Temporal and Real-Time Databases: A Survey

Gultekin Özsoyoğlu* **Richard T. Snodgrass†**

Abstract

A *temporal database* contains time-varying data. In a *real-time database* transactions have deadlines or timing constraints. In this paper we review the substantial research in these two heretofore separate research areas. We first characterize the time domain, then investigate temporal and real-time data models. We evaluate temporal and real-time query languages along several dimensions. Temporal and real-time DBMS implementation is examined. We conclude with a summary of the major accomplishments of the research to date, and list several research questions that should be addressed next.

Keywords: object-oriented database, relational databases, query language, temporal data model, time-constrained database, transaction time, user-defined time, valid time

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 world and to respond within time constraints to changes in the real world as well as to application-dependent operations is essential to many computer applications, such as accounting, banking, econometrics, geographical information systems, inventory control, law, medical records, multi-media, process control, reservation systems, and scientific data analysis.

Conventional databases represent the state of an enterprise at a single moment of time. Although the contents of the database (DB) continue to change as new information is added, these changes are viewed as modifications to the state, with the old, out-of-date data being deleted from the database.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 snapshot of the enterprise. Additionally, conventional DBMSs execute queries and transactions in the order that they arrive, and provide no guarantees concerning query or transaction completion times.

*Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH 44106. `tekin@alpha.ces.cwru.edu`

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

In this paper, we survey two database research areas that may benefit from cross infusion, namely, temporal database research for providing application-independent DBMS support for time-varying information, and real-time database research for completing database operations within time constraints. Our view is that the real-time database research may benefit directly by utilizing temporal data models in real-time transaction and query specification and management, thereby providing better temporal data semantics and better querying capabilities The temporal database research may benefit indirectly by the use and extension of temporal data models into another area, namely, real-time databases. This paper attempts to capture and summarize the major concepts, approaches, and implementation strategies that have been generated by these two research areas. A common terminology is used to emphasize concepts independently generated. Wherever possible, research directions that relate the two areas are listed.

A temporal database is one that supports some aspect of time [95]. We will shortly encounter more specific characterizations that concern the kind of time(s) supported. Most applications manage historical information, yet conventional DBMSs provide little support.

In most of the literature, a real-time database is defined as a database where transactions have deadlines or timing constraints. Real-time databases are commonly used within real-time computing applications that require timely access to data. And, usually, the definition of timeliness is not quantified; for some applications, it is milliseconds, and, for others, it is minutes [203]. Therefore, real-time databases can perhaps be better viewed as *time-constrained* databases. In this survey, we define a real-time database as a database that has timing constraints in every operational aspect of the database such as responding to queries, processing transactions, processing database insertions, deletions, and updates, and maintaining the integrity of the database via integrity constraint enforcement and view management. Although current real-time database research literature uses temporal data and "temporal consistency constraints," temporal data models and temporal query languages are not utilized. Since real-time databases naturally deal with time, researchers should use models and theories developed in temporal databases.

A series of five bibliographies concerning temporal databases [29, 142, 201, 197, 114] references some 600 papers through October, 1993. A bibliography on space and time in databases [12] lists 144 temporal database papers. An annotated bibliography on schema evolution [161] includes eight temporal database papers.

The book edited by Tansel provides a still-current snapshot of temporal databases research [207]. Several chapters of that book supplement this survey, in particular the excellent surveys on temporal reasoning [148] and on temporal deductive databases [18].

Other extant surveys include those on temporal data models [97, 187], temporal query languages [38, 144, 189], and temporal access methods [168].

In the last eight years, about 150 papers on real-time databases have been published. There exist several surveys on this topic [126, 159, 213, 231].

Our emphasis is not on peripheral database research areas that may be useful to temporal and real-time databases, such as main-memory databases, active databases, or multi-dimensional (spatial) databases. Instead, we hope to provide the reader with a general understanding of the major issues that have been addressed in the areas of temporal and real-time databases; space limitations preclude delving into any topic in detail.

In the sections that follow, we review the substantial research on temporal and real-time databases. We first examine the time domain: its structure, dimensionality (interestingly, there are several time dimensions), temporal indeterminacy, and the types of time associated with real-time data. Section 3 examines the many temporal and real-time data models that have been proposed in the literature.

We next consider languages for expressing temporal queries. We briefly discuss and compare some three dozen temporal relational and object-oriented query languages.

Real-time data and query languages are discussed in Section 5, which presents the characteristics of real-time data and transactions, and new types of data and transaction consistency requirements that are unique to real-time databases.

Real-time and temporal DBMS implementation is the topic of Section 6. We examine the impact to each of the components of a DBMS of adding real-time and temporal support, with query processing and transaction processing discussed in some detail.

We conclude with a summary of the major accomplishments of research into real-time and temporal databases, and point to future work.

2 The Time Domain

In this section we focus on time itself. 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* [160, 218]. 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 [225]. 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 a partially ordered set [62]. Additional axioms can be introduced to specify 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 [39].

Axioms may also be added to temporal logics to characterize the *density* of the time line [218]. 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 [44]. *Dense* models of time are isomorphic to 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 referred to as a *chronon* [95]. 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.

Axioms can also describe the *boundedness* of time. Time can be bounded orthogonally in the past and in the future. Models of time may include the concept of *distance* (most temporal logics do not do so, however; exceptions include [160]). 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, 12 ± 6 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 [80]). 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, with the finite energy of the universe eventually dissipating.

Finally, one can differentiate *relative* time from *absolute* time (more precise terms are *unanchored* 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 'absolute' only with respect to another time (in this example, midnight, January 1, A.D. 1). 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.

2.2 Dimensionality

In the context of databases, two time dimensions are of general interest [185]. *Valid time* concerns the time a fact was true in reality. The valid time of an event is the 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 expected 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 a fact identifies the transaction that inserted the fact into the database and the transaction that removed this fact from the database.

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 *valid-time*; one that supports only transaction time is termed *transaction-time*; and one that supports both valid and transaction time is termed *bitemporal* (*temporal* is a generic term implying some kind of time support) [95]. Both linear and branching transaction time have been employed in temporal databases.

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 the creation time, nothing was stored), and doesn't extend past the present (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, transaction-time 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 differing valid times.

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* [98], valid and transaction time are identical.

Multiple transaction times may also be stored in the same relation, termed *temporal generalization* [98]. For example, a fact may be stored in one relation, with one associated transaction time, then later copied to a summary relation, with a different associated transaction time. By retaining the original transaction time, the summary relation can support queries that select information based on when that information resided in the original relation.

These valid and transaction times may also be related to each other, or to the valid time, in various specialized ways.

There is a third kind of time that may be included: *user-defined time*. This term refers to the fact that the semantics of these values are known only to the user, and are not interpreted by the DBMS, as differentiated from valid and transaction time, whose semantics are supported by the DBMS.

Recently a separate kind of time, termed *event time* [36] or *decision time*, has been defined (we'll use the latter term, to avoid confusion with the time of an event in an active or real-time database). The decision time of a fact, such as the promotion of a professor to a new rank, is the time that decision occurred, i.e., the promotion was determined. Note that this time may be different than the valid time of the new position, and may also be different than the transaction time of when the new position was recorded. Note that decision time is an instant; generally valid and transaction time are intervals. Note also that decision time may be recorded as a valid time for a separate table (e.g., as a promotion table, as distinct from a rank table). It is still an open issue as to whether the increase in complexity of including decision time in the data model is justified by the increased expressive power [112].

2.3 Indeterminacy

Information that is *temporally indeterminate* can be characterized as "don't know exactly when" information. This kind of information is prevalent; it arises in various situations, including finer system granularity, imperfect dating techniques, uncertainty in planning, and unknown or imprecise event times. There have been several proposals for adding temporal indeterminacy to the time model [55, 65, 117], as well as more specific work on accommodating multiple time granularities [122, 223, 221]. The *possible chronons* model [56] unifies treatment of both aspects [57]. In this model, an event is *determinate* if it is known 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 temporally indeterminate. The indeterminacy refers to the *time* when the event occurred, not *whether* the event did or did not occur. Temporal 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.

2.4 Time in Real-Time Databases

Current real-time database research does not explicitly distinguish between the various time dimensions and the related time issues that are surveyed in this paper. However, close inspection reveals that real-time databases use valid time and transaction time [30, 69, 126, 159, 190, 191, 213]. *Valid time* is used for data items that have immediate counterparts (external objects) in the real (physical) world. External events corresponding to value changes for these external objects are closely monitored, and write-only transactions record them to the database. For example, programmable logic controllers use specialized sensors and channels to detect events with their actual occurrence times. An example event may be "to detect furnace temperature changes when they are above a threshold x".

Real-time databases use *transaction time* for transactions that set parameters of a real-time system with the help of a real-time database and specialized output devices. That is, the event of setting the value of a real-world object is performed by a transaction. An example may be a transaction "to increase the coolant level of a furnace to level y when a computed database value goes above the threshold x". However, the occurrence time of such an event can be any time between the transaction begin time and the transaction commit time. (Usually, these transactions are never aborted or rolled back). Also, transaction time is used when new values are derived for data items on the basis of the values of other data items. Times associated with these derived values generally correspond to transaction commit times.

Real-time database research usually does not refer to the 'future', and assume a linear, bounded (in the past and until 'now') time model. However, we expect the use of temporal models with linear- or branching-time future for real-time databases.

Presently, temporal indeterminacy is not dealt with in real-time database research. However, temporally indeterminate information is encountered frequently in real-time systems as unknown or imprecise event times. For example, the temperature of a furnace changes, but, we are not sure exactly when; this change and its time needs to be stored into the database and made available to real-time operations such as queries and transactions. Clearly, real-time transaction processing needs to be extended with data models that allow temporal indeterminacy.

Transactions themselves have time constraints (deadlines) that are specified in varying granularities as distances with respect to a specified time. Such constraints relate valid and transaction time, in that the transaction commit time must be before the specified valid time.

It is also important to note that, for some real-time systems where missing a deadline has serious implications and should not happen, termed *hard real-time systems*, the values of data items in the database, however recent, may still not be correct since the reality being modeled may change immediately after the value was recorded [231]. The possibility of incorrect data, and thus incorrect computation, has serious implications in hard real-time systems.

Clearly, not all real-time applications need timely behaviour for all operations in a database. However, one can find real-time applications [100] that need timeliness in some, if not all, of the operations in a real-time database. For example, stock market, air traffic control, or on-board airplane databases all require timeliness in database updates and transaction completions, while factory-floor databases commonly require timely query and transaction completions and, not necessarily, timely database updates.

3 Data Models

The previous section explored models and representations for the time domain. We now turn to associating time with facts. Research in temporal data models and real-time data models have proceeded independently, with little cross fertilization.

3.1 Temporal Data Models

Time has been added to many data models, including the entity-relationship model [11, 51, 58, 115, 227], semantic data models [73, 217], knowledge-based data models [49], and deductive databases [18, 28]. However, by far the majority of work in temporal databases is based on the relational and object-oriented (OO) data models. For this reason, we focus on these two data models in subsequent discussion.

Support for time in conventional database systems is entirely at the level of user-defined time (i.e., attribute values drawn from a temporal domain). In SQL-92 `Date` has a granularity of a day; a `Time` has a granularity of a second, but a range of only 100 hours; a `Timestamp` combines the range of a `Date` with the granularity of a second (there are `Timestamp` variants with a granularity of fractions of a second) [146].

These implementations are limited in scope and are, in general, unsystematic in their design [48]. SQL-92 corrects some of the inconsistencies in the time support provided by DB2, but inherits its basic design limitations.

None of the other object-oriented database standards, including OMG's IDL data model supporting CORBA [152] and ODMG-93 [35], support valid or transaction time. In fact, CORBA doesn't even support user-defined time. ODMG's support of user-defined time is identical to that of SQL-92.

An effort has just been completed to consolidate approaches to temporal data models and calculus-based query languages, to achieve a consensus extension to SQL-92 and an associated data model upon which future research can be based. This extension is termed the *Temporal Structured Query Language*, or TSQL2 [188]. An analogous consensus object-oriented extension to SQL3 would be highly desirable, but thus far little progress has been made towards this goal.

Table 1 lists most of the temporal relational data models that have appeared in the literature[1]. We include all models that we are aware of that have appeared in the literature. We omit Gadia's multihomogeneous model [63], which is a precursor to his heterogeneous model (Gadia-2). We also do not include the data model used as the basis for defining temporal relational completeness [43], because it is a generic data model that does not force decisions on most of the aspects to be discussed here.

Some models are defined only over valid time or transaction time; others are defined over both. The last column indicates a short identifier which denotes the model; the table is sorted on this column[2].

We classify the extant object-oriented temporal data models in Table 2. The fourth column will be discussed further in Section 3.1.2. Those with "arbitrary" indicated in the third and fourth columns support time with user- or system-provided classes; hence, anything is possible. N/A denotes "not applicable".

Only a few relational or object-oriented data models explicitly support user-defined time; where absent, such support is not difficult to add.

3.1.1 Valid Time

These models may be compared along the valid-time dimension by asking three basic questions: how is valid time represented, how are facts associated with valid time, and how are attribute values represented. Here we list some of the many answers to these questions.

Valid times can be represented with single chronon identifiers (i.e., event time-stamps), with intervals (i.e., as interval time-stamps), or as *valid-time elements*, which are finite sets of intervals [64]. Valid time can be associated with individual attribute values, groups of attributes, or with an entire tuple or object. A fifth alternative, associating valid time with sets of tuples (i.e., relations) or object graphs, has not been incorporated into any of the proposed data models, primarily because it lends itself to high data redundancy. Table 3 categorizes most the the data models along these two dimensions. We do not include the OODAPLEX, Sciore-1, and TIGUKAT data models as these two aspects are arbitrarily specifiable in these models.

[1] Many models are described in several papers; the one referenced is the initial journal paper in which the model was defined.

[2] If the model has not been given a name, we use the name of the first designer of the model as an identifier; this is also done for models with identical acronyms.

Data Model Name	*Citation*	*Temporal Dimension(s)*	*Identifier*
Accounting Data Model	[211]	both	ADM
—	[185]	both	Ahn
Temporally Oriented Data Model	[14]	both	Ariav
—	[17]	valid	Bassinouni
—	[25]	both	Bhargava
Bitemporal Conceptual Data Model	[99]	both	BCDM
Time Relational Model	[21]	both	Ben-Zvi
DATA	[113]	transaction	DATA
DM/T	[92]	transaction	DM/T
Homogeneous Relational Model	[64]	valid	Gadia-1
Heterogeneous Relational Model	[66]	valid	Gadia-2
Historical Data Model	[45]	valid	HDM
Historical Relational Data Model	[42]	valid	HRDM
—	[101]	valid	Jones
—	[137]	transaction	Lomet
Temporal Relational Model	[139]	valid	Lorentzos
—	[140]	valid	Lum
—	[145]	both	McKenzie
Temporal Relational Model	[150]	valid	Navathe
—	[166]	valid	Sadeghi
—	[170]	valid	Sarda
Temporal Data Model	[175]	valid	Segev
—	[186]	both	Snodgrass
—	[208]	valid	Tansel
Time Oriented Databank Model	[222]	valid	Wiederhold
—	[229]	both	Yau

Table 1: Temporal relational data models

Data Model Name	*Citation*	*Temporal Dimension(s)*	*Transaction Timestamp Representation*	*Identifier*
—	[34]	both	chronon	Caruso
IRIS	[20]	transaction	chronon, identifier	IRIS
—	[111]	transaction	version hierarchy	Kim
MATISSE	[4]	transaction	chronon, identifier	MATISSE
OODAPLEX	[228]	arbitrary	arbitrary	OODAPLEX
OSAM*/T	[206]	valid	N/A	OSAM*/T
OVM	[103]	transaction	identifier	OVM
Postgres	[205]	transaction	interval	Postgres
—	[172]	arbitrary	arbitrary	Sciore-1
—	[173]	both	chronon	Sciore-2
TEDM	[41]	valid	N/A	TEDM
TIGUKAT	[68]	both	identifier	TIGUKAT
TMAD	[104]	valid	N/A	TMAD
Temporal Object-Oriented Data Model	[163]	both	temporal element	TOODM

Table 2: Temporal object-oriented data models

	Single chronon	Interval (pair of chronons)	Valid-time Element (set of chronons)
Time-stamped attribute values	ADM Caruso Lorentzos	Bassiouni Gadia-2 McKenzie Tansel	Bhargava Gadia-1 HRDM TOODM
Time-stamped Groups of Attributes	Sciore-2		
Time-stamped tuples	Ariav HDM Lum Sadeghi Segev Wiederhold	Ahn Ben-Zvi Jones Navathe Sarda Snodgrass Yau	BCDM
Timestamped Objects	TEDM	OSAM*/T TMAD	

Table 3: Representation of valid time

	Single chronon	Interval (pair of chronons)	Three Chronons	Transaction-time element (set of chronons)	Other
Time-stamped attribute values	Caruso			Bhargava TOODM	Sciore-1
Time-stamped Groups of Attributes	Sciore-2				OVM
Time-stamped tuples	Ariav DATA DM/T Lomet	Snodgrass Postgres Yau	Ben-Zvi	BCDM	
Time-stamped Objects	IRIS TIGUKAT				IRIS Kim
Time-stamped sets of tuples	ADM Ahn	McKenzie			
Object Graph	MATISSE TIGUKAT				MATISSE
Time-stamped Schema	MATISSE TIGUKAT	McKenzie Postgres		BCDM	MATISSE

Table 4: Representation of transaction time

3.1.2 Transaction Time

The same general issues are involved in transaction time, but there are about thrice as many alternatives, in part because transaction time is often used to support versioning, which generally implies an object-oriented data model. The choices made in the various data models are characterized in Table 4. OODAPLEX is not included, as it can support virtually any of these options (while that is also possible in TIGUKAT, specific support for versioning has been added to the data model and language). More detail on the representation of "Other" may be found in the fourth column of Table 2. Specifically, those data models supporting versions often allow arbitrary user-supplied identifiers to be associated with versions. One model even allows an entire version hierarchy to be associated with a version.

3.1.3 A Suite of Temporal Data Models

A temporal data model should simultaneously satisfy many goals. It should capture the semantics of the application to be modeled in a clear and concise fashion. It should be a consistent, minimal extension of an existing data model, such as the relational model. It is best if the temporal data model presents all the time-varying behaviour of a fact or object coherently. The data model should be easy to implement, while attaining high performance.

The experience of the last fifteen years and some forty data models appearing in the literature agures that designing a temporal data model with all of these characteristics is elusive at best, and probably not possible.

TSQL2 takes a different tack [99]. This language employs the Bitemporal Conceptual Data Model as its underlying data model, in which temporal databases are designed and queries are expressed. This data model retains the simplicity and generality of the relational model. A separate, *representational* data model, of equivalent expressive power, is employed for implementation and for ensuring high performance. Other *presentational* data models may be used to render time-varying behavior to the user or application. A coordinated suite of data models can achieve in concert goals that no single data model could attain.

3.2 Real-time Data Models

Until recently, real-time database research in the literature has not specifically dealt with data modeling issues, much less temporal data modeling issues. Most of the research literature, especially that concerning real-time transaction processing, assumes a database having data items with varying granularities. This approach has limitations since it means that the semantic knowledge (in general and of time) that may be very useful to the system for meeting transaction deadlines is not utilized.

The relational model is used as a data model for real-time databases [190]; but, it has not been extended with the semantics of time by researchers. There are, however, two real-time query processing approaches proposed in the literature that modify the relational model: (i) the use of sets of *approximate relations* [184, 220] defined for any relation in the relational model for timely evaluation of queries, and revised for a better approximation and for a "better" query response iteratively, and (ii) the use of *fragmentation lattices* of relations [84, 155], also for query processing purposes. Both of these approaches do not add a temporal dimension into the data model.

Recently, there have been some initial research into adapting the object-oriented data model for real-time databases, both because one can utilize the rich data semantics in real-time transaction processing [54], and because complex real-time applications may need the advantages provided by the object-oriented data model [125] such as complex objects, encapsulation, methods and messages. Significant research is needed to provide these features in real-time databases where timeliness and meeting time guarantees are required. To give an example, storage of complex objects and implementations of complex object algebra operators [154] carry high time costs that need to be reduced for real-time databases. More importantly, time cost formulas for such operators are highly parameterized, making it difficult to provide guarantees for expected complex object algebra time costs with small variances. As another example, encapsulation for objects in object-oriented databases brings with it a number of advantages [16]. However, encapsulation also forces users to access objects in restricted ways, possibly delaying timely executions of transactions. Clearly, there is a balance between providing new data modeling features and ensuring timeliness in database operations, that needs to be further investigated.

DiPippo et al proposes a real-time object-oriented database that supports a rich variety of data semantics, temporal consistency constraints, and a range of transaction correctness criteria that relax serializability [54]. This model also permits bounded temporary "imprecision" in data values and in transaction output by allowing concurrent, but not serializable, executions of transactions (methods). The idea is to let time-constrained transactions complete with incorrect data as long as the introduced error (i.e., "imprecision") is bounded. Lee and Son propose a simple real-time object-oriented database system having atomic objects and a class manager [125]. Both of the proposed approaches are being implemented, but have not yet been experimentally evaluated.

Various temporal data models and temporal query languages may be more suitable for particular real-time databases. The choice of temporal data model and query language will have an as yet undefined effect on how to provide completion time guarantees for queries.

4 Temporal Query Languages

4.1 Overview

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 [212]. In the two previous sections 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.

Table 5 lists the major temporal relational query language proposals to date. The underlying data model is a reference to Table 1. The next column lists the conventional query language the temporal proposal is based on, from the following: DEAL, an extension of the relational algebra incorporating functions, recursion, and deduction [52]; IL_s, an intensional logic formulated in the context of computational linguistics [147]; QBE, Query-by-Example [232], a domain calculus-based query language; Quel, the tuple calculus-based query language [82], originally defined for the Ingres relational DBMS; relational algebra, a procedural language with relations as objects [46]; and SQL, a tuple calculus-based language, the lingua franca of conventional rela-

Name	*Citation*	*Underlying Data Model*	*Based On*	*Formal Semantics*	*Underlying Algebra*
HQL	[167]	Sadeghi	DEAL	partial	[166]
HQuel	[210]	Tansel	Quel	yes	[208]
HSQL	[170]	Sarda	SQL	no	[169]
HTQuel	[64]	Gadia-1	Quel	yes	[64]
Legol 2.0	[101]	Jones	relational algebra	no	N/A
TDM	[175]	Segev	SQL	no	
Temporal Relational Algebra	[139]	Lorentzos	relational algebra	yes	N/A
TempSQL	[229]	Yau	SQL	yes	
Time-By-Example	[209]	Tansel	QBE	yes	[208]
TOSQL	[14]	Ariav	SQL	no	
TQuel	[186]	Snodgrass	Quel	yes	[145]
TSQL	[150]	Navathe	SQL	no	
TSQL2	[188]	BCDM	SQL-92	no	[198]
—	[211]	ADM	relational algebra	yes	N/A
—	[17]	Bassiouni	Quel	yes	
—	[21]	Ben-Zvi	SQL	yes	[21]
—	[92, 93]	DM/T	relational algebra	yes	IM/T [94]
—	[63]	Gadia-2	Quel	no	
—	[45]	HDM	IL_s	yes	
—	[42]	HRDM	relational algebra	yes	N/A
—	[145]	McKenzie	relational algebra	yes	N/A

Table 5: Temporal relational query languages

tional databases [146]. Most of the temporal relational query languages have a formal definition. Some of the calculus-based query languages have an associated algebra that provides a means of evaluating queries.

Table 6 lists the object-oriented query languages that support time. Note that many "nested" relational query languages and data models, such as HQuel, HRDM, HTQuel, TempSQL and TBE, have features that might be considered to be object-oriented.

The data model and conventional query language on which the temporal query language is

Name	*Citation*	*Underlying Data Model*	*Based On*	*Impl.*	*Underlying Algebra*
MATISSE	[4]	MATISSE	SQL	√	
OODAPLEX	[228]	OODAPLEX	DAPLEX		[50]
OSQL	[20]	IRIS	SQL	√	
OQL	[103]	OVM	SQL	√	
OQL/T	[206]	OSAM*/T	OSAM*/OQL		TA-algebra
Orion	[111]	Kim	SQL	√	
PICQUERY$^+$	[31]	TEDM	PICQUERY	√	
Postquel	[205]	Postgres	Quel	√	
TMQL	[104]	TMAD	SQL		
TQL	[156]	TIGUKAT	SQL	√	
TOOSQL	[165]	TOODM	SQL	√	[164]
TOSQL	[163]	TOODM	SQL		[164]
VISION	[34]	Caruso	meta-functions	√	
—	[172]	Sciore-1	annotations		
—	[173]	Sciore-2	EXTRA/EXCESS		[32]

Table 6: Temporal object-oriented query languages

based are identified in the third and fourth columns. The fourth column indicates whether the language has been implemented. A few proposals provide an algebra for their query language. It is rare for an object-oriented query language to have a formal semantics.

We do not consider the related topics of *temporal reasoning* (also termed *inferencing* or *rule-based search*) [105, 127, 148, 200] and *temporal abstraction* [181, 182] that uses artificial intelligence techniques to perform more sophisticated analyses of temporal relationships and intervals, generally with much lower query processing efficiency. Also not included are knowledge representation languages, such as Telos [149] or TSOS [15], which, while supporting either valid or transaction time, or both, are not strictly object-oriented query languages.

We first examine user-defined time, as it is the most straightforward aspect of time to support, in both the data model and in the query language. Subsequent sections consider valid and transaction time.

4.2 User-defined Time

User-defined time is supported by most commercial relational DBMSs as another domain that can be associated with attributes. Hewlett-Packard's object-oriented query language OSQL [141] and UniSQL [109] continue in the SQL tradition by including `Date`, `Time`, and `Timestamp` types.

Overmyer proposed making time an abstract data type, with its own set of operations [153]. Many of the object-oriented query languages (e.g., Postgres [205] and ZQL[27]) can support user-defined time in just this fashion.

Others have advocated that various notions of time (c.f., Section 2) should be supported more directly, as either a regular or a primitive class (i.e., a class with no attributes). The primary benefit is that the power of the object-oriented data model (with subtyping, inheritance, and polymorphic functions with late binding) can be used to express the semantics of time appropriate to the application.

4.3 Valid Time

Valid time support can have a much greater impact than support for user-defined time. There have been three general approaches to adding valid-time support to a data model and query language. We discuss each in turn.

The first approach utilizes the substantial expressive power of the relational or object-oriented data model directly, and thus requires no changes either to the model or to the query language to support time-varying information. Lorentzos' temporal relational algebra is the defining example in the relational domain [139]. Within object-oriented query languages, OODAPLEX and TQL utilize this approach. The OODAPLEX type system, supporting the parameterized types of set, multiset, tuple, and function, is sufficient for modeling temporal information. The TIGUKAT type system is also sufficient. The advantages are that various semantics of valid time can be specified by the user (as also holds for user-defined time), and the schema specification and query languages remain uncluttered with additional time-specific clauses, assuming, of course, that the necessary features, including functions and sets as first-class citizens, are available. The user is required to "roll their own" support for time when specifying the schema, and also when specifying queries. This approach renders query optimization much more difficult, as there are no hints provided in the language that access methods or storage structures oriented towards time-varying values should be employed.

A second approach is to include general extensions to the data model and query language for other reasons, and then show how these extensions may be used to support time-varying information. This approach has been used only with object-oriented query languages. Sciore has advocated the use of *annotations* [202] to support the various kinds of versioning, including histories, revisions, and alternatives [172]. The VISION system adopted a similar approach. Query optimization is still difficult, as all manipulation of time is done by user-defined functions (which themselves could perhaps be individually optimized).

In contrast to the previous approaches, most researchers have proposed specific data modeling and query language constructs to support information varying over valid time. This is the approach adopted by the vast majority of temporal relational query languages. Most add numerous new constructs and temporal operators, yet attempt to retain *snapshot reducibility* [186] to the non-temporal query language on which they are based, which ensures that the user's intuition about the base language carries over to the temporal extension.

Turning to object-oriented query languages, TOODM, the data model behind TOSQL and TOOSQL, encodes attribute values as *time sequences*, which are sequences of (value, temporal element) pairs [163, 164]. The OQL/T query language for the temporal object-oriented knowledge representation model OSAM*/T incorporates an optional when clause, as well as a set of temporal functions and operators [206]. Sciore has extended the EXTRA data model [32] to differentiate between versioned and unversioned attributes, and made several changes to the EXCESS query language to support selection of times and default times, termed *contexts* [173]. TEDM supports evolution, fusion, and fission of objects over time [31]. The Illustra DBMS, a commercialization of Postgres, includes a time series datablade that allows (value, timestamp) sequences as attribute values.

4.4 Transaction Time

Transaction time concerns when facts were logically present in the database, as opposed to when the facts were true in reality. Supporting transaction time is necessary when one would like to *roll back* the state of the database to a previous point in time. Transaction time is also useful when storing versions, say, of an engineering design. In such situations, transaction time is often branching, versus the linear time model underlying valid time.

In considering support for transaction time, an important distinction must be made: either the tuples, object instances or attributes are themselves versioned (termed *extension versioning*), or the *definitions* of those objects are versioned (termed *schema versioning*). If extension versioning is adopted, then schema versioning may or may not be supported. If extension versioning is not supported, then schema versioning is not relevant, as only the most recent version of the schema need be retained.

4.4.1 Extension Versioning

As with support for valid time, there are three general approaches to supporting extension versioning. The first is to use the model directly, making no changes to the data model or query language. OODAPLEX follows this approach. Since the semantics of time is arbitrary (the user can implement whatever is desired), transaction time can be accommodated.

In the second approach, general extensions to the data model and query language are exploited to support time-varying information. Sciore's annotations, discussed earlier, can be used to support revisions and alternatives (i.e., branching transaction time). Generic references are also capable of supporting transaction time [173], as are also the meta-functions in VISION [34].

In the third approach, the data model and language are modified to explicitly support transaction time. As with valid time, the majority of proposals are in this camp. TSQL2 is perhaps the most ambitious, in that it supports both arbitrary expressions on the transaction-time timestamp associated with tuples, and *vacuuming*, in which old data is purged from the system to reduce secondary storage requirements [96, 205].

The Chou-Kim versioning model [40, 108] has garnered the widest acceptance in the object-oriented community; it has been implemented in ORION [111], in the IRIS object-oriented DBMS [20, 224] (though this support has not yet been incorporated into the commercial version and its query language OSQL), and in OQL [102]. Objects are versioned with branching transaction time, with the added feature of two versions being merged to create a new version, hence transaction time is truly a graph, rather than a tree.

4.4.2 Schema Versioning

In schema *evolution*, the schema can change in response to the varying needs of the application. In schema *versioning*, there are multiple schemas logically in effect. Schema versioning has been examined both in the context of relational databases [21, 143, 161, 162] and in the context of object-oriented databases. The Chou-Kim versioning model accommodates the most extensive form of schema versioning of the object-oriented data models. Schema versioning under the Chou-Kim model has not yet been implemented, though detailed data structures, storage representations, and object accessing algorithms have been presented [110]. Multiple schemas may also be defined via *object-oriented views* [23] or *semantic contexts* [13] (which should be differentiated from Sciore's contexts [173]). An essential difference between these approaches and schema versioning is that in the latter an object created in a specific schema version is only visible in that schema version, rather than in all views or contexts [24].

Having examined temporal data models and query languages in some detail, we now turn to th data models and query languages supporting real-time applications.

5 Real-Time Data and Transaction Properties

Real-time transactions must be timely, i.e., they must be scheduled in such a way that they complete within their time constraints (deadlines) and satisfy transaction constraints. Timely execution of transactions requires good estimates for their worst-case execution times, which is very difficult to obtain when (i) transactions' execution times depend on the values of data items they access (e.g., range of a looping construct in a transaction is defined through the value of a data item), (ii) transactions block or abort due to data and resource conflicts, and (iii) I/O scheduling or buffer management techniques used also affect transaction execution times.

Hard transactions are those for which missing a deadline is disastrous and must not happen. If completed transactions are assigned values (to measure the benefits gained by completing them), a hard transaction with a missed deadline is given a large negative value. *Soft transactions* are those that may miss their deadlines and still have some, monotonically decreasing, value assignments that drop to zero at some point P in time. *Firm transactions* are those for which P and the deadline are the same.

The following factors [120, 159, 213, 231] are used to characterize real-time transactions, and influence the transaction processing techniques used to schedule them [120].

(a) *The implication of missing a specified transaction deadline*: hard, soft, or firm transactions.

(b) *Transaction arrival pattern*, which is one of periodic, sporadic, or aperiodic.

(c) *Data access types*, which may be random (i.e., unknown) or predefined with (i) write-only (update) transactions that collect information about the state of the real-world and write into the database, or (ii) read-only transactions that read the values of data items and modify the state of the real-world, through specialized output devices.

(d) *Accessed object properties.* The real-world object whose state is maintained by a data item in the database may be *continuous* (i.e., its state always has a value and may change at any time) or *discrete.*

(e) *Knowledge of items to be used*, i.e., whether the accessed items are known a priori.

(f) *CPU and I/O time knowledge*, i.e., whether the CPU and I/O usage of transactions are known a priori.

For some real-time applications, it is desirable to have real-time databases that provide a (preferably semantically rich) model of data, satisfy integrity constraints, and permit only serializable executions [22] of transactions. Such databases are said to be *internally consistent* [132]. Clearly, for these applications, traditionally accepted transaction correctness properties, i.e., ACID (Atomicity, Consistency, Isolation, and Durability) properties, are adhered to [72].

For some real-time databases, having an externally consistent database may be more important than having serializable transactions [132]. A database is *externally consistent* if, whenever a real-world object changes its value, its counterpart data item in the database (if it exists) is also changed. External consistency requires that the data used by a transaction always reflect the physical environment at the time. In comparison, the internal consistency requires that the data in the database satisfy predefined constraints. In applications where a timely response to external state changes are needed (e.g., auto-pilot systems, automated factories with robots), external consistency may take precedence over internal consistency. Thus, for some applications, executing transactions that maintain external consistency take precedence over having serializable transactions or over having the integrity constraints of the database satisfied all the time [132].

Clearly, it is possible to have an externally consistent database and serializable transactions that always use the most recent values of data items: one can detect, at a cost, transactions that use externally inconsistent data, and abort or roll them back. But, such an approach requires transaction abortions and rollbacks, and may be too wasteful. On the other hand, it is not possible to have both external consistency and integrity constraint satisfaction, as newly inserted values of external objects may cause immediate integrity constraint violations. For example, an integrity constraint involving a factory furnace temperature and the furnace coolant level may be violated by rising temperature levels. Also, external consistency maintenance may lead to triggers (for example, the violation of the factory furnace constraint may trigger a transaction that informs an operator or requests an automated agent to increase coolant levels); transactions that need the most up-to-date data item values of external objects may require abortions (for example, a transaction that performs periodic computations and initiates physical activities on objects produced in the furnace may need to be aborted and restarted due to furnace temperature changes); or integrity constraint violations may need to be resolved by new transactions. The interplay between external and internal consistency is an unexplored research area.

The discussion above suggests that, at least for some real-time database applications, transactions interact and synchronize with each other. That is, transactions in such environments *cooperate* with each other. In comparison, transactions in traditional databases are isolated, and *compete* with each other for completion. The concept of cooperating transactions has three implications.

(i) The traditional correctness notion, i.e., serializability, for transaction execution needs to be extended or replaced by new correctness notions. The next section discusses two such correctness notions.

(ii) Transaction interaction in some real-time database environments has similarities with those encountered in active databases [67] where various events and conditions trigger actions of other transactions. One example is "if transaction *Furnace-Temp-Alarm* writes object

Alarm between now and (now + 1 minute) then start and complete transaction *Notify-Automated-Agent* in 1 second". One major difference between this example and active databases is that the transaction *Notify-Automated-Agent* is requested to be completed within a time deadline.

(iii) Transactions in these new real-time database environments now synchronize (i.e., cooperate) with each other much like processes (tasks) of concurrent computing models. For example, the periodic transaction *Objects-Produced-in-Furnace*, upon seeing an old furnace temperature value, may "wait" for a "recent" value of data item *Furnace-Temperature* to be added into the database by transaction *Modify-Furnace-Temperature* and then continue execution. Such waits can be specified and controlled either by an independent agent such as a transaction manager/scheduler that enforces rules, or directly using process communication techniques such as semaphore waits and signals or message passing. A survey of external and internal consistency definitions concluded that "there may be a room for a theory of interactions of programs that are both cooperating" (as in concurrent processes) "and competitive" (as in transactions) [69]. Our view is that this theory of interaction among "programs", regardless of whether they are called programs or transactions, needs to be investigated by real-time database researchers as these programs interact directly with a database. This survey also contains an insightful comparison between a transaction and a process.

Another major difference between conventional and real-time database transaction processing is their approach to resolving data and resource conflicts. Conventional databases attempt either to be *fair* in data and resource allocation or to *maximize resource utilization*. In real-time databases, *timely execution of transactions* is more important, and both fairness and maximum resource utilization become secondary goals. A related issue is measuring transaction processing performance. In comparison with conventional databases that use transaction response time and throughput as performance measures, real-time databases use the percentage of transactions that complete within their deadlines or the total value of completed transactions using a function that assigns values to completed transactions. Finally, real-time transactions are prioritized on the basis of their deadlines and values, and the transaction manager uses transaction priorities in transaction scheduling as well as in conflict resolution among transactions.

5.1 Real-Time Temporal Data and Transaction Consistency

When real-time databases are used in safety-critical real-time systems, the validity of data as well as its correct use by transactions becomes very important. Real-time temporal data constraints originate from the requirements that the age of data in the database is important for some real-time transactions, and that, sometimes, a set of data values stored in the database need to have similar age values [132, 135, 219].

We make a distinction below between *base* (data) *items* that correspond to external objects, and whose values are associated with valid times recorded by specialized input devices of a real-time system, and *derived* (data) *items*, whose values are associated with absolute transaction times. Assume that each base item value has an *absolute valid time* (indicating the real-world data observation time) and an *validity interval*, the length of the time following the absolute valid time during which the value is considered to be valid. Real-time temporal data constraints have two components [136], the first of which is readily applicable to any temporal database, not just a real-time database [98].

(a) *Absolute data consistency* states that the validity interval of the most recent value of a base (or derived) item is always bigger than the time interval elapsed between its absolute valid (or absolute transaction) time and *now*. This indicates that the data has absolute validity.

(b) *Relative data consistency.* Frequently in real-time databases, a set of data items must be observed within a small time interval so that they can be used to derive a new value for a data item. Such a set of data items forms a *relatively consistent set of items.* For example, the temperature and the pressure of a furnace together may be used to compute the "safety level" of the furnace, and thus, need to be measured close in time. These two data items, together with the safety level item, form a relatively consistent set of items. Note that each relatively consistent set is associated with a time interval, called the *relative validity interval.* We can now define the *current* relatively consistent set of items as follows. Consider the most recent value v of a base or derived item d and any relatively consistent set R that contains d. d is *current relative set consistent with respect to* R if the time distance between the time of v and the time of the most recent value of each item in R is less than the relative validity interval of R. One can also define *total relative set consistency* of d with respect to R as follows. For any value w of d, there exists a set S of values, one for each item in R, such that the time distance between w and any value in S is less than the relative validity interval of R.

One currently open research problem on real-time temporal data constraints involves enforcing relative consistency of data items in a database when there are multiple and overlapping relatively consistent sets and/or when new data items are being added to and deleted from the database.

The consistency of real-time transactions are related to the temporal consistency requirements of data items discussed above. Real-time databases are commonly used by real-time systems to store data about physical devices and to set parameters of physical devices. Therefore, transactions need to read the "most recent" values of data items as defined by the two constraints below [231].

(i) *External transaction consistency.* Ideally, most real-time transactions would like to read and use the current values of external objects in their computations. However, this may not be possible as the most recent values in the database may differ from the current values of real-world objects. External transaction consistency states that the difference between the time of a transaction operation on a data value and the absolute valid time of the data value is less than a given small threshold.

(ii) *Temporal transaction consistency.* Real-time transactions may use a snapshot of the real-world. Therefore, data values read by a transaction are desired to have the same valid or transaction times. Again, this may not be possible. Let V denote the values of a set of items that a given transaction T reads. Then T is *temporally consistent* if the difference between the valid or transaction times of any two base or derived values in V is less than a given small threshold.

The performance of a class of lock-based, multiversion concurrency control algorithms in maintaining temporal transaction consistency for periodic transactions has been empirically evaluated under mixed workloads of read-only, write-only, and read-write transactions [196]. Maintaining temporal transaction consistency was found to be easier when conflicting transactions are close in the lengths of their periods. Also, the conflict pattern of transactions was found to have a more significant effect on temporal transaction consistency than the transaction load levels.

Assume that there are write-only transactions that periodically record the changes in the real-world into the database. Two real-time transaction consistency enforcement issues are: (a) given a set of transactions with consistency requirements, find the period of each write-only transaction so that the consistency requirements of other transactions are satisfied; and (b) given a set of periodic write-only transactions, find the level of real-time transaction consistency that can be guaranteed [131].

Transaction consistency notions other than those described above may be useful. For example, referring to the temporal transaction consistency given above, there may be distinct threshold values for different subsets of V. It is also possible to define a range of transaction correctness criteria that relax serializability so that interleaved executions of transactions that use incorrect item values (but bounded in their absolute differences with the correct item values) are permitted [54]. More research is needed in finding new, general-purpose consistency notions for cooperating transactions in real-time databases.

5.2 Real-Time Query Languages

There is no reported research on real-time database query languages or transaction specification languages that can allow users to specify time constraints, temporal constraints, temporal properties of data values, and semantic knowledge useful for query/transaction processing. This is because, until recently, specifying a query or a transaction involved only the specifications of a time constraint and temporal transaction constraints, which can be achieved by simple extensions to conventional query languages or transaction specification techniques. We think that, in the near future, there will be research needed in real-time query and transaction specification languages that will allow users to specify semantic knowledge and interactions between cooperating transactions, and that are based on various temporal data models. As described at the beginning of Section 5, these languages may also have their basis on temporal and modal logics [59] or on models of concurrent processes [123].

6 Architectural Issues

We now turn to the implementation of the temporal and real-time data models and query languages. For both temporal and real-time databases, we focus on relational databases; there is little experience with implementing temporal or real-time object-oriented databases.

6.1 Query Processing of Temporal Queries

There are two aspects discussed here, query optimization and query evaluation.

Optimization of temporal queries is substantially more involved than that for conventional queries, for several reasons. 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 may be larger, and often grow monotonically, implying that unoptimized queries take longer and longer to execute. It is reasonable to try harder to optimize queries on such data, and to spend more execution time to perform the optimization.

The predicates used in temporal queries are harder to optimize [128, 129]. In traditional database applications, queries generally specify equality predicates (hence the prevalence of equi-joins and natural joins); if an inequality predicate is involved, it is rarely in combination with other such predicates. In contrast, in temporal queries, joins with a conjunction of several inequality predicates appear more frequently. As an example, the TSQL2 `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.

On the other hand, there is greater opportunity for query optimization when time is present [129]. Time advances in one direction: the (transaction) time domain is continuous expanding, and the most recent time point is the largest value in the domain. This implies that a natural clustering on sort order will manifest itself, which can be exploited during query optimization and evaluation. Query optimization can also consider time-oriented integrity constraints. The integrity constraint $begin(t) \leq end(t)$ 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 [53, 175, 176]. 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 [61, 178, 183].

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 [8]. The results were very discouraging. 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 did 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. We emphasize that these results do *not* imply that converting a time-varying database implemented on a conventional DBMS will be much less efficient on a brute-force temporal DBMS. In fact, simulating a time-varying database on a conventional DBMS, which is currently the only alternative available to application programmers, will encounter all of the problems listed above.

There have been four basic responses to this challenge. The first was a proposal to separate the valid-time and transaction-time data, which grows monotonically, from the current data, whose size is fairly stable and whose access is more frequent [140]. This separation, termed *temporal partitioning*, was shown to significantly improve performance of some queries [9], and was later generalized to allow multiple cached states, which further improves performance [94]. The other three responses, to be discussed below, were the design of new query optimization strategies, new join algorithms, and new temporal indexes.

6.1.1 Query Optimization

A single query can be optimized 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 many of the algebras listed in Table 5. Some of these temporal algebras support the tautologies defined in the standard relational algebra, 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 the time interval over which the relation is defined (termed the *lifespan* [42]), 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 the null values that are sometimes introduced when attributes within a tuple aren't synchronized [174]. 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 [9, 10], and selectivity estimates on the size of the results of various temporal joins have been derived [70, 174].

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 [171, 177]. A state transition network appears to be a good way to organize this complex task [94]. Materialized views 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., [92]).

6.1.2 Temporal Joins

A wide variety of binary joins have been considered, including *time-join* and *time-equijoin* (TE-join) [42], *event-join* and *TE-outerjoin* [71], *contain-join*, *contain-semijoin* and *intersect-join* [129], and *temporal natural join* [199]. The various algorithms proposed for these joins have generally been extensions to nested loop or merge joins that exploit sort orders or local workspace, as well as hash joins. More work is necessary to design a join strategy that is superior over most of the parameter space.

6.1.3 Temporal Indexes

Conventional indexes have long been used 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. There has been a great deal of research over the past five years in temporal indexing. The worst-case performance for most proposals has been evaluated, in terms of total space required, update per change, and several important queries [168]. Average case analysis is of course much more difficult. While preliminary performance studies have been carried out for each of these indexes in isolation, there has been little effort to empirically compare them. Such an empirical 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).

6.2 Transaction Processing

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 that effectively eliminate the need for additional reads [47, 138, 204]. (We revisit this issue in the context of real-time databases in Section 6.2.2.) Time-stamping in a distributed setting has also been considered, as has integrating temporal indexes with concurrency control to increase the available concurrency [138]. 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 [138].

In the remainder of this section, we discuss the issues related to processing real-time transactions. Evaluations of the techniques discussed, if any, are invariably by either a testbed system, e.g., [87] or through simulation, e.g., [3, 76, 215]. One exception is the approximate analysis of real-time DBs [79] where an analytical approach is used to approximate the steady state fraction of real-time transactions that complete successfully.

6.2.1 Processing Transactions with Hard Deadlines

All transactions with hard deadlines must complete within their deadlines. This means that they need to be scheduled with complete knowledge, which in turn means that various factors about transactions as listed in Section 5 must be known a priori. For example, transaction arrival patterns must be known, i.e., transactions must be periodic. Also, transaction data access types, items to be accessed, and CPU and I/O access times must be known. With this knowledge, tight estimates of worst-case execution times of transactions can be obtained, and real-time task scheduling techniques can be used to guarantee timely execution of transactions.

6.2.2 Processing Transactions with Soft Deadlines

Transaction Priority Assignment Policies: Priorities and values for real-time transactions are used for conflict resolution and CPU scheduling. The literature contains various transaction priority or value assignment algorithms [26, 75, 76, 86, 91, 134] and their evaluations [3, 1, 86]. Some of these policies are *Earliest-deadline-first*, *Highest-value-first*, *Least-slack-time-first* (where slack time is the maximum amount of time that a transaction can spend without executing, and still complete within its deadline), *Fixed-priority-with-a-priority-ceiling*, and *Weighted-priorities*. One interesting approach is to use a *dynamic priority assignment* policy in which a continuous workload evaluation method evaluates the priority of a transaction several times during its execution [83]. Priority assignment algorithms are important since they directly influence the performance of the transaction scheduling algorithms.

Concurrency Control Techniques with Serializability: Concurrency control techniques for real-time databases that use serializability as the correctness criteria include *lock-based protocols* such as two-phase locking and its variants [3, 1, 7, 60, 86, 88, 179, 180, 193, 214, 216], *optimistic concurrency control protocols* [74, 78, 88, 121], and *timestamp-ordering protocols* [133, 195, 215]. Using any of these techniques, conflicts between two real-time transactions or between one real-time transaction and a set of real-time transactions are detected.

For lock-based protocols, transaction conflicts are resolved by either transaction blocking or transaction abort. When a data item held by a transaction is requested by another transaction with a *conflicting* lock request such as a write-lock, the alternatives for resolving such a conflict are to block or abort the lock-requesting transaction, or to abort the lock-holding transaction. When a higher priority transaction is blocked by a lower priority transaction during conflict resolution, *priority inversion* results. One way to avoid this unfortunate situation is to use *priority inheritance* [179], in which the lower priority transaction that is blocking other transactions inherits the highest priority of the transactions it blocks, until it releases the lock. Note that, even with an inherited priority, a transaction may later block other higher priority transactions, thereby increasing its priority even further.

The *Priority abort* approach [3], in constrast, grants the lock to the higher priority transaction. If the lock-requesting transaction has higher priority, it is granted the lock after the lock-holding transaction is aborted. Otherwise, the lock-requesting transaction is blocked. The performance of both priority inheritance and priority abort as well as their variations have been compared in several studies [1, 3, 77, 89, 215].

Other lock-based protocols for real-time databases include *ordered sharing* [5, 6, 7] that eliminates blocking, and *dynamic adjustment of the serialization order* [133] of transactions in order to execute high-priority transactions before low-priority transactions.

Timestamp-ordering protocols [194, 215, 216] use timestamps that are assigned to transactions when they are started, for resolving conflicts during transaction execution. Compared with the optimistic concurrency control that resolves transaction conflicts at the end of transaction execution and during transaction validation, early conflict resolution of timestamp-ordering protocols is an advantage. On the negative side, timestamp-ordering protocols suffer from priority inversion in the sense that a higher priority transaction is aborted when it attempts to access a data item modified by a lower-priority transaction with a higher timestamp value. Different priority-based timestamp-ordering protocols are proposed and evaluated in the literature [195]. One interesting approach is to assign timestamps to transactions dynamically whenever actual conflicts occur [19]. A similar approach is to assign timestamp intervals to transactions instead of a single value, and to adjust (reduce) the timestamp intervals when conflicts arise to guarantee serializable transactions [19, 230].

Optimistic concurrency control protocols validate (certify) transactions for commitment after they complete their executions [74, 77, 78, 87]. Conflicts among transactions are solved using aborts and restarts. The advantage of the optimistic concurrency control technique is that it is nonblocking and deadlock-free, making it attractive for real-time databases. On the other hand, transaction aborts and restarts waste resources which may be critical for real-time databases.

Optimistic protocols can use *backward validation*, in which the validating transaction is checked against committed transactions, and is either aborted due to conflicts or committed. An alternative is to use *forward validation*, in which the validating transaction is checked against the currently running, active transactions, and, in the case of conflicts, either the validating transaction or other conflicting transactions are aborted. The literature contains various real-time optimistic protocol variants that are based on the forward validation and their experimental evaluations [74, 87]. Instead of the knowledge of the dynamic read sets of active transactions, these variants utilize (i) transaction priorities [74, 88], (ii) forcing the validating transaction into a wait state until the conflicts are resolved [74], or (iii) using dynamic adjustment of the serialization order [124].

Other concurrency control techniques for real-time databases include using multiple copies of data [119, 213], and hybrid protocols that combine lock-based, timestamp-ordering-based, and optimistic concurrency control techniques [195]. Comparative evaluations of most of these techniques are not yet available.

The discussion in this section so far has assumed the use of *flat* transactions. When a transaction consists of subtransactions, each to be executed in a distributed database environment, varying deadlines for subtransactions can be assigned individually in order to reflect the importance and the urgency of each subtransaction [159]. Several heuristics to this problem, called the *subtask deadline assignment* problem, are proposed and evaluated [107], both for serial subtransactions where subtransactions execute serially, and for parallel subtransactions where subtransactions execute in parallel. However, only one-level nested transactions are accommodated. The problem has not been investigated for nested transactions with arbitrary depth. In fact, very little research has considered using nested transactions in a real-time database environment.

Concurrency Control Techniques that Relax Serializability: For real-time databases where serializability of transactions is not absolutely necessary, the performance may be improved by allowing nonserializable or temporarily nonserializable transaction executions. One such approach is the *epsilon serializability* [106, 158, 192, 226] which allows bounded inconsistency during conflict resolution. Conflicting accesses for read-only transactions due to read-write conflicts are permitted using a *divergence control* algorithm as long as inconsistencies are within a prespecified limit. Another approach is to use application semantics and data similarities to obtain higher levels of concurrency among transactions [116, 118]. Imprecise data values and partial computations are utilized for the same purpose in [130].

6.3 Stored Data Manager

Many storage structures have been proposed, including *reverse chaining* (all history versions for a key are linked in reverse order) [21, 47, 140], *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 being used to compare their space and time efficiency [9]. Page layout for temporal relations is more complicated than for 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

Section 3. Often such attributes are stored as linked lists, for example representing a valid-time element (set of valid-time chronons) as a linked list of intervals. Hsu has developed an analytical model to determine the optimal block size for such linked lists [85].

6.4 Buffer Management for Real-Time Transaction Scheduling

In conventional databases, buffer contention among transactions can impact the performance significantly. Prioritized versions of two buffer management algorithms, namely, Priority-Least-Recently-Used (Priority-LRU), and Priority-DBMIN have been investigated [33]. Priority-LRU groups buffer elements into priority levels, where each level holds the pages of transactions with the same priority. Pages are replaced from the LRU page of the lowest priority group. Priority-DBMIN allocates a set of buffer elements ("locality set") for each file to be accessed by each transaction before the transaction is admitted, and uses an optimum replacement policy for each locality set. An easier-to-implement version of Priority-DBMIN turns out to exhibit even better performance [90].

Another approach assumes that write requests are delayed to the end of transaction execution, and uses different queues to buffer read and write requests [2]. The idea is to have a minimum free buffer space in the write buffer for write requests, and always process read requests first as long as the write requests can be placed in the write buffer without reducing the free space into a size less than the specified minimum. A variant of this approach creates deadlines for writing the contents of a buffer element into the disk, and uses these deadlines to service write requests [2].

6.5 Scheduling Disk I/O for Real-Time Transaction Processing

Because disk I/O constitutes a significant part of transaction execution times, it is natural to revise disk I/O scheduling algorithms to ensure timely transaction executions. Conventional disk scheduling algorithms, called SCAN algorithms, sort I/O requests and service them by scanning disk cylinders. Several variants of a SCAN algorithm are proposed for real-time databases [2, 33], and are empirically evaluated. An algorithm, called FD-SCAN (Feasible Deadline SCAN), is shown to have a superior performance, where the scanning direction is decided by locating the I/O request with the earliest feasible deadline.

In another study [1], transactions are assumed to perform all write requests when they commit, and write requests have the lowest priority as they are related to transactions that have completed their executions. Read requests are then assigned priorities on the basis of their deadlines. Priority inheritance can be added into this scheme for write requests that block other high priority transactions (due to write locks) [119].

SCAN algorithms can also be revised using the knowledge of prioritized groups of disk requests [33]. Another approach tries to balance time constraints of disk requests and the overall I/O performance in terms of the average seek time [37].

7 Conclusions and Future Work

Temporal database research has been active for about twenty years. Initially the focus was on temporal relational databases. A decade ago research on temporal object-oriented data models was launched under the guise of versions in an engineering database [157]. Real-time database research is somewhat newer, with the first papers appearing in 1986.

There have been many significant accomplishments within the two fields.

- The semantics of the time domain, including its structure, dimensionality, indeterminacy, and real-time aspects, is well-understood.
- A great amount of research has been focused on temporal data models, including both relational and object-oriented models, addressing this extraordinarily complex and subtle design problem. Satisfying all desirable objectives within a single temporal data model is probably unattainable. Instead, a coordinated suite of data models, each tailored to a particular aspect, is a more appropriate approach.
- 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.
- Several commercial temporal OODBMS's are now on the market.
- The real-time and temporal database communities are starting to interact. A first step is the adoption of common terminology, and a delineation of shared concepts originating in these two areas.
- The interaction between transaction time support and concurrency control and transaction management has been studied to some depth. Most of the research in real-time databases has been on modifying and adapting both the traditional transaction processing techniques of conventional databases and the task scheduling techniques for real-time systems.

We feel that the following research areas need to be addressed.

- Real-time data models, supporting timely database operations, should be temporal, to capture semantic knowledge needed for timely execution of database operations.

- Temporal and real-time database design is still in its infancy, hindered by the plethora of temporal data models and the absence of real-time data models. With the emergence of a consensus temporal relational query language, TSQL2, the task of database modeling should be investigated within the context of this consensus language.

- In some real-time databases, transactions cooperate, rather than compete [132]. Such transaction interactions should be investigated. A theory of *cooperating transactions* needs to be developed for real-time databases, perhaps, similar to the cooperating transaction hierarchies [151], or cooperative software engineering environment transactions [81]. Such a theory may involve flat transactions and transaction execution rules, or nested transactions where subtransactions have execution rules. Also, this theory should be developed both with and without serializability as different applications may or may not need serializable transactions. The notion of a linear transaction time model will need to be modified to accommodate non-serial transactions.

- It should be determined whether the increased efficiency and ease of use justifies the added complexity of explicit temporal constructs in the object-oriented temporal data model and query language. This resolution is a prerequisite for the design of a consensus temporal object-oriented query language.

- Real-time query languages are needed for users to specify the semantic knowledge captured in real-time data models and to use it different ways. Also, there should be transaction execution specification languages that will allow users to specify the cooperation among cooperating transactions.

- Integrity constraints need to be investigated in depth. Time-constrained access, manipulation, and enforcement of (possibly temporal) integrity constraints is an area that needs to be researched.

- Achieving adequate performance in a temporal or real-time DBMS remains a challenge. In particular, in temporal databases, attention should be directed towards empirical studies that compare temporal join algorithms and temporal indexing. In real-time databases, more research is needed on techniques that satisfy both integrity constraints and temporal data and transaction constraints, and provide timely enforcement of the consistency of database views when the underlying database changes.

The development of new models and theories, as outlined above, may provide a sound basis for real-time and temporal databases. We feel that closer interaction of the until-now isolated research communities may yield database technology supporting all applications involving data with a time component.

8 Acknowledgements

The first author was supported in part by NSF grant ISI-9224660. The second author was supported in part by NSF grant IRI-9302244 and by an AT&T Foundation grant. James Clifford was helpful in understanding structural aspects of models of time. Comments on previous drafts by Aziz Ait-Braham, José Blakeley, Curtis Dyreson, Christian S. Jensen, Wolfgang Käfer, Won Kim, Nick Kline, Ellen Rose, Arie Segev, Michael Soo and Stanley Su, and the reviewers, as well as comments from Dan Fishman, Shashi Gadia, Edward Sciore and Michael Stonebraker, helped improve the presentation and identify inaccuracies.

References

[1] Abbott, R., and Garcia-Molina, H., "Scheduling Real-Time Transactions with Disk resident Data," *Proceedings of the International Conference on Very Large Databases*, Aug. 1989.

[2] Abbott, R., and Garcia-Molina, H., "Scheduling I/O Requests with Deadlines: A Performance Evaluation," *11th Real-Time Systems Symposium*, 1990.

[3] Abbott, R., and Garcia-Molina, H., "Scheduling Real-Time Transactions: A Performance Study," *ACM Transactions on Database Systems 17*, 3, pp. 513–560, Sept 1992.

[4] ADB. Matisse Technology Overview. Technical report, ADB/Intellitic, 1992.

[5] Agrawal, D., and Abbadi, A. E., "Locks with Constrained Sharing," in *Proceedings of the ACM PODS Conference*, pp. 85–93, 1990.

[6] Agrawal, D., Abbadi, A. E., and Lang, A. E., "Performance Characteristics of Protocols with Ordered Shared Locks," in *Proceedings of the International Conference on Data Engineering*, pp. 592–601, 1991.

[7] Agrawal D., Abbadi, A. E., and Jeffers, R., "Using Delayed Commitment in Locking Protocols for Real-Time Databases," in *Proceedings of the ACM International Conference on Management of Data*, pp. 104–113, 1992.

[8] Ahn, I., and Snodgrass, R.T., "Performance evaluation of a temporal database management system," in C. Zaniolo, editor, *Proceedings of the ACM International Conference on Management of Data*, pp. 96–107, May 1986.

[9] Ahn, I., and Snodgrass, R.T., "Partitioned storage for temporal databases," *Information Systems 13*, 4, pp. 369–391, 1988.

[10] Ahn, I., and Snodgrass, R.T., "Performance analysis of temporal queries," *Information Sciences 49*, pp. 103–146, 1989.

[11] Ait-Braham, A., Theodoulidis, B., and Karvelis, G., "Conceptual Modelling and Manipulation of Temporal Databases," in *Proceedings of the Entity-Relationship Conference*, 1994.

[12] Al-Taha, K.K., Snodgrass, R.T., and Soo, M.D., "Bibliography on spatiotemporal databasesm" *International Journal of Geographical Information Systems 8*, pp. 195–203, January-February 1994.

[13] Andany, J., Leonard, M., and Palisser, C., "Management of schema evolution in databases," in *Proceedings of the Conference on Very Large Databases*, Barcelona, Spain, pp. 161–170, September 1991.

[14] Ariav, G., "A temporally oriented data model," *ACM Transactions on Database Systems 11*, 4, pp. 499–527, December, 1986.

[15] Barbic, F., and Pernici, B.. "Time modeling in office information systems," In S. Navathe, editor, *Proceedings of the ACM International Conference on Management of Data*, Austin, TX, pp. 51–62, May 1985.

[16] Bancilhon, F., "Object-Oriented Database Systems," *Proceedings of the ACM SIGACT/SIGMOD PODS Conference*, Austin, Texas, pp. 152–162, 1988.

[17] Bassiouni, M.A., and Llewellyn, M.J., "A Relational-Calculus Query Language for Historical Databases," *Journal of Computer Languages 17*, 3, pp. 185–197, 1992.

[18] Baudinet, M., Chomicki, J., and Wolper, P., *Temporal Deductive Databases*, chapter 13, pp. 294–320. Benjamin/Cummings, 1993.

[19] Bayer, R., Elhardt, K., Heigert, J., and Reiser, A., "Dynamic Timestamp Allocation for Transactions in Database Systems," in *Proceedings of the International Symposium on Distributed Databases*, pp. 9-20, 1982.

[20] Beech, D., and Mahbod, B., "Generalized version control in an object-oriented database," in *Proceedings of the International Conference on Very Large Databases*, pp. 14-22, February 1988.

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

[22] Bernstein, P. A., Hadzilacos, V., Goodman, N., *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[23] Bertino, E., "A view mechanism for object-oriented databases," in *Proceeedings of the International Conference on Extending Database Technology*, Vienna, Austria, March 1992.

[24] Bertino E., and Martino, L., *Object-Oriented Database Systems.* International Computer Science Series. Addison-Wesley, 1993.

[25] Bhargava, G., and Gadia, S.K., "Relational database systems with zero information loss," *IEEE Transactions on Knowledge and Data Engineering 5*, 7, pp. 76–87, February 1993.

[26] Biyabani, S., Stankovic, J. A., and Ramamritham, K., "The Integration of Deadline and Criticalness in Hard Real-Time Scheduling," in *Proceedings of the 9th Real-Time Systems Symposium*, pp. 152–160, 1988.

[27] Blakeley, J.A., "ZQL[C++]: Extending a Persistent C++ Language with a Query Capability," Technical Report ITB-91-10-01, Computer Science Laboratory, Texas Instruments Inc., Dallas, TX, October 1991.

[28] Böhlen, M., and Marti, R., "Handling Temporal Knowledge in a Deductive Database System," Datenbanksysteme in Büro, Technik und Wisenschaft, Springer-Verlag, 1993.

[29] Bolour, A., Anderson, T.L., Dekeyser, L.J., and Wong, H.K.T., "The role of time in information processing: A survey," *SigArt Newsletter 80*, pp. 28–48, April 1982.

[30] Buchmann, A.P., McCarthy, D.R., Hsu, M., and Dayal, U., "Time-Critical Database Scheduling: A Framework for Integrating Real-Time Scheduling and Concurrency Control," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 470–480, January 1989.

[31] Cardenas, A., Ieong, I., Taira, R., Barker, R., and Breant, C., "The Knowledge-Based Object-Oriented PICQUERY+ Language," *IEEE Transactions on Knowledge and Data Engineering 5*, 4, pp. 644–657, August 1993.

[32] Carey, M.J., DeWitt, D.J., and Vandenburg, S.L., "A data model and query language for EXODUS," In *Proceedings of the ACM International Conference on Management of Data*, Chicago, IL, pp. 413–423, June 1988.

[33] Carey, M. J., Jauhari, R., and Livny, M., "Priority in DBMS Resource Scheduling," in *Proceedings of the International Conference on Very Large Databases*, 1989.

[34] Caruso, M., and Sciore, E., "Meta-functions and contexts in an object-oriented database language," In *Proceedings of the ACM International Conference on Management of Data*, Chicago, IL, pp. 56–65, June 1988.

[35] Cattell, R.G. (ed.), *The Object Database Standard: ODMG-93,* Morgan Kaufmann, 1994.

[36] Chakravarthy, S. and Kim, S.-K., "Resolution of Time Concepts in Temporal Databases," *Information Sciences 80*, 1–2, pp. 91–125, September 1994.

[37] Chen, S., Stankovic, J. A., Kurose, J. and Townley, D., "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems," *Journal of Real-Time Systems 3*, pp. 307–337, 1991.

[38] Chomicki, J., "Temporal query languages: A survey," in Ohlbach, H. J. Gabbay, D. M., editor, *Proceedings of the First International Conference on Temporal Logic*. Lecture Notes in Artificial Intelligence 827, Springer-Verlag, pp. 506–534, July 1994.

[39] Chomicki, J., and Imelinski, T., "Relational specifications of infinite query answers," in *Proceedings of the ACM International Conference on Management of Data*, pp. 174–183, May 1989.

[40] Chou, H.T., and Kim, W., "A unifying framework for version control in a cad environment," in *Proceedings of the International Conference on Very Large Databases*, pp. 336–344, 1986.

[41] Chu, W. W., Ieong, I. T., Taira, R. K., Breant, C. M., "A Temporal Evolutionary Object-Oriented Data Model and Its Query Language for Medical Image Management," in *Proceedings of the Very Large Database Conference*, August 1992.

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

[43] Clifford, J., Croker, A., and Tuzhilin, A., "On completeness of historical relational query languages," *ACM Transactions on Database Systems 19*, 1, pp. 64–116, March 1994.

[44] Clifford, J., and Tansel, A.U., "On an algebra for historical relational databases: Two views," in S. Navathe, editor, *Proceedings of the ACM International Conference on Management of Data*, pp. 247–265, Austin, TX, May 1985.

[45] Clifford, J., and Warren, D.S., "Formal semantics for time in databases," *ACM Transactions on Database Systems 8*, 2, pp. 214–254, June 1983.

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

[47] Dadam, P., Lum, V., and Werner, H.-D., "Integration of time versions into a relational database system," in U. Dayal, G. Schlageter and L.H. Seng, editors, *Proceedings of the Conference on Very Large Databases*, pp. 509–522, Singapore, August 1984.

[48] Date, C.J., "A proposal for adding date and time support to SQL," *ACM SIGMOD Record 17*, 2, pp. 53–76, June 1988.

[49] Dayal, U., and Smith, J.M., *PROBE: A Knowledge-Oriented Database Management System*. Springer-Verlag, 1986.

[50] Dayal, U., and Wuu, G.T.J., "A uniform approach to processing temporal queries," in *Proceedings of the Conference on Very Large Databases*, Vancouver, Canada, August 1992.

[51] DeAntonellis, V., Degli, A., Mauri, G., and Zonta, B., "Extending the entity-relationship approach to take into account historical aspects of systems," in P. Chen, editor, *Proceedings of the International Conference on the E-R Approach to Systems Analysis and Design*. North Holland, 1979.

[52] Deen, S.M. "Deal: A relational language with deductions, functions and recursions," *Data and Knowledge Engineering 1*, 1985.

[53] Dreyer, W., Dittrich, A.K., Schmidt, D., "An Object-Oriented Data Model for a Time Series Management System," in *Proceedings of the International Working Conference on Scientific and Statistical Database Management*, J.C. French and H. Hinterberger, editors, Charlottesville, VA, pp. 185–95, September, 1994.

[54] Di Pippo, L.B.C., and Wolfe, V.F., "Object-Based Semantic Real-Time Concurrency Control," in *Proceedings of the 14th IEEE Real-Time Systems Symposium*, Dec. 1993.

[55] Dutta, S., "Generalized events in temporal databases," in *Proceedings of the International Conference on Data Engineering*, pp. 118–126, Los Angeles, CA, February 1989.

[56] Dyreson, C.E., and Snodgrass, R.T., "Valid-time Indeterminacy," in *Proceedings of the International Conference on Data Engineering*, pp. 335–343, Vienna, Austria, April 1993.

[57] Dyreson, C.E., and Snodgrass, R.T., "Temporal granularity and indeterminacy: Two sides of the same coin," Technical Report TR 94-06, Computer Science Department, University of Arizona, Tucson, February 1994.

[58] Elmasri, R., and Wuu, G.T.J., "A temporal model and query language for ER databases," in *Proceedings of the International Conference on Data Engineering*, pp. 76–83, February 1990.

[59] Emerson, E.A., "Temporal and Modal Logic," in J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp. 997–1067, Elsevier Science Publishers, B.V., 1990.

[60] Eswaran, K.P., Gray, J.N., Lorie, R.A., and Traiger, I.L., "The Notions of Consistency and Predicate Locks in a database system," *Communications of the ACM 19*, 11, pp. 624–633, 1976.

[61] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., "Fast Subsequence Matching in Time-Series Databases," in Proceedings of the ACM SIGMOD International Conference on Management of Data, R.T. Snodgrass and M. Winslett, editors, pp. 419–429, Minneapolis, MN, May, 1994.

[62] Frank, A.U., "Qualitative Temporal Reasoning is GIS-Ordered Time Scales," Technical Report, Dept. for Geo-Information, Technische Universität Wien, 1994.

[63] Gadia, S.K., "Toward a multihomogeneous model for a temporal database," in *Proceedings of the International Conference on Data Engineering*, pp. 390–397, Los Angeles, CA, February 1986.

[64] Gadia, S.K., "A homogeneous relational model and query languages for temporal databases," *ACM Transactions on Database Systems 13*, 4, pp. 418–448, December 1988.

[65] Gadia, S.K., Nair, S., and Poon, Y.-C.. "Incomplete information in relational temporal databases," in *Proceedings of the Conference on Very Large Databases*, Vancouver, Canada, August 1992.

[66] Gadia, S.K., and Yeung, C.S., "A generalized model for a relational temporal database," in *Proceedings of the ACM International Conference on Management of Data*, pp. 251–259, Chicago, IL, June 1988.

[67] Gehani, N.H., Jagadish, H.V., Shumeli, O., "Event Specification in an Active Object-Oriented Database," in *Proceedings of the ACM SIGMOD Conference*, pp. 81–90, San Diego, CA, June 1992.

[68] Goralwalla, I., and Özsu, M.T., "Temporal Extensions to a Uniform Behavioral Object Model," in *Proceedings of the International Conference on Entity-Relationship Approach*, Dallas, TX, June 1993.

[69] Graham, M. H., "Issues in Real-Time Data Management," *Journal of Real-Time Systems 4*, pp. 185–202, 1992.

[70] Gunadhi, H., and Segev, A., "A Framework for Query Optimization in Temporal Databases", in *Proceedings of the International Conference on Statistical and Scientific Database Management Systems*, volume 420, pp. 131–147, Springer Verlag, April 1990.

[71] Gunadhi, H., and Segev, A., "Query processing algorithms for temporal intersection joins," in *Proceedings of the International Conference on Data Engineering*, Kobe, Japan, 1991.

[72] Haerder, T., and Reuter, A., "Principles of Transaction-Oriented Database Recovery," *ACM Computing Surveys 15*, 4, pp. 287–317, 1983.

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

[74] Haritsa, J. R., Carey, M. J., and Livny, M., "Dynamic Real-Time Optimistic Concurrency Control," in *Proceedings of the 11th Real-Time Systems Symposium*, 1990.

[75] Haritsa, J. R., Livny, M., and Carey, M.J., "Earliest Deadline Scheduling for Real-Time Database Systems," in *Proceedings of the 12th Real-Time Systems Symposium*, 1991.

[76] Haritsa, J. R., Carey, M. J., and Livny, M., "Value-Based Scheduling in Real-Time Database Systems," *VLDB Journal 2*, 2, 1993.

[77] Haritsa, J. R., Carey, M. J., and Livny, M., "Data Access Scheduling in Firm Real-Time Database Systems," *Journal of Real-Time Systems 4*, pp. 203–241, 1992.

[78] Haritsa, J. R., Carey, M. J., and Livny, M., "On Being Optimistic About Real-Time Constraints," in *Proceedings of the ACM PODS Conference*, pp.331–343, 1990.

[79] Haritsa, J. R., "Approximate Analysis of Real-Time database Systems," IEEE, 1994.

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

[81] Heiler, S., Haradhvala, S., Zdonik, S., Blaustein, B., Rosenthal, A., "A Flexible Framework for Transaction Management in Engineering Environments," in *Database Transaction Models for Advanced Applications*, Ed. Elmagarmid, A.K., Morgan Kauffman Pub., 1993.

[82] Held, G.D., Stonebraker, M., and Wong, E., "Ingres–a relational data base management system," In *National Computer Conference*, volume 44, pp. 409–416, Anaheim, CA, May 1975.

[83] Hong, D., Johnson, T., and Chakravarthy, S., "Real-Time Transaction Scheduling: A Cost Conscious Approach," in *Proceedings of the ACM International Conference on Management of Data*, 1993.

[84] Hou, W-C., Özsoyoğlu, G., "Processing Real-Time Aggregate Queries in CASE-DB", *ACM Transactions on Database Systems*, pp. 224–261, June 1993.

[85] Hsu, S., and Snodgrass, R.T., "Optimal block size for set-valued attributes," *Information Processing Letters 45*, 3, pp. 153–158, March 1993.

[86] Huang, J., Stankovic, J., Towsley, D., and Ramamritham K., "Experimental Evaluation of Real-Time Transaction Processing," in *Proceedings of the Real-Time Systems Symposium*, December 1989.

[87] Huang, J., Stankovic, J., Ramamritham, K., and Towsley, D., "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes," in *Proceedings of the International Conference on Very Large Databases*, pp. 35–46, 1991.

[88] Huang, J., Stankovic, J., Ramamritham, K., and Towsley, D., "On Using Priority Inheritance in Real-Time Databases," in *Proceedings of the 12th Real-Time Systems Symposium*, pp. 210–221, 1991.

[89] Huang, J., Stankovic, J., Ramamritham, K., Towsley, D., and Purimetla, B., "Priority Inheritance in Soft Real-Time Databases," *Journal of Real-Time Systems 4*, pp. 243–268, 1992.

[90] Jauhari, R., Carey, M. J., and Livny, M., "Priority-Hints: An Algorithm for Priority-based Buffer management," in *Proceedings of the International Conference on Very Large Databases*, pp. 708–721, 1990.

[91] Jensen, E.D., Douglas, C., Locke, C.D., and Tokuda, H., "A Time-Driven Scheduler for Real-Time Operating Systems," in *Proceedings of the IEEE Real-Time System Symposium*, December 1986.

[92] Jensen, C.S., Mark, L., and Roussopoulos, N., "Incremental implementation model for relational databases with transaction time," *IEEE Transactions on Knowledge and Data Engineering 3*, 4, pp. 461–473, December 1991.

[93] Jensen, C.S., and Mark, L. "Queries on change in an extended relational model," *IEEE Transactions on Knowledge and Data Engineering 4*, 2 April 1992, pp. 192–200.

[94] Jensen, C.S., Mark, L., Roussopoulos, N., and Sellis, T., "Using differential techniques to efficiently support transaction time," *The VLDB Journal 2*, 1, pp. 75–111, January 1993.

[95] Jensen, C.S., Clifford, J., Elmasri, R., Gadia, S.K., Hayes, P., and Jajodia, S. (eds.). "A consensus glossary of temporal database concepts," Technical Report R 93-2035, Department of Mathematics and Computer Science, Institute for Electronic Systems, Denmark, November 1993.

[96] Jensen, C.S., "Vacuuming in TSQL2," TSQL2 Commentary, September 1994, 10 pages.

[97] Jensen, C.S., and Snodgrass, R.T., "The TSQL2 Data Model," TSQL2 Commentary, September 1994, 62 pages.

[98] Jensen, C.S., and Snodgrass, R.T., "Temporal Specialization and Generalization," *IEEE Transactions on Knowledge and Data Engineering 6*, 6, pp. 954–974, December 1994.

[99] Jensen, C. S., Soo, M. D., and Snodgrass, R.T., "Unifying Temporal Models via a Conceptual Model," *Information Systems 19*, 7, pp. 513–547, December 1994.

[100] Jerome, L., "Real-Time Systems Moving to Unix and Sun", *The Sun Observer Journal*, Dec. 1989.

[101] Jones, S., Mason, P, and Stamper, R., "Legol 2.0: A relational specification language for complex rules," *Information Systems 4*, 4, pp. 293–305, November 1979.

[102] Käfer, W., "History and Version Management of Complex Objects (in German)" , Ph.D. thesis, Fachbereich Informatik, University Kaiserslautern, 1992.

[103] Käfer, W., and Schöning, H., "Mapping a Version Model to a Complex-Object Data Model" , in *Proceedings of International Conference on Data Engineering*, pp. 348–357, 1992.

[104] Käfer, W., and Schöning, H., "Realizing a Temporal Complex-Object Data Model", in *Proceedings of the ACM International Conference on Management of Data*, pp. 266–275, 1992.

[105] Kahn, K., and Gorry, G.A., "Mechanizing temporal knowledge," *Artificial Intelligence*, pp. 87–108, 1977.

[106] Kamath, M., and Ramamritham, K., "Performance Characteristics of Epsilon Serializability with Hierarchical Inconsistency Bounds," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 587–594, 1993.

[107] Kao, B., and Garcia-Molina, H., "Deadline Assignment in a Distributed Soft Real-Time System," in *Proceedings of the 13th International Conference on Distributed Computing Systems*, pp. 428–437, 1993.

[108] Kim, W., *Introduction to Object-Oriented Databases.* MIT Press, Cambridge, MA, 1990.

[109] Kim, W., "On object-oriented database technology," Technical report, UniSQL, Inc., Austin, TX, 1993.

[110] Kim, W., and Chou, H.T., "Versions of Shema in OODB," in *Proceedings of the International Conference on Very Large Databases*, pp. 148–159, Long Beach, CA, 1988.

[111] Kim, W., Garza, J.F., Ballou, N., and Woelk, D., "Architecture of the orion next-generation database system," *IEEE Transactions on Knowledge and Data Engineering 2*, 1, pp. 109–124, March 1990.

[112] Kim, S.-K. and Chakravarthy, S., "Temporal Databases with Two-Dimensional Time: Modeling and Implementation of Multihistory," *Information Sciences 80*, 1–2, pp. 43–89, September 1994.

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

[114] Kline, N., "An Update of the Temporal Database Bibliography," *ACM SIGMOD Record 22*, 4, pp. 66–80, December 1993.

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

[116] Korth, H. F., Soparkar, N., and Silberschatz, A., "Triggered Real-Time Databases with Consistency Constraints," in *Proceedings of the International Conference on Very Large Databases*, 1990.

[117] Koubarakis, M., "Representation and querying in temporal databases: the power of temporal constraints," in *Proceedings of the International Conference on Data Engineering*, pp. 327–334, Vienna, Austria, April 1993.

[118] Kuo, T. W., and Mok, A. K., "Application Semantics and Concurrency Control of Real-Time Data-Intensive Applications," in *Proceedings of the Real-Time Systems Symposium*, pp. 35–45, 1992.

[119] Kim, W., and Srivastava, J., "Enhancing Real-Time DBMS Performance with Multiversion Data and Priority Based Disk Scheduling," *in Proceedings of the 12th Real-Time Systems Symposium*, pp. 222–231, 1991.

[120] Kim, Y-K., and Son, S.H., "An Approach Towards Predictable Real-Time Transaction Processing," in *Proceedings of the 5th Euromicro Workshop on Real-Time Systems*, Oulu, Finland, 1993.

[121] Kung, H.T., and Robinson, J.T., "On Optimistic Concurrency Control," *ACM Transactions on Database Systems 6*, 2, pp.213–226, 1981.

[122] Ladkin, P., *The Logic of Time Representation.* Ph.D. thesis, University of California, Berkeley, November 1987.

[123] Lamport, L., and Lynch, N., "Distributed Computing: Models and Methods," in J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp. 1159–1196, Elsevier Science Publishers, B.V., 1990.

[124] Lee, J., and Son, S.H., "Using Dynamic Adjustment of Serialization Order for Real-Time Database Systems," in *Proceedings of the 14th IEEE Real-Time Systems Symposium*, Raleigh-Durham, North Carolina, December 1993.

[125] Lee, J., and Son, S.H., "Issues in Developing Object-Oriented Database Systems for Real-Time Applications," in *Proceedings of the IEEE Workshop on Real-Time Applications*, Washington, D.C., 1994.

[126] Lee, J., and Son, S. H., "Performance of Concurrency Control Algorithms for Real-Time Database Systems," a chapter in *Performance of Concurrency Control Mechanism in Centralized Database Systems*, V. Kumar (ed), Prentice-Hall, 1995 (to appear).

[127] Lee, R.M., Coelho, H., and Cotta, J.C., "Temporal inferencing on administrative databases," *Information Systems 10*, 2, pp. 197–206, 1985.

[128] Leung T.Y., and Muntz, R., "Query processing for temporal databases," in *Proceedings of the International Conference on Data Engineering*, Los Angeles, California, February 1990.

[129] Leung, T.Y.C., and Muntz, R.R., "Stream Processing: Temporal Query Processing and Optimization," Chapter 14, *Temporal Databases: Theory, Design, and Implementation*, A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.). Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, pp. 329–355, 1993.

[130] Lin, K. J., Natarajan, S., and Liu, W. S., "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," in *Proceedings of the Real-Time Systems Symposium*, pp. 210–217, 1987.

[131] Lin, K. J., "Designing Databases in Real-Time Embedded Systems," in *Proceedings of the IEEE Symposium on Computer-Aided Control System Design*, pp. 285–292, March 1992.

[132] Lin, K. J., "Consistency Issues in Real-Time Database Systems," in *Proceedings of the 22nd Hawaii International Conference System Sciences*, Hawaii, January 1989.

[133] Lin, Y., and Son, S. H., "Concurrent Control in Real-Time Databases by Dynamic Adjustment of Serialization Order," in *Proceedings of the 11th IEEE Real-Time Systems Symposium*, Orlando, Florida, December 1990.

[134] Liu, C. L., and Layland, J. W., "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," *JACM 20*, 1, pp. 46–61, 1973.

[135] Liu, J., Lin, K., Shih, W., Yu, A., Chung, J., and Zhao, W., "Algorithms for Scheduling Imprecise Computation," *IEEE Computer 24*, 5, May 1991.

[136] Liu, J., Lin, K., Song, X, "A Position Paper on Scheduling Hard Real-Time Transactions" *Proc. 1988 Workshop on Real-Time Operating Systems and Software*, pp. 12-13, May 1988.

[137] Lomet, D., and Salzberg, B., "Access methods for multiversion data," in *Proceedings of the ACM International Conference on Management of Data*, pp. 315–324, June 1989.

[138] Lomet, D., and Salzberg, B., "Transaction-Time Databases," Chapter 16, *Temporal Databases: Theory, Design, and Implementation*, A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.). Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, pp. 388–417, 1993.

[139] Lorentzos, N.A., and Johnson, R.G., "Extending relational algebra to manipulate temporal data," *Information Systems 13*, 3, pp. 289–296, 1988.

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

[141] Lyngbæk, P. "OSQL: A Language for Object Databases," Technical Report HPL-DTD-91-4, Hewlett-Packard Laboratories, Palo Alto, CA, January 1991.

[142] McKenzie, M., "Bibliography: Temporal Databases," *ACM SIGMOD Record 15*, 4, pp. 40–52, December 1986.

[143] McKenzie, E., and Snodgrass, R.T., "Schema Evolution and the Relational Algebra," *Information Systems 15*, 2, pp. 207–232, June 1990.

[144] McKenzie, E., and Snodgrass, R., "An Evaluation of Relational Algebras Incorporating the Time Dimension in Databases," *ACM Computing Surveys 23*, 4, pp. 501–543, December 1991.

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

[146] Melton, J., and Simon, A.R., *Understanding the New SQL: A Complete Guide.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.

[147] Montague, R., *The proper treatment of quantification in ordinary English.* D. Reidel Publishing Co., Dordrecht, Holland, 1973.

[148] Montanari, A. and Pernici, B., *Temporal Reasoning*, chapter 21, pp. 534–562. , *Temporal Databases: Theory, Design, and Implementation*, A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.). Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, 1994.

[149] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M., "Telos: Representing Knowledge about Information Systems," *ACM Transactions on Office Information Systems 8*, 4, pp. 325–362, October 1990.

[150] Navathe, S.B. and Ahmed, R., "A temporal relational model and a query language," *Information Sciences 49*, pp. 147–175, 1989.

[151] Nodine, M.H., Ramaswamy, S., Zdonik, S.B., "A Cooperative Transaction Model for Design Databases," in *Database Transaction Models for Advanced Applications*, Ed. Elmagarmid, A.K., Morgan Kauffman Pub., 1993.

[152] OMG and Xopen. *The Common Object Request Broker: Architecture and Specification.* Object Management Group and X/Open, Framingham, MA and Reading Berkshire, UK, 1992.

[153] Overmyer, R., and Stonebraker, M., "Implementation of a time expert in a database system," *ACM SIGMOD Record 12*, 3, pp. 51–59, April 1982.

[154] Özsoyoğlu, G., and Hafez, A., "Near-Optimum Storage Models for Nested Relations Based on Workload Information," *IEEE Transactions for Knowledge and Data Engineering*, , Vol. 5, No. 6, pp. 1018-1038, December 1993.

[155] Özsoyoğlu, G., Guruswamy, S., Du, K., Hou, W-C., "Time-Constrained Query Processing in CASE-DB," *IEEE Transactions for Knowledge and Data Engineering*, (1995) (to appear).

[156] Özsu, M.T., Peters, R., Szafron, D., Irani, B., Lipka, A., and Muñoz, "TIGUKAT: A Uniform Behavioral Objectbase Management System," *VLDB Journal*, (1995) (to appear).

[157] Plouffe, W., Kim, W., Lorie, R., and McNabb, D., "Versions in an engineering database system," IBM Research Report RJ4085, San Jose, CA, October 1983.

[158] Pu, C., and Leff, A., "Replica Control in Distributed Systems: An Asynchronous Approach," in *Proceedings of the ACM International Conference on Management of Data*, pp. 377–386, 1991.

[159] Ramamritham, K., "Real-Time Databases," *International Journal of Distributed and Parallel Databases 1*, 2, pp. 199–206, 1993.

[160] Rescher, N.C., and Urquhart, A., *Temporal Logic*, Springer-Verlag, 1971.

[161] Roddick, J.F., "Schema evolution in database systems — an annotated bibliography," *ACM SIGMOD Record 21*, 4, pp. 35–40, December1992.

[162] Roddick, J.F., and Snodgrass, R.T., "Schema Versioning Support in TSQL2," TSQL2 Commentary, September 1994, 19 pages.

[163] Rose, E., and Segev, A., "TOODM — A Temporal Object-oriented Data Model with Temporal Constraints," In *Proceedings of the International Conference on the Entity Relationship Approach*, October 1991.

[164] Rose, E., and Segev, A., "TOOA: A Temporal Object-oriented Algebra," In *Proceedings of the European Conference on Object-Oriented Programming*, July 1993.

[165] Rose, E., and Segev, A., "TOOSQL – a temporal object-oriented query language," In *Proceedings of the International Conference on the Entity-Relationship Approach*, Dallas, TX, 1993.

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

[167] Sadeghi, R., Samson, W.B., and Deen, S.M., "HQL — A Historical Query Language," Technical report, Dundee College of Technology, Dundee, Scotland, September 1987.

[168] Slazberg, B., and V.J. Tsotras, "A Comparison of Access Methods for Time Evolving Data," Technical Report CATT-TR-94-81, Polytechnic University, 1994.

[169] Sarda, N., "Algebra and query language for a historical data model," *The Computer Journal 33*, 1, pp. 11–18, February 1990.

[170] Sarda, N., "Extensions to sql for historical databases," *IEEE Transactions on Knowledge and Data Engineering 2*, 2, pp. 220–230, June 1990.

[171] Satoh, K., Tsuchida, M., Nakamura, F., and Oomachi, K., "Local and global query optimization mechanisms for relational databases," In A. Pirotte and Y. Vassiliou, editors, *Proceedings of the International Conference on Very Large Databases*, pp. 405–417, Stockholm, Sweden, August 1985.

[172] Sciore, E., "Using annotations to support multiple kinds of versioning in an object-oriented database system," *ACM Transactions on Database Systems 16*, 3, pp. 417–438, September 1991.

[173] Sciore, E., "Versioning and configuration management in an object-oriented data model," *VLDB Journal* (to appear), 1994.

[174] Segev, A., Himawan, G., Chandra, R., and Shanthikumar, J., "Selectivity estimation of temporal data manipulations," *Information Sciences 74*, 1-2, October 1993.

[175] Segev, A., and Shoshani, A., "Logical modeling of temporal data," In U. Dayal and I. Traiger, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 454–466, San Francisco, CA, May 1987.

[176] Segev, A., and Chandra, R., "A Data Model for Time-Series Analysis," *Advanced Database Systems*, N.R. Adam and B.K. Bhargava, editors, Springer-Verlag, 1993, pp. 191–212.

[177] Sellis, T.K., "Global query optimization," In C. Zaniolo, editor, *Proceedings of the ACM International Conference on Management of Data*, pp. 191–205, Washington, DC, May 1986.

[178] Seshadri, P., Livny, M., and Ramakrishnan, R., "Sequence Query Processing," in Proceedings of the ACM SIGMOD International Conference on Management of Data, R.T. Snodgrass and M. Winslett, editors, pp. 430–441, Minneapolis, MN, May, 1994.

[179] Sha, L., Rajkumar, R., and Lehoczky, J., "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," *IEEE Transactions on Computers 39*, pp. 1175–1185, 1990.

[180] Sha, L., Rajkumar, R., Son, S. H., and Chang, C., "A Real-Time Locking Protocol," *IEEE Transactions on Computers 40*, 7, pp. 782–800, July 1991.

[181] Shahar, Y., Tu, S.W., Musen, M.A., "Knowledge Acquisition for Temporal-Abstraction Mechanisms," *Knowedge Acquisition 4*, 2, pp. 217-236, June 1992.

[182] Shahar, Y., Musen, M.A., "RESUME: A Temporal-Abstraction System for Patient Monitoring," *Computers and Biomedical Research 26*, 3, pp. 255–273, June 1993.

[183] Shenoy, S., and Özsoyoğlu, Z., "Design and Implementation of a Semantic Query Optimizer" , *IEEE Transactions on Data and Knowledge Engineering 1*, 3, pp. 344–361, September 1989.

[184] Smith, K. P., and Liu, J. W. S., "Monotonically Improving Approximate Answers to Relational Algebra Queries," in *Proceedings of the COMPSAC 89 Conference*, Orlando, Florida, September 1989.

[185] Snodgrass, R.T., and Ahn, I., "Temporal Databases," *IEEE Computer 19*, 9, pp. 35–42, September 1986.

[186] Snodgrass, R.T., "The Temporal Query Language TQuel," *ACM Transactions on Database Systems 12*, 2, pp. 247–298, June 1987.

[187] Snodgrass, R.T., "Temporal Databases," *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, A. U. Frank, I. Campari, and U. Formentini (eds.), Springer-Verlag Lecture Notes in Computer Science 639, pp. 22–64, September 1992.

[188] Snodgrass, R.T., Ahn, I., Ariav, G., Batory, D.S., Clifford, J., Dyreson, C.E., Elmasri, R., Grandi, F., Jensen, C.S., Käfer, W., Kline, N., Kulkanri, K., Leung, C.Y.T., Lorentzos, N., Roddick, J.F., Segev, A., Soo, M.D., and Sripada, S.M., "TSQL2 Language Specification," *ACM SIGMOD Record 23*, 1, pp. 65–86, March 1994.

[189] Snodgrass, R.T., "Temporal Object-Oriented Databases: A Critical Comparison," chapter 19 in *Modern Database Systems: The Object Model, Interoperability and Beyond*, W. Kim, editor, Addison-Wesley/ACM Press, 1995, pp. 386–408.

[190] Son, S. H., editor, "Real-Time Database Systems: Issues and Approaches," *ACM SIGMOD Record 17*, 1, Special Issue on Real-Time Database Systems, March 1988.

[191] Son, S. H., Lee, J., and Kang, H., "Approaches to Design of Real-Time Database Systems," in *Proceedings of the Symposium on Database Systems for Advanced Applications*, Korea, pp. 274–281, April 1989.

[192] Son, S.H., and Kouloumbis, S., "Replication Control for Distributed Real-Time Database Systems," in *Proceedings of the 12th Distr. Computing System Conference*, 1992.

[193] Son, S. H., Park, S., and Lin, Y., "An Integrated Real-Time Locking Protocol," in *Proceedings of the International Conference on Data Engineering*, pp. 527–534, 1992.

[194] Son, S. H., and Lee, J., "Scheduling Real-Time Transactions in Distributed Database Systems," in *Proceedings of the 7th IEEE Workshop on Real-Time Operating Systems and Software*, Charlottesville,Virginia, pp. 39–43,May 1990.

[195] Son, S. H., Lee, J., and Lin, Y., "Hybrid Protocols using Dynamic Adjustment of Serialization Order for Real-Time Concurrency Control," *Real-Time Systems Journal*, pp. 269–276, September 1992.

[196] Song, X., and Liu, J., "How Well can Data Temporal Consistency be Maintained?," in *Proceedings of the IEEE Symposium on Computer-Aided Control System Design*, pp. 276–284, March 1992.

[197] Soo, M.D., "Bibliography on temporal databases," *ACM SIGMOD Record 20*, 1, pp. 14–23, March 1991.

[198] Soo, M.D., Jensen, C.S., and Snodgrass, R.T., "An Algebra for TSQL2," TSQL2 Commentary, September 1994, 36 pages.

[199] Soo, M.D., Snodgrass, R.T., and Jensen, C.S., "Efficient evaluation of the valid-time natural join," in *Proceedings of the International Conference on Data Engineering*, pp. 282–292, February 1994.

[200] Sripada, S., "A logical framework for temporal deductive databases," In *Proceedings of the International Conference on Very Large Databases*, pp. 171–182, Los Angeles, CA, 1988.

[201] Stam, R., and Snodgrass, R.T., "A Bibliography on Temporal Databases," *IEEE Database Engineering 7*, 4, pp. 231–239, December 1988.

[202] Stefik, M., Bobrow, D., and Kahn, K., "Integrating Access-oriented Programming into a Multiparadigm Environment," *IEEE Software 3*, 1, pp. 10–18, January 1986.

[203] Stankovic, J., "Misconceptions about Real-Time Computing," *IEEE Computer 21*, 10, pp. 10–19, October 1988.

[204] Stonebraker, M., "The Design of the Postgres Storage System," In P. Hammersley, editor, *Proceedings of the International Conference on Very Large Databases*, pp. 289–300, Brighton, England, September 1987.

[205] Stonebraker, M., Rowe, L., and Hirohama, M., "The Implementation of Postgres," *IEEE Transactions on Knowledge and Data Engineering 2*, 1, pp. 125–142, March 1990.

[206] Su, S.Y.W., and Chen, H.M., "A Temporal Knowledge Representation Model OSAM*/T and Its Query Language OQL/T," In *Proceedings of the International Conference on Very Large Databases*, 1991.

[207] Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A., and Snodgrass, R.T. (eds.). *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, 1994.

[208] Tansel, A.U., "Adding time dimension to relational model and extending relational algebra," *Information Systems 11*, 4, pp. 343–355, 1986.

[209] Tansel, A.U., Arkun, M.E., and Özsoyoğlu, G., "Time-By-Example Query Language for Historical Databases," *IEEE Transactions on Software Engineering 15* 4, pp. 464–478, April 1989.

[210] Tansel, A.U., "A Historical Query Language," *Information Sciences 53*, pp. 101–133, 1991.

[211] Thompson, P.A., *A Temporal Data Model Based on Accounting Principles.* Ph.D. thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, March 1991.

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

[213] Ulusoy, O., "Research Issues in Real-Time Database Systems," Tech. Report BU-CEIS-9432, Bilkent University, Ankara, Turkey 1994.

[214] Ulusoy, O., "A Study of Two Transaction Processing Architectures for Distributed Real-Time Database Systems," *The Journal of Systems and Software* (to appear), 1995.

[215] Ulusoy, O., and Belford, G. G., "Real-Time Transaction Scheduling in Database Systems," *Information Systems Journal 18*, 8, 1993.

[216] Ulusoy, O., and Belford, G. G., "Real-Time Lock Based Concurrency Control in a Distributed Database System," in *Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 136–143, 1992.

[217] Urban, S.D., and Delcambre, L.M.L., "An analysis of the structural, dynamic, and temporal aspects of semantic data models," in *Proceedings of the International Conference on Data Engineering*, volume IEEE Computer Society Order Number 655, pp. 382–389, Los Angeles, CA, February 1986. IEEE Computer Society, IEEE Computer Society Press.

[218] Van Benthem, J.F.K.A., *The Logic of Time: A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse.* Reidel, Hingham, MA, 1982.

[219] Vrbsky, S., and Lin, K. J., "Recovering Imprecise Transaction with Real-Time Constraints," in *Proceedings of the 7th Symo. on Reliable Distributed Systems*, pp. 185–193, Oct. 1988.

[220] Vrbsky, S. V., Liu, J. W. S., and Smith, K. P., "An Object-Oriented Query Processor That Produces Monotonically Improving Approximate Answers," in *Proceedings of the IEEE Conference on Data Engineering*, Kobe, Japan, 1991.

[221] Wang, X., Jajodia, S., and Subrahamanian, V., "Temporal Modules: An Approach Toward Temporal Databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data,* R.T. Snodgrass and M. Winslett, editors, Minneapolis, MN, pp. 227–236, May 1994.

[222] Wiederhold, G., Fries, J.F., and Weyl, S., "Structured organization of clinical data bases," in *Proceedings of the National Computing Conference*, pp. 479–485, 1975.

[223] Wiederhold, G., Jajodia, S., and Litwin, W., "Dealing with granularity of time in temporal databases," in *Proc. 3rd Nordic Conference on Advanced Information Systems Engineering*, Trondheim, Norway, May 1991.

[224] Wilkinson, K., Lyngbæk, P., and Hasan, W., "The IRIS Architecture and Implementation," *IEEE Transactions on Knowledge and Data Engineering 2*, 1, pp. 63–75, March 1990.

[225] Worboys, M.F., "Reasoning about GIS using temporal and dynamic logics," in *Temporal GIS Workshop.* University of Maine, October 1990.

[226] Wu, K. L., Yu, P. S., and Pu, C., "Divergence Control for Epsilon Serializability," in *Proceedings of the International Conference on Data Engineering*, pp. 2–11, 1992.

[227] Wuu, G.T.J., "SERQL: An ER Query Language Supporting Temporal Data Retrieval," in *Proceedings of the 10th International Phoenix Conference on Computers and Communications*, pp. 272–279, March 1991.

[228] Wuu, G.T.J., and Dayal, U., "A uniform model for temporal object-oriented databases," In *Proceedings of the International Conference on Data Engineering*, pp. 584–593, Tempe, Arizona, February 1992.

[229] Yau, C., and Chat, G. S. W., "TempSQL – A Language Interface to a Temporal Relational Model," *Information Sc. & Tech.*, pp. 44–60, October, 1991.

[230] Yu, P.S., Heiss, H.-U., Dias, D. M., "Modeling and Analysis of a Time-Stamp History Based Certification Protocol for Concurrency Control," *IEEE Transactions on Knowledge Engineering 3*, 4, pp. 525–537, December 1991.

[231] Yu, P.S., Wu, K-L., Lin, K-J., Son, S. H., "On Real-Time Databases: Concurrency Control and Scheduling," *Proceedings of the IEEE 82*, 1, 1994.

[232] Zloof, M., "Query By Example," in *Proceedings of the National Computer Conference*, volume 44, 1975.

Contents

List of Tables