temporal Information Analysis*. PhD. Diss. University of California, Berkeley, May 1984.

[Shenoy & Özsoyoğlu 1989] Shenoy, S. and Z. Özsoyoğlu. *Design and Implementation of a Semantic Query Optimizer. IEEE Transactions on Data and Knowledge Engineering*, 1, No. 3, Sep. 1989, pp. 344–361.

[Smith & Chang 1975] Smith, J.M. and P.Y-T. Chang. *Optimizing the Performance of a Relational Algebra Database Interface. Communications of the Association of Computing Machinery*, 18, No. 10, Oct. 1975, pp. 568–579.

[Snodgrass et al. 1993] Snodgrass, R., S. Gomez and E. McKenzie. *Aggregates in the Temporal Query Language TQuel. IEEE Transactions on Knowledge and Data Engineering*, (to appear) (1993).

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

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

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

[Soo & Snodgrass 1992A] Soo, M. D. and R. Snodgrass. *Multiple Calendar Support for Conventional Database Management Systems.* Technical Report 92-7. Computer Science Department, University of Arizona. Feb. 1992.

[Soo & Snodgrass 1992B] Soo, M. D. and R. Snodgrass. *Mixed Calendar Query Language Support for Temporal Constants.* TempIS Technical Report 29. Computer Science Department, University of Arizona. Revised May 1992.

[Soo et al. 1992] Soo, M. D., R. Snodgrass, C. Dyreson, C. S. Jensen and N. Kline. *Architectural Extensions to Support Multiple Calendars.* TempIS Technical Report 32. Computer Science Department, University of Arizona. Revised May 1992.

[Sripada 1988] Sripada, S. *A Logical Framework for Temporal Deductive Databases*, in *Proceedings of the Conference on Very Large Databases.* Los Angeles, CA: 1988, pp. 171–182.

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

[Stonebraker et al. 1976] Stonebraker, M., E. Wong, P. Kreps and G. Held. *The Design and Implementation of Ingres. ACM Transactions on Database Systems*, 1, No. 3, Sep. 1976, pp. 189–222.

[Stonebraker 1987] Stonebraker, M. *The Design of the POSTGRES Storage System*, in *Proceedings of the Conference on Very Large Databases.* Ed. P. Hammersley. Brighton, England: Sep. 1987, pp. 289–300.

[Stonebraker et al. 1990] Stonebraker, M., L. Rowe and M. Hirohama. *The Implementation of POSTGRES. IEEE Transactions on Knowledge and Data Engineering*, 2, No. 1, Mar. 1990, pp. 125–142.

[Tandem 1983] Tandem Computers, Inc. *ENFORM Reference Manual.* Cupertino, CA, 1983.

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

[Tansel & Arkun 1986] Tansel, A. U. and M. E. Arkun. *HQuel, A Query Language for Historical Relational Databases*, in *Proceedings of the Third International Workshop on Statistical and Scientific Databases.* July 1986.

[Tansel et al. 1989] Tansel, A.U., M.E. Arkun and G. Özsoyoğlu. *Time-By-Example Query Language for Historical Databases. IEEE Transactions on Software Engineering*, 15, No. 4, Apr. 1989, pp. 464–478.

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

[Tsichritzis & Lochovsky 1982] Tsichritzis, D.C. and F.H. Lochovsky. *Data Models.* Software Series. Prentice-Hall, 1982.

[Tuzhilin and Clifford 1990] Tuzhilin, A. and J. Clifford. *A Temporal Relational Algebra as a Basis for Temporal Relational Completeness*, in *Proceedings of the Conference on Very Large Databases.* Brisbane, Australia: Aug. 1990.

[Ullman 1988] Ullman, Jeffrey David *Database and Knowledge - Base Systems — II: The New Technologies.* Vol. II of Principles of Computer Science. 1803 Research Boulevard, Rockville, MD 20850: Computer Science Press, 1988.

[Urban & Delcambre 1986] Urban, S. D. and L. M. L. Delcambre. *An Analysis of the Structural, Dynamic, and Temporal Aspects of Semantic Data Models*, in *Proceedings of the International Conference on Data Engineering.* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 382–389.

[Van Benthem 1982] Van Benthem, J. F. K. A. *The Logic of Time.* Reidel, 1982.

[Vrana 1989] Vrana, R. *Historical Data as an Explicit Component of Land Information Systems. International Journal of Geographical Information Systems*, 3, No. 1 (1989), pp. 33–49.

[Wiederhold et al. 1975] Wiederhold, G., J.F. Fries and S. Weyl. *Structured Organization of Clinical Data Bases*, in *Proceedings of the AFIPS National Computer Conference.* AFIPS. 1975, pp. 479–485.

[Wiederhold et al. 1991] Wiederhold, G., S. Jajodia and W. Litwin. *Dealing with Granularity of Time in Temporal Databases*, in *Proc. 3rd Nordic Conf. on Advanced Information Systems Engineering.* Trondheim, Norway: May 1991.

[Worboys 1990] Worboys, M. F. *Reasoning About GIS Using Temporal and Dynamic Logics*, in *Temporal GIS Workshop.* University of Maine. Oct. 1990.

[Wuu & Dayal 1992] Wuu, G. and U. Dayal. *A Uniform Model for Temporal Object-Oriented Databases*, in *Proceedings of the International Conference on Data Engineering.* Tempe, Arizona: Feb. 1992, pp. 584–593.

[Zloof 1975] Zloof, M. *Query By Example*, in *Proceedings of the National Computer Conference.* AFIPS. 1975.