Temporal and Real-Time Databases: A Survey

Gultekin Özsoyoğlu and 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 previously separate areas. First we characterize the time domain; then we investigate temporal and real-time data models. We evaluate temporal and real-time query languages along several dimensions. We examine temporal and real-time DBMS implementation. Finally, we summarize major research accomplishments to date and list several unanswered research questions.**

***Index Terms*—Object-oriented databases, relational databases, query languages, temporal data models, time-constrained databases, transaction time, user-defined time, valid time.**

I. 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, multimedia, 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database change as new information is added, these changes are viewed as modifications to the state, deleting the old, out-of-date data from the database.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 their order of arrival and provide no guarantees of query or transaction completion times.

In this paper we survey two database research areas that may benefit from cross infusion: 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 real-time database research may benefit directly from utilizing temporal data models in real-time transaction and query specification and management, thereby providing better temporal data semantics and better querying capabilities. Temporal database research may benefit indirectly from the use and extension of temporal data models into another area—namely, real-time databases. We attempt here to capture and summarize the major concepts, approaches, and implementation strategies generated by these two research areas. We use a common terminology to emphasize concepts independently generated. Wherever possible, we list research directions relating the two areas.

Manuscript received Mar. 29, 1995.

G. Özsoyoğlu is with the Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH 44106; e-mail: tekin@alpha.ces.cwru.edu.

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

IEEECS Log Number K95045.

A temporal database is one that supports some aspect of time [95]. We will shortly discuss 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 in which transactions have deadlines or timing constraints. Real-time databases are commonly used in 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 are perhaps better viewed as *time-constrained* databases. In this survey we define a real-time database as one that has timing constraints in every operational aspect, such as responding to queries; processing transactions; processing database insertions, deletions, and updates; and maintaining database integrity via integrity constraint enforcement and view management. Although current literature on real-time database research discusses temporal data and "temporal consistency constraints," it does not utilize temporal data models and temporal query languages. 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] lists 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.

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

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

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

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

In the following sections we review the substantial research on temporal and real-time databases. Section II examines 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 III examines the many temporal and real-time data models proposed in the literature.

We consider languages for expressing temporal queries in Section IV. 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 V, which presents the characteristics of real-time data and transactions and new types of data and transaction consistency requirements unique to real-time databases.

The topic of Section VI is real-time and temporal DBMS implementation. We examine the impact on each DBMS component of adding real-time and temporal support and discuss query processing and transaction processing in some detail.

In Section VII we summarize the major accomplishments of research into real-time and temporal databases and conclude by pointing to future work.

II. The Time Domain

In this section we focus on time itself. The next section will combine time with facts to model time-varying information.

A. Structure

We initially assume that there is one dimension of time. The distinctions we address here will apply to each of 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 past to 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, we can specify linear time 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; that is, they are 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 "Big Bang" cosmology posits that time began 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, time will end when the universe collapses back onto itself in what is called the "Big Crunch." If it is open, 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, "9 A.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; for example, one could envision a relative time of –9 hours, whereas a distance is unsigned.

B. Dimensionality

In the context of databases, two time dimensions are of general interest [185]. *Valid time* denotes the time a fact was true in reality. An event's valid time is the time the event occurred in the real world, independent of its recording in some database. Valid time can also be in the future, if it is expected that a fact will be true at a specified future time. *Transaction time* is the time during which the fact was present in the database as stored data. A fact's transaction time (an interval) identifies the transaction that inserted the fact into the database and the transaction that removed the fact from the database.

These two dimensions are orthogonal. A data model supporting neither is termed a *snapshot*, as it captures only a single snapshot in time of both the database and the enterprise modeled by the database. A data model supporting only valid time is termed a *valid-time model*; one that supports only transaction time is termed a *transaction-time model*; 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 of 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, called *degenerate* [98], valid and transaction times 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, and 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.

A third kind of time may be included in the time domain: *user-defined time*. This term indicates that the semantics of these values are known only to the user and are not interpreted by the DBMS, in contrast to valid and transaction times, 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 time the promotion was determined. This time may be different from the valid time of the new position and from the transaction time when the new position was recorded. Decision time is an instant; valid and transaction times generally are intervals. Decision time may be recorded as a valid time for a separate table (e.g., a promotion table, as distinct from a rank table). Whether the increased complexity of including decision time in the data model is justified by the increased expressive power is still an open issue [112].

C. 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 when (i.e., during which chronon) it occurred is known. A determinate event cannot overlap two chronons. If when an event occurred is unknown, but that it did occur is known, the event is temporally indeterminate. The indeterminacy refers to the *time* the event occurred, not *whether* the event occurred. Temporal indeterminacy occurs only in valid time. The granularity of a transaction time timestamp is the smallest intertransaction time. Transaction times are always determinate since the chronon during which a transaction takes place is always known.

D. 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 we survey here. 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, such an event's occurrence time 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 other data items' values. Times associated with these derived values generally correspond to transaction-commit times.

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

Presently, real-time database research does not deal with temporal indeterminacy. However, temporally indeterminate information is encountered frequently in real-time systems as unknown or imprecise event times. For example, a furnace's temperature changes, but we are not sure exactly when; this change and its time must be stored in the database and made available to real-time operations such as queries and transactions. Clearly, real-time transaction processing should 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 times in that the transaction-commit time must be before the specified valid time.

For some real-time systems, called *hard real-time systems*, missing a deadline has serious implications and should not happen. But the values of data items in the database, however recent, may be incorrect because the reality being modeled changed immediately after the value was recorded [213]. 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 behavior for all database operations. However, one can find real-time applications [100] that need timeliness in some, if not all, database operations. For example, stock market, air traffic control, and on-board airplane databases all require timely database updates and transaction completions, while factory-floor databases commonly require timely query and transaction completions but not necessarily timely database updates.

III. Data Models

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.

A. 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 data models. For this reason, we focus on these two data models in this 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). The tuple calculus-based language 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].

Time support in conventional database system implementations are limited in scope and are, in general, unsystematic in their design [48]. SQL-92 corrects some inconsistencies in the time support provided by the commercial database system 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 to consolidate approaches to temporal data models and calculus-based query languages has just been completed, achieving a consensus extension to SQL-92 and an associated data model upon which future research can be based. This extension is called 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 toward this goal.

Table I lists most of the temporal relational data models that we are aware of in the literature.[1] We omit Gadia's multihomogeneous model [63], which is a precursor to his heterogeneous model (Gadia-2). We also omit 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.

TABLE I
Temporal Relational Data Models

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	Bassiouni
—	[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

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

1. Many models are described in several papers; the one referenced is the first journal paper that defined the model.

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

Table II classifies the extant object-oriented temporal data models. We further discuss its fourth column in Section III.A.2. Models with "arbitrary" 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.

TABLE II
TEMPORAL OBJECT-ORIENTED 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

A.1. Valid Time

We can compare temporal data models along the valid-time dimension by asking three basic questions: How is valid time represented? How are facts associated with valid time? How are attribute values represented? Here we list some of the many answers to these questions.

Valid time can be represented with single-chronon identifiers (event timestamps), with intervals (as interval timestamps), 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 an entire tuple or object. Another 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 III categorizes most of the data models in terms of valid time representation. We do not include the OODAPLEX, Sciore-1, and TIGUKAT data models because valid time is arbitrarily specifiable in these models.

TABLE III
REPRESENTATION OF VALID TIME

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

A.2. Transaction Time

The same general issues are involved in transaction time, but there are about three times as many alternatives, partly because transaction time is often used to support versioning, which generally implies an object-oriented data model. Table IV characterizes the choices made in the various data models. OODAPLEX is not included, as it can support virtually any of these options (that is also possible in TIGUKAT, but specific support for versioning has been added to the data model and language). More detail on the representation "Other" appears in the fourth column of Table II. Specifically, 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.

A.3. A Suite of Temporal Data Models

A temporal data model should simultaneously satisfy many goals. It should clearly and concisely capture the semantics of the application to be modeled. It should be a consistent, minimal extension of an existing data model, such as the relational model. The best temporal data model presents all the time-varying behavior of a fact or object coherently. The data model should be easy to implement, while attaining high performance. The experience of the past 15 years and some 40 data models appearing in the literature demonstrate that designing a temporal data model with all 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, employed for implementation, ensures high performance. Other, *presentational* data models may be used to render time-varying behavior to the user or application. Thus, a coordinated suite of data models can achieve goals that no single data model can.

TABLE IV
REPRESENTATION OF TRANSACTION 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

B. Real-time Data Models

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

The relational model is used as a data model for real-time databases [190]; but researchers have not extended it with the semantics of time. There are, however, two proposed real-time query-processing approaches that modify the relational model: 1) the use of sets of *approximate relations* [184], [220], defined for any relation in the relational model for timely evaluation of queries, and revised iteratively for a better approximation and a better query response; and 2) the use of *fragmentation lattices* of relations [84], [155], also for query-processing. Neither approach adds a temporal dimension to the data model.

Recently, there has been some initial research into adapting the object-oriented data model for real-time databases, both because one can utilize its rich data semantics in real-time transaction processing [54] and because complex real-time applications may need its advantages [125], such as complex objects, encapsulation, methods, and messages. Significant research is needed to provide these features in real-time databases requiring timeliness and meeting of time guarantees. For example, the storage of complex objects and the implementation of complex-object algebra operators [154] carry high time costs that must 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 of objects in object-oriented databases brings a number of advantages [16]. However, encapsulation also forces users to access objects in restricted ways, possibly delaying timely execution of transactions. Clearly, the balance between providing new data-modeling features and ensuring timeliness in database operations should be further investigated.

Di Pippo and Wolfe propose a real-time object-oriented database that supports a rich variety of data semantics and 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 transaction output by allowing concurrent, but not serializable, executions of transactions. The idea is to let time-constrained transactions compete with incorrect data as long as the introduced error (i.e., the "imprecision") is bounded. Lee and Son propose a simple real-time object-oriented database system with atomic objects and a class manager [125]. Both 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.

IV. TEMPORAL QUERY LANGUAGES

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 the structure of and constraints on the objects of temporal relational databases, the temporal relations. In this section we complete the picture by discussing the operations, specifically temporal query languages.

Table V lists the major temporal relational query language proposals to date. The "Underlying Data Model" column refers to Table I. The next column lists the conventional query language on which the temporal proposal is based, from the following: DEAL, an extension of the relational algebra incorporating functions, recursion, and deduction [52]; IL_s, an intensional logic formulated in computational linguistics [147]; QBE, Query-by-Example, a domain calculus-based query language [232]; Quel, a tuple calculus-based query language, originally defined for the Ingres relational DBMS [82]; relational algebra, a procedural language with relations as objects [46]; and SQL, a tuple calculus-based language, the lingua franca of conventional relational 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 V
TEMPORAL RELATIONAL QUERY LANGUAGES

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
Temp SQL	[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 VI 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 object-oriented.

The data model and conventional query language on which the temporal query language is based are identified in the third and fourth columns. The fifth column indicates whether the language has been implemented. A few proposals provide algebras for their query languages. It is rare for an object-oriented query language to have a formal semantics.

TABLE VI
TEMPORAL OBJECT-ORIENTED QUERY LANGUAGES

Name	Citation	Underlying Data Model	Based On	Implemented	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	√	
PIC-QUERY+	[31]	TEDM	PIC-QUERY	√	
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	metafunctions	√	
—	[172]	Sciore-1	annotations		
—	[173]	Sciore-2	EXTRA/EXCESS		[32]

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], which 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, although supporting 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 the query language.

A. 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 and Stonebraker 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 (see Section II) 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 object-oriented data model's power (with subtyping, inheritance, and polymorphic functions with late binding) can be used to express the semantics of time appropriate to the application.

B. Valid Time

Valid-time support can have a much greater impact than user-defined-time support. There are three general approaches to adding valid-time support to a data model and query language.

The first approach utilizes the substantial expressive power of the relational or object-oriented data model directly and thus requires no changes to either the model or the query language to support time-varying information. Lorentzos and Johnson's temporal relational algebra is the defining example in the relational domain [139]. Among 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 the user can specify various valid-time semantics (as is also true of user-defined time), and the schema specification and query languages remain uncluttered by additional time-specific clauses—assuming, of course, that the necessary features, including functions and sets as first-class citizens, are available. Users are required to "roll their own" time support when specifying the schema and queries. This approach renders query optimization much more difficult, as the language provides no hints that access methods or storage structures oriented toward 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 can 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, since all time manipulation 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 nontemporal query language on which they are based, ensuring 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 and 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 has 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 and timestamp pair sequences as attribute values.

C. Transaction Time

Transaction time indicates when facts were logically present in the database, as opposed to when facts were true in reality. Transaction time support 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, in contrast to the linear time model underlying valid time.

In considering transaction time support, 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.

C.1. Extension Versioning

As with valid-time support, 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 time semantics is arbitrary (the user can implement whatever is desired), transaction time can be accommodated.

In the second approach, general data model and query language extensions 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 the metafunctions in VISION [34].

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

Chou and Kim's 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 it has not yet been incorporated in 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 merging to create a new version; hence, transaction time is truly a graph, rather than a tree.

C.2. Schema Versioning

In schema *evolution*, the schema can change in response to the application's varying needs. In schema *versioning*, multiple schemas are logically in effect. Schema versioning has been examined in both relational databases [21], [143], [161], [162] and object-oriented databases. The Chou-Kim model accommodates the most extensive form of schema versioning of object-oriented data models. Schema versioning under the Chou-Kim model has not yet been implemented, though Chou and Kim have presented detailed data structures, storage representations, and object-accessing algorithms [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 visible only in that schema version, rather than in all views or contexts [24].

We now turn to data models and query languages supporting real-time applications.

V. Real-Time Data and Transaction Properties

Real-time transactions must be timely—they must be scheduled so as to complete within their time constraints (deadlines) and satisfy transaction constraints. Timely execution of transactions requires good estimates of their worst-case execution times, which are very difficult to obtain when 1) transactions' execution times depend on the values of data items they access (e.g., the range of a looping construct in a transaction is defined through the value of a data item), 2) transactions block or abort due to data and resource conflicts, and 3) I/O scheduling or buffer management techniques 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* may miss their deadlines and still have some monotonically decreasing value assignments that drop to zero at some point P in time. For *firm transactions*, P and the deadline are the same.

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

1) The implication of missing a specified transaction deadline: *hard*, *soft*, or *firm* transactions.
2) Transaction arrival pattern: *periodic*, *sporadic*, or *aperiodic*.
3) Data access type: *Random* (i.e., unknown) or *predefined* with a) write-only (update) transactions that collect information about the state of the real world and write into the database, or b) read-only transactions that read the values of data items and modify the state of the real world through specialized output devices.
4) 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*.
5) Knowledge of items to be used: Whether the accessed items are known a priori.
6) CPU and I/O time knowledge: 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) data model, satisfy integrity constraints, and permit only serializable transaction executions [22]. Such databases are said to be *internally consistent* [132]. Clearly, for these applications, the database should maintain the traditionally accepted transaction correctness properties known as the ACID properties (atomicity, consistency, isolation, and durability) [72].

For some real-time databases, an externally consistent database may be more important than 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) also changes. External consistency requires that the data used by a transaction always reflect the physical environment at the time. In comparison, internal consistency requires that the data in the database satisfy predefined constraints. In applications that need a timely response to external state changes (e.g., autopilot systems, automated factories with robots), external consistency often takes precedence over internal consistency. Thus, for some applications, executing transactions that maintain external consistency takes precedence over having serializable transactions or satisfying the database's integrity constraints 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 them or roll them back. But such an approach 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, 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. Finally, integrity constraint violations may need to be resolved by new transactions. The interplay between external and internal consistency is an unexplored research area.

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

1) The traditional correctness notion, serializability, for transaction execution must be extended or replaced by new correctness notions. The next section discusses two such correctness notions.
2) Transaction interaction in some real-time database environments has similarities with those encountered in active databases [67] in which various events and conditions trigger actions of other transactions. An 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." A major difference between this example and active databases is that the transaction *Notify-Automated-Agent* is given a completion deadline.
3) 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 to 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, by 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 real-time database researchers should investigate this theory of interaction among "programs," regardless of whether they are called programs or transactions, since such programs interact directly with a database. The 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 transaction execution is more important, and both fairness and maximum resource utilization become secondary goals. A related issue is measuring transaction-processing performance. In contrast 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 transaction conflict resolution.

A. Real-Time Temporal Data and Transaction Consistency

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

We make a distinction between *base* (data) *items*, which 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 a *validity interval*, the length of the time following the absolute valid time during which the value is considered 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]:

1) *Absolute data consistency* states that the validity interval of the most recent value of a base (or derived) item is always longer than the time interval between its absolute valid (or absolute transaction) time and "now." This indicates that the data has absolute validity.
2) *Relative data consistency*. Frequently in real-time databases, a set of data items must be observed within a small time interval so that the items can be used to derive a new value for a data item. Such a set 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 must 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 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. Then d is the *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. We can also define the *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 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 is related to the temporal consistency requirements of data items. Real-time systems commonly use real-time databases to store data about physical devices and to set parameters of physical devices. Therefore, transactions must read the "most recent" values of data items as defined by the following two constraints [231]:

1) *External transaction consistency.* Ideally, most real-time transactions would 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 means 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.
2) *Temporal transaction consistency.* Real-time transactions may use a snapshot of the real world. Therefore, data values read by a transaction ideally would 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 proved easier when conflicting transactions were close in the lengths of their periods. Also, the transaction conflict pattern had a more significant effect on temporal transaction consistency than the transaction load level.

Assume that there are write-only transactions that periodically record real-world changes in the database. Two real-time transaction consistency enforcement issues are the following: 1) 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. 2) 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 we've described may be useful. For example, referring to the temporal transaction consistency constraint, 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 to permit interleaved executions of transactions that use incorrect item values (but are bounded in their absolute differences with the correct item values) [54]. More research is needed to find new, general-purpose consistency notions for cooperating transactions in real-time databases.

B. Real-Time Query Languages

There is no reported research on real-time database query languages or transaction specification languages that allow users to specify time constraints, temporal constraints, temporal properties of data values, and semantic knowledge useful for query/transaction processing. The reason for this is that, until recently, specifying a query or a transaction involved only specifying a time constraint and temporal transaction constraints, which can be achieved by means of simple extensions to conventional query languages or transaction specification techniques. In the near future, research will be needed in real-time query and transaction specification languages that will allow users to specify semantic knowledge and interactions of cooperating transactions, and that are based on various temporal data models. These languages may also be based on temporal and modal logics [59] or on models of concurrent processes [123].

VI. Architectural Issues

We now turn to the implementation of 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.

A. Temporal-Query Processing

We discuss two aspects here: query optimization and query evaluation.

Temporal-query optimization is substantially more involved than conventional-query optimization for several reasons. Temporal-query optimization 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 equijoins 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. For example, the TSQL2 OVERLAP operator is translated into two less-than predicates on the underlying timestamps. 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 continuously 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, 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 Ingres DBMS was minimally extended 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 is to avoid looking at all 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 produce all the problems just listed.

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*, significantly improved performance of some queries [9] and was later generalized to allow multiple cached states, which further improve performance [94]. The other three responses were the design of new query optimization strategies, new join algorithms, and new temporal indexes.

A.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 V. Some of these temporal algebras support the tautologies defined in the standard relational algebra, enabling existing query optimizers to be used.

Determining which access method is best for each algebraic operator requires *metadata*—statistics on the stored temporal data—and *cost models*—predictors of the execution cost for each operator implementation/access method combination. Temporal data requires additional metadata, such as the time interval over which the relation is defined (termed the *lifespan* [42]), lifespans of the tuples, surrogate and tuple arrival distributions, distributions of time-varying attributes, regularity and granularity of temporal data, and frequency of the null values sometimes introduced when attributes within a tuple are not 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, with the goal of producing 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 necessary (see [92]).

A.2. Temporal Joins

Researchers have considered a wide variety of binary joins, 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.

A.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. Indexes are even more important in temporal relations that grow monotonically in size. There has been a great deal of research in temporal indexing over the past five years. 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. An empirical comparison would have to consider the differing abilities of each (those supporting no nontemporal keys would be useful for doing temporal cartesian products but perhaps less useful for temporal joins involving equality predicates on nontemporal attributes). It would also have to consider various underlying distributions of time and nontemporal keys (the indexes presume various nonuniform distributions to achieve their performance gains over conventional indexes, which generally assume a uniform key distribution).

B. 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 timestamp 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 written earlier 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 VI.B.2.) Timestamping 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 were made by means of either a testbed system (e.g., [87]) or simulation (e.g., [3], [76], [215]). One exception is the approximate analysis of real-time databases [79], which used an analytical approach to approximate the steady-state fraction of real-time transactions that complete successfully.

B.1. Processing Transactions with Hard Deadlines

All transactions with hard deadlines must complete within their deadlines. This means that they must be scheduled with complete knowledge, which in turn means that the various transaction factors listed in Section V must be known a priori. For example, transaction arrival patterns must be known; that is, 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 transaction execution times can be obtained, and real-time task-scheduling techniques can be used to guarantee timely transaction execution.

B.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 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 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]. These techniques detect conflicts between two real-time transactions or between one real-time transaction and a set of real-time transactions.

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 contrast, 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 aborts. Otherwise, the lock-requesting transaction is blocked. The performances of priority inheritance and priority abort and 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] assign timestamps to transactions when they start, 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, the timestamp-ordering protocols' early conflict resolution 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, instead of a single value, to transactions, and to adjust (reduce) the intervals when conflicts arise to guarantee serializable transactions [19], [230].

Optimistic concurrency control protocols validate (certify) transactions for commitment after they complete execution [74], [77], [78], [87]. Conflicts among transactions are solved with 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 that 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 *forward validation*, in which the validating transaction is checked against the currently running, active transactions, and, in the case of conflict, the validating transaction or other conflicting transactions are aborted. The literature contains various real-time optimistic protocol variants based on forward validation, and their experimental evaluations [74], [87]. Instead of using knowledge of the dynamic read sets of active transactions, these variants use transaction priorities [74], [88], force the validating transaction into a wait state until the conflicts are resolved [74], or dynamically adjust 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, deadlines can be assigned to each subtransaction individually to reflect its importance and urgency [159]. Several heuristics to this *subtask deadline assignment* problem have been proposed and evaluated [107], both for subtransactions that execute serially and for subtransactions that execute in parallel. However, these proposals accommodate only one-level nested transactions. 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 in which transaction serializability is not absolutely necessary, performance may be improved by allowing nonserializable or temporarily nonserializable transaction execution. One such approach is *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].

C. 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 to compare their space and time efficiency) [9]. Page layout for temporal relations is more complicated than for conventional relations if the nonfirst normal form (i.e., nonatomic attribute values) is adopted, as is proposed in many of the temporal data models listed in Section III. 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 and Snodgrass have developed an analytical model to determine the optimal block size for such linked lists [85].

D. Buffer Management for Real-Time Transaction Scheduling

In conventional databases, buffer contention among transactions can impact performance significantly. Prioritized versions of two buffer management algorithms, *priority-least-recently-used (priority-LRU)* and *priority-DBMIN*, have been investigated [33]. Priority-LRU groups buffer elements into priority levels, each level holding 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 exhibits 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 to process read requests first, as long as the write requests can be placed in the write buffer without reducing the free space to 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].

E. Scheduling Disk I/O for Real-Time Transaction Processing

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

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 execution. Read requests are then assigned priorities on the basis of their deadlines. Priority inheritance can be added to this scheme for write requests that block other high-priority transactions (due to write locks) [119].

SCAN algorithms can also be revised with 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].

VII. Conclusions and Future Work

Temporal database research has been active for about 20 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.
- Much research has 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 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 the proposed temporal query languages have a strong formal basis.
- Several commercial temporal object-oriented DBMSs 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 the two areas.
- The interaction of transaction time support and concurrency control and transaction management has been studied to some depth. Most of the research in real-time databases has focused on modifying and adapting the traditional transaction-processing techniques of conventional databases and the task-scheduling techniques of real-time systems.

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 the temporal relational query language TSQL2, we can now investigate the task of database modeling 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* should be developed for real-time databases, perhaps similar to 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 in which subtransactions have execution rules. Also, this theory should be developed both with and without serializability because applications may or may not need serializable transactions. The notion of a linear transaction time model must be modified to accommodate nonserial transactions.
- We should determine whether increased efficiency and ease of use justifies the added complexity of explicit temporal constructs in the object-oriented temporal data model and query language. Resolving this issue is a prerequisite for the design of a consensus temporal object-oriented query language.
- Users need real-time query languages to specify the semantic knowledge captured in real-time data models and to use it in various ways. Also, users need transaction execution specification languages to specify the interactions of cooperating transactions.
- Integrity constraints—particularly time-constrained access, manipulation, and enforcement of (possibly temporal) integrity constraints—must be investigated in depth.
- Achieving adequate performance in a temporal or real-time DBMS remains a challenge. In temporal databases, we need empirical studies comparing temporal join algorithms and temporal indexing. In real-time databases, we need more research on techniques that satisfy integrity constraints and temporal data and transaction constraints and that provide timely enforcement of database view consistency when the underlying database changes.

This development of new models and theories may provide a sound basis for real-time and temporal databases. We feel that closer interaction of the previously isolated research communities will yield database technology supporting all applications involving data with a time component.

ACKNOWLEDGMENTS

Gultekin Özsoyoğlu was supported in part by National Science Foundation grant ISI-9224660. Richard T. Snodgrass was supported in part by NSF grant IRI-9302244 and an AT&T Foundation grant. James Clifford was helpful to our understanding of the structural aspects of time models. 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, 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] R. Abbott and H. Garcia-Molina, "Scheduling real-time transactions with disk resident data," *Proc. Int'l Conf. Very Large Databases*, Aug. 1989.

[2] R. Abbott and H. Garcia-Molina, "Scheduling I/O requests with deadlines: A performance evaluation," *11th Real-Time Systems Symp.*, 1990.

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

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

[5] D. Agrawal and A.E. Abbadi, "Locks with constrained sharing," *Proc. ACM PODS Conf.*, pp. 85–93, 1990.

[6] D. Agrawal, A.E. Abbadi, and A.E. Lang, "Performance characteristics of protocols with ordered shared locks," *Proc. Int'l Conf. Data Eng.*, pp. 592–601, 1991.

[7] D. Agrawal, A.E. Abbadi, and R. Jeffers, "Using delayed commitment in locking protocols for real-time databases," *Proc. ACM Int'l Conf. Management of Data*, pp. 104–113, 1992.

[8] I. Ahn and R.T. Snodgrass, "Performance evaluation of a temporal database management system," C. Zaniolo, ed., *Proc. ACM Int'l Conf. on Management of Data*, pp. 96–107, May 1986.

[9] I. Ahn and R.T. Snodgrass, "Partitioned storage for temporal databases," *Information Systems*, vol. 13, no. 4, pp. 369–391, 1988.

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

[11] A. Ait-Braham, B. Theodoulidis, and G. Karvelis, "Conceptual modeling and manipulation of temporal databases," *Proc. Entity-Relationship Conf.*, 1994.

[12] K.K. Al-Taha, R.T. Snodgrass, and M.D. Soo, "Bibliography on spatiotemporal databasesm," *Int'l J. Geographical Information Systems*, vol. 8, pp. 195–203, Jan.-Feb. 1994.

[13] J. Andany, M. Leonard, and C. Palisser, "Management of schema evolution in databases," *Proc. Conf. Very Large Databases*, Barcelona, Spain, pp. 161–170, Sept. 1991.

[14] G. Ariav, "A temporally oriented data model," *ACM Trans. Database Systems*, vol. 11, no. 4, pp. 499–527, Dec. 1986.

[15] F. Barbic and B. Pernici, "Time modeling in office information systems," S. Navathe, ed., *Proc. ACM Int'l Conf. Management Data*, Austin, Texas, pp. 51–62, May 1985.

[16] F. Bancilhon, "Object-oriented database systems," *Proc. ACM SIGACT/SIGMOD PODS Conf.*, Austin, Texas, pp. 152–162, 1988.

[17] M.A. Bassiouni and M.J. Llewellyn, "A relational-calculus query language for historical databases," *J. Computer Languages*, vol. 17, no. 3, pp. 185–197, 1992.

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

[19] R. Bayer, K. Elhardt, J. Heigert, and A. Reiser, "Dynamic timestamp allocation for transactions in database systems," *Proc. Int'l Symp. Distributed Databases*, pp. 9-20, 1982.

[20] D. Beech and B. Mahbod, "Generalized version control in an object-oriented database," *Proc. Int'l Conf. Very Large Databases*, pp. 14-22, Feb. 1988.

[21] J. Ben-Zvi, "The Time Relational Model," PhD thesis, Computer Science Dept., UCLA, 1982.

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

[23] E. Bertino, "A view mechanism for object-oriented databases," *Proc. Int'l Conf. Extending Database Technology*, Vienna, Austria, Mar. 1992.

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

[25] G. Bhargava and S.K. Gadia, "Relational database systems with zero information loss," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, no. 7, pp. 76–87, Feb. 1993.

[26] S. Biyabani, J.A. Stankovic, and K. Ramamritham, "The integration of deadline and criticalness in hard real-time scheduling," *Proc. Ninth Real-Time Systems Symp.*, pp. 152–160, 1988.

[27] J.A. Blakeley,, "ZQL[C++]: Extending a persistent C++ language with a query capability," Technical Report ITB-91-10-01, Computer Science Laboratory, Texas Instruments, Dallas, Oct. 1991.

[28] M. Böhlen and R. Marti, "Handling temporal knowledge in a deductive database system," *Datenbanksysteme in Büro, Technik und Wisenschaft*, Springer-Verlag, 1993.

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

[30] A.P. Buchmann, D.R. McCarthy, M. Hsu, and U. Dayal, "Time-critical database scheduling: A framework for integrating real-time scheduling and concurrency control," *Proc. IEEE Int'l Conf. Data Eng.*, pp. 470–480, Jan. 1989.

[31] A. Cardenas, I. Ieong, R. Taira, R. Barker, and C. Breant, "The knowledge-based object-oriented PICQUERY+ language," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, no. 4, pp. 644–657, Aug. 1993.

[32] M.J. Carey, D.J. DeWitt, and S.L. Vandenburg, "A data model and query language for EXODUS," *Proc. ACM Int'l Conf. Management Data*, Chicago, pp. 413–423, June 1988.

[33] M.J. Carey, R. Jauhari, and M. Livny, "Priority in DBMS resource scheduling," *Proc. Int'l Conf. Very Large Databases*, 1989.

[34] M. Caruso and E. Sciore, "Meta-functions and contexts in an object-oriented database language," *Proc. ACM Int'l Conf. Management Data*, Chicago, pp. 56–65, June 1988.

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

[36] S. Chakravarthy and S.-K. Kim, "Resolution of time concepts in temporal databases," *Information Sciences*, vol. 80, nos. 1–2, pp. 91–125, Sept. 1994.

[37] S. Chen, J.A. Stankovic, J. Kurose, and D. Townley, "Performance evaluation of two new disk scheduling algorithms for real-time systems," *J. Real-Time Systems*, vol. 3, pp. 307–337, 1991.

[38] J. Chomicki, "Temporal query languages: A survey," H.J. Ohlbach and D.M. Gabbay, eds., *Proc. First Int'l Conf. Temporal Logic*. Lecture Notes in Artificial Intelligence 827, Springer-Verlag, pp. 506–534, July 1994.

[39] J. Chomicki and T. Imelinski, "Relational specifications of infinite query answers," *Proc. ACM Int'l Conf. Management Data*, pp. 174–183, May 1989.

[40] H.T. Chou and W. Kim, "A unifying framework for version control in a CAD environment," *Proc. Int'l Conf. Very Large Databases*, pp. 336–344, 1986.

[41] W.W. Chu, I.T. Ieong, R.K. Taira, and C.M. Breant, "A temporal evolutionary object-oriented data model and its query language for medical image management," *Proc. Very Large Database Conf.*, Aug.1992.

[42] J. Clifford and A. Croker, "The historical relational data model (hrdm) and algebra based on lifespans," *Proc. Int'l Conf. Data Eng.*, pp. 528–537, Los Angeles, Feb. 1987.

[43] J. Clifford, A. Croker, and A. Tuzhilin, "On completeness of historical relational query languages," *ACM Trans. Database Systems*, vol. 19, no. 1, pp. 64–116, Mar. 1994.

[44] J. Clifford and A.U. Tansel, "On an algebra for historical relational databases: Two views," S. Navathe, ed., *Proc. ACM Int'l Conf. Management Data*, pp. 247–265, Austin, Texas, May 1985.

[45] J. Clifford and D.S. Warren, "Formal semantics for time databases," *ACM Trans. Database Systems*, vol. 8, no. 2, pp. 214–254, June 1983.

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

[47] P. Dadam, V. Lum, and H.-D. Werner, "Integration of time versions into a relational database system," U. Dayal, G. Schlageter, and L.H. Seng, eds., *Proc. Conf. Very Large Databases*, pp. 509–522, Singapore, Aug. 1984.

[48] C.J. Date, "A proposal for adding date and time support to SQL," *ACM SIGMOD Record*, vol. 17, no. 2, pp. 53–76, June 1988.

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

[50] U. Dayal and G.T.J. Wuu, "A uniform approach to processing temporal queries," *Proc. Conf. Very Large Databases*, Vancouver, Canada, Aug.1992.

[51] V. DeAntonellis, A. Degli, G. Mauri, and B. Zonta, "Extending the entity-relationship approach to take into account historical aspects of systems," P. Chen, ed., *Proc. Int'l Conf. E-R Approach to Systems Analysis and Design*. North Holland, 1979.

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

[53] W. Dreyer, A.K. Dittrich, and D. Schmidt, "An object-oriented data model for a time series management system," *Proc. Int'l Working Conf. Scientific and Statistical Database Management*, J.C. French and H. Hinterberger, eds. Charlottesville, Va., pp. 185–95, Sept. 1994.

[54] L.B.C. Di Pippo and V.F. Wolfe, "Object-based semantic real-time concurrency control," *Proc. 14th IEEE Real-Time Systems Symp.*, Dec. 1993.

[55] S. Dutta, "Generalized events temporal databases," *Proc. Int'l Conf. Data Eng.*, pp. 118–126, Los Angeles, Feb. 1989.

[56] C.E. Dyreson and R.T. Snodgrass, "Valid-time indeterminacy," *Proc. Int'l Conf. Data Eng.*, pp. 335–343, Vienna, Austria, Apr. 1993.

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

[58] R. Elmasri and G.T.J. Wuu, "A temporal model and query language for ER databases," *Proc. Int'l Conf. Data Eng.*, pp. 76–83, Feb. 1990.

[59] E.A. Emerson, "Temporal and modal logic," J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, pp. 997–1,067, Elsevier Science Publishers, B.V., 1990.

[60] K.P. Eswaran, J.N. Gray, R.A. Lorie, and I.L. Traiger, "The notions of consistency and predicate locks in a database system," *Comm. ACM*, vol. 19, no. 11, pp. 624–633, 1976.

[61] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, R.T. Snodgrass and M. Winslett, eds., pp. 419–429, Minneapolis, May 1994.

[62] A.U. Frank, "Qualitative temporal reasoning in GIS-ordered time scales," Technical Report, Dept. of Geo-Information, Technische Univ., Vienna, 1994.

[63] S.K. Gadia, "Toward a multihomogeneous model for a temporal database," *Proc. Int'l Conf. Data Eng.*, pp. 390–397, Los Angeles, Feb. 1986.

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

[65] S.K. Gadia, S. Nair, and Y.-C. Poon, "Incomplete information in relational temporal databases," *Proc. Conf. Very Large Databases*, Vancouver, Canada, Aug.1992.

[66] S.K. Gadia, and C.S. Yeung, "A generalized model for a relational temporal database," *Proc. ACM Int'l Conf. Management of Data*, pp. 251–259, Chicago, June 1988.

[67] N.H. Gehani, H.V. Jagadish, and O. Shumeli, "Event specification in an active object-oriented database," *Proc. ACM SIGMOD Conf.*, pp. 81–90, San Diego, Calif., June 1992.

[68] I. Goralwalla and M.T. Özsu, "Temporal extensions to a uniform behavioral object model," *Proc. Int'l Conf. Entity-Relationship Approach*, Dallas, June 1993.

[69] M. H. Graham, "Issues in real-time data management," *J. Real-Time Systems*, vol. 4, pp. 185–202, 1992.

[70] H. Gunadhi and A. Segev, "A framework for query optimization in temporal databases," *Proc. Int'l Conf. Statistical and Scientific Database Management Systems*, vol. 420, pp. 131–147, Springer Verlag, Apr. 1990.

[71] H. Gunadhi and A. Segev, "Query processing algorithms for temporal intersection joins," *Proc. Int'l Conf. Data Eng.*, Kobe, Japan, 1991.

[72] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys*, vol. 15, no. 4, pp. 287–317, 1983.

[73] M. Hammer and D. McLeod, "Database description with SDM: A semantic database model," *ACM Trans. Database Systems*, vol. 6, no. 3, pp. 351–386, Sept. 1981.

[74] J.R. Haritsa, M.J. Carey, and M. Livny, "Dynamic real-time optimistic concurrency control," *Proc. 11th Real-Time Systems Symp.*, 1990.

[75] J.R. Haritsa, M. Livny, and M.J. Carey, "Earliest deadline scheduling for real-time database systems," *Proc. 12th Real-Time Systems Symp.*, 1991.

[76] J.R. Haritsa, M.J. Carey, and M. Livny, "Value-based scheduling in real-time database systems," *VLDB J.*, vol. 2, no. 2, 1993.

[77] J.R. Haritsa, M.J. Carey, and M. Livny, "Data access scheduling in firm real-time database systems," *J. Real-Time Systems*, vol. 4, pp. 203–241, 1992.

[78] J.R. Haritsa, M.J. Carey, and M. Livny, "On being optimistic about real-time constraints," *Proc. ACM PODS Conf.*, pp.331–343, 1990.

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

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

[81] S. Heiler, S. Haradhvala, S. Zdonik, B. Blaustein, and A. Rosenthal, "A flexible framework for transaction management in engineering environments," *Database Transaction Models for Advanced Applications*, A.K. Elmagarmid, ed., Morgan Kauffman, 1993.

[82] G.D. Held, M. Stonebraker, and E. Wong, "Ingres–A relational data base management system," *Proc. Nat'l Computer Conf.*, vol. 44, pp. 409–416, Anaheim, Calif., May 1975.

[83] D. Hong, T. Johnson, and S. Chakravarthy, "Real-time transaction scheduling: A cost conscious approach," *Proc. ACM Int'l Conf. Management Data*, 1993.

[84] W-C. Hou and G. Özsoyoğlu, "Processing real-time aggregate queries in CASE-DB," *ACM Trans. Database Systems*, pp. 224–261, June 1993.

[85] A. Hsu and R.T. Snodgrass, "Optimal block size for set-valued attributes," *Information Processing Letters*, vol. 45, no. 3, pp. 153–158, Mar. 1993.

[86] J. Huang, J. Stankovic, D. Towsley, and K. Ramamritham, "Experimental evaluation of real-time transaction processing," *Proc. Real-Time Systems Symp.*, Dec. 1989.

[87] J. Huang, J. Stankovic, K. Ramamritham, and D. Towsley, "Experimental evaluation of real-time optimistic concurrency control schemes," *Proc. Int'l Conf. Very Large Databases*, pp. 35–46, 1991.

[88] J. Huang, J. Stankovic, K. Ramamritham, and D. Towsley,, "On using priority inheritance in real-time databases," *Proc. 12th Real-Time Systems Symp.*, pp. 210–221, 1991.

[89] J. Huang, J. Stankovic, K. Ramamritham, D. Towsley, and B. Purimetla, "Priority inheritance in soft real-time databases," *J. Real-Time Systems*, vol. 4, pp. 243–268, 1992.

[90] R. Jauhari, M.J. Carey, and M. Livny, "Priority-hints: An algorithm for priority-based buffer management," *Proc. Int'l Conf. Very Large Databases*, pp. 708–721, 1990.

[91] E.D. Jensen, C. Douglas, C.D. Locke, and H. Tokuda, "A time-driven scheduler for real-time operating systems," *Proc. IEEE Real-Time System Symp.*, Dec. 1986.

[92] C.S. Jensen, L. Mark, and N. Roussopoulos, "Incremental implementation model for relational databases with transaction time," *IEEE Trans. Knowledge and Data Eng.*, vol. 3, no. 4, pp. 461–473, Dec. 1991.

[93] C.S. Jensen and L. Mark,. "Queries on change in an extended relational model," *IEEE Trans. Knowledge and Data Eng.*, vol. 4, no. 2, pp. 192–200, Apr. 1992.

[94] C.S. Jensen, L. Mark, and N. Roussopoulos, and T. Sellis, "Using differential techniques to efficiently support transaction time," *VLDB J.*, vol. 2, no. 1, pp. 75–111, Jan. 1993.

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

[96] C.S. Jensen,"Vacuuming in TSQL2," *TSQL2 Commentary*, Sept.1994.

[97] C.S. Jensen and R.T. Snodgrass, "The TSQL2 data model," *TSQL2 Commentary*, Sept. 1994.

[98] C.S. Jensen and R.T. Snodgrass, "Temporal specialization and generalization," *IEEE Trans. Knowledge and Data Eng.*, vol. 6, no. 6, pp. 954–974, Dec. 1994.

[99] C.S. Jensen, M.D. Soo, and R.T. Snodgrass, "Unifying temporal models via a conceptual model," *Information Systems*, vol. 19, no. 7, pp. 513–547, Dec. 1994.

[100] L. Jerome, "Real-time systems moving to Unix and Sun," *Sun Observer J.*, Dec. 1989.

[101] S. Jones, P. Mason, and R. Stamper, "Legol 2.0: A relational specification language for complex rules," *Information Systems*, vol. 4, no. 4, pp. 293–305, Nov. 1979.

[102] W. Käfer, "History and version management of complex objects" in German, PhD thesis, Fachbereich Informatik, Univ. Kaiserslautern, 1992.

[103] W. Käfer and H. Schöning, "Mapping a version model to a complex-object data model," *Proc. Int'l Conf. Data Eng.*, pp. 348–357, 1992.

[104] W. Käfer and H. Schöning,"Realizing a temporal complex-object data model," *Proc. ACM Int'l Conf. Management Data*, pp. 266–275, 1992.

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

[106] M. Kamath and K. Ramamritham, "Performance characteristics of epsilon serializability with hierarchical inconsistency bounds," *Proc. IEEE Int'l Conf. Data Eng.*, pp. 587–594, 1993.

[107] B. Kao and H. Garcia-Molina, "Deadline assignment in a distributed soft real-time system," *Proc. 13th Int'l Conf. Distributed Computing Systems*, pp. 428–437, 1993.

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

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

[110] W. Kim and H.T. Chou, "Versions of shema OODB," *Proc. Int'l Conf. Very Large Databases*, pp. 148–159, Long Beach, Calif., 1988.

[111]W. Kim, J.F. Garza, N. Ballou, and D. Woelk, "Architecture of the orion next-generation database system," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, no. 1, pp. 109–124, Mar. 1990.

[112]S.-K. Kim and S. Chakravarthy, "Temporal databases with two-dimensional time: Modeling and implementation of multihistory," *Information Sciences*, vol. 80, nos. 1–2, pp. 43–89, Sept. 1994.

[113]K.A. Kimball, "The DATA system," master's thesis, Univ. of Pennsylvania, 1978.

[114]N. Kline, "An update of the temporal database bibliography," *ACM SIGMOD Record*, vol. 22, no. 4, pp. 66–80, Dec. 1993.

[115]M.R. Klopprogge, "TERM: An approach to include the time dimension in the entity-relationship model," *Proc. Int'l Conf. Entity Relationship Approach*, pp. 477–512, Washington, D.C., Oct. 1981.

[116]H. F. Korth, N. Soparkar, and A. Silberschatz, "Triggered real-time databases with consistency constraints," *Proc. Int'l Conf. Very Large Databases*, 1990.

[117]M. Koubarakis, "Representation and querying temporal databases: The power of temporal constraints," *Proc. Int'l Conf. Data Eng.*, pp. 327–334, Vienna, Apr. 1993.

[118]T.W. Kuo and A. K. Mok, "Application semantics and concurrency control of real-time data-intensive applications," *Proc. Real-Time Systems Symp.*, pp. 35–45, 1992.

[119]W. Kim and J. Srivastava, "Enhancing real-time DBMS performance with multiversion data and priority based disk scheduling," *Proc. 12th Real-Time Systems Symp.*, pp. 222–231, 1991.

[120]Y-K. Kim and S.H. Son, "An approach towards predictable real-time transaction processing," *Proc. Fifth Euromicro Workshop Real-Time Systems*, Oulu, Finland, 1993.

[121]H.T. Kung and J.T. Robinson, "On optimistic concurrency control," *ACM Trans. Database Systems*, vol. 6, no. 2, pp. 213–226, 1981.

[122]P. Ladkin, "The logic of time representation," PhD thesis, Univ. of California, Berkeley, Nov. 1987.

[123]L. Lamport and N. Lynch, "Distributed computing: Models and methods," J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, pp. 1,159–1,196, Elsevier Science Publishers, B.V., 1990.

[124]J. Lee and S.H. Son, "Using dynamic adjustment of serialization order for real-time database systems," *Proc. 14th IEEE Real-Time Systems Symp.*, Raleigh-Durham, N.C., Dec. 1993.

[125]J. Lee and S.H. Son, "Issues in developing object-oriented database systems for real-time applications," *Proc. IEEE Workshop Real-Time Applications*, Washington, D.C., 1994.

[126]J. Lee and S.H. Son, "Performance of concurrency control algorithms for real-time database systems," *Performance of Concurrency Control Mechanism Centralized Database Systems*, V. Kumar, ed., Prentice Hall, to appear in1995.

[127]R.M. Lee, H. Coelho, and J.C. Cotta, "Temporal inferencing on administrative databases," *Information Systems*, vol. 10, no. 2, pp. 197–206, 1985.

[128]T.Y. Leung and R. Muntz, "Query processing for temporal databases," *Proc. Int'l Conf. Data Eng.*, Los Angeles, Feb. 1990.

[129]T.Y.C. Leung and R.R. Muntz, "Stream processing: temporal query processing and optimization," chap. 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, Calif., pp. 329–355, 1993.

[130]K.J. Lin, S. Natarajan, and W.S. Liu, "Imprecise results: utilizing partial computations in real-time systems," *Proc. Real-Time Systems Symp.*, pp. 210–217, 1987.

[131]K.J. Lin, "Designing databases in real-time embedded systems," *Proc. IEEE Symp. Computer-Aided Control System Design*, pp. 285–292, Mar. 1992.

[132]K.J. Lin, "Consistency issues in real-time database systems," *Proc. 22nd Hawaii Int'l Conf. System Sciences*, Honolulu, Jan. 1989.

[133]Y. Lin and S.H. Son, "Concurrent control in real-time databases by dynamic adjustment of serialization order," *Proc. 11th IEEE Real-Time Systems Symp.*, Orlando, Fla., Dec. 1990.

[134]C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[135]J. Liu, K. Lin, W. Shih, A. Yu, J. Chung, and W. Zhao, "Algorithms for scheduling imprecise computation," *Computer*, vol. 24, no. 5, May 1991.

[136]J. Liu, K. Lin, and X. Song, "A position paper on scheduling hard real-time transactions" *Proc. 1988 Workshop Real-Time Operating Systems and Software*, pp. 12-13, May 1988.

[137]D. Lomet and B. Salzberg, "Access methods for multiversion data," *Proc. ACM Int'l Conf. Management Data*, pp. 315–324, June 1989.

[138]D. Lomet and B. Salzberg, "Transaction-time databases," chap. 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, Calif., pp. 388–417, 1993.

[139]N.A. Lorentzos and R.G. Johnson, "Extending relational algebra to manipulate temporal data," *Information Systems*, vol. 13, no. 3, pp. 289–296, 1988.

[140]V. Lum, P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner, and J. Woodfill, "Designing DBMS support for the temporal dimension," B. Yormark, ed., *Proc. ACM Int'l Conf. Management Data*, pp. 115–130, Boston, June 1984.

[141]P. Lyngbæk, "OSQL: A language for object databases," Technical Report HPL-DTD-91-4, Hewlett-Packard Laboratories, Palo Alto, Calif., Jan. 1991.

[142]M. McKenzie, "Bibliography: Temporal databases," *ACM SIGMOD Record*, vol. 15, no. 4, pp. 40–52, Dec. 1986.

[143]E. McKenzie and R.T. Snodgrass, "Schema evolution and the relational algebra," *Information Systems*, vol. 15, no. 2, pp. 207–232, June 1990.

[144]E. McKenzie and R.T. Snodgrass,"An evaluation of relational algebras incorporating the time dimension databases," *ACM Computing Surveys*, vol. 23, no. 4, pp. 501–543, Dec. 1991.

[145]E. McKenzie and R.T. Snodgrass, "Supporting valid time in an historical relational algebra: Proofs and extensions," Technical Report TR-91-15, Dept. of Computer Science, Univ.of Arizona, Tucson, Aug. 1991.

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

[147]R. Montague, *The Proper Treatment of Quantification in Ordinary English*. D. Reidel Publishing, Dordrecht, Holland, 1973.

[148]A. Montanari and B. Pernici, "Temporal Reasoning," chap. 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, Calif., 1994.

[149]J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: Representing knowledge about information systems," *ACM Trans. Office Information Systems*, vol. 8, no. 4, pp. 325–362, Oct. 1990.

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

[151]M.H. Nodine, S. Ramaswamy, and S.B. Zdonik, "A cooperative transaction model for design databases," *Database Transaction Models for Advanced Applications*, A.K. Elmagarmid, ed., Morgan Kauffman, 1993.

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

[153]R Overmyer and M. Stonebraker, "Implementation of a time expert in a database system," *ACM SIGMOD Record*, vol. 12, no. 3, pp. 51–59, Apr. 1982.

[154]G. Özsoyoğlu and A. Hafez, "Near-optimum storage models for nested relations based on workload information," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, no. 6, pp. 1,018-1,038, Dec. 1993.

[155]G. Özsoyoğlu, S. Guruswamy, K. Du, and W-C. Hou, "Time-constrained query processing in CASE-DB," *IEEE Trans. Knowledge and Data Eng.*, to appear in 1995.

[156]M.T. Özsu, R. Peters, D. Szafron, B. Irani, A. Lipka, and Muñoz, "TIGUKAT: A uniform behavioral objectbase management system," *VLDB J.*, to appear in 1995.

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

[158]C. Pu and A. Leff, "Replica control in distributed systems: An asynchronous approach," *Proc. ACM Int'l Conf. Management Data*, pp. 377–386, 1991.

[159]K. Ramamritham, "Real-time databases," *Int'l J. Distributed and Parallel Databases*, vol. 1, no. 2, pp. 199–206, 1993.

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

[161]J.F. Roddick, "Schema evolution in database systems—An annotated bibliography," *ACM SIGMOD Record*, vol. 21, no. 4, pp. 35–40, Dec. 1992.

[162]J.F. Roddick and R.T. Snodgrass, "Schema versioning support in TSQL2," *TSQL2 Commentary*, Sept.1994.

[163]E. Rose and A. Segev, "TOODM—A temporal object-oriented data model with temporal constraints," *Proc. Int'l Conf. Entity Relationship Approach*, Oct. 1991.

[164]E. Rose and A. Segev, "TOOA: A temporal object-oriented algebra," *Proc. European Conf. Object-Oriented Programming*, July 1993.

[165]E. Rose and A. Segev, "TOOSQL—A temporal object-oriented query language," *Proc. Int'l Conf. Entity-Relationship Approach*, Dallas, 1993.

[166]R. Sadeghi, "A Database Query Language for Operations on Historical Data," PhD thesis, Dundee College of Technology, Dundee, Scotland, Dec. 1987.

[167]R. Sadeghi, W.B. Samson, and S.M. Deen, "HQL—A historical query language," Technical report, Dundee College of Technology, Dundee, Scotland, Sept. 1987.

[168]B. Slazberg and V.J. Tsotras, "A comparison of access methods for time evolving data," Technical Report CATT-TR-94-81, Polytechnic Univ., 1994.

[169]N. Sarda, "Algebra and query language for a historical data model," *The Computer J.*, vol. 33, no. 1, pp. 11–18, Feb. 1990.

[170]N. Sarda, "Extensions to SQL for historical databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, no. 2, pp. 220–230, June 1990.

[171]K. Satoh, M. Tsuchida, F. Nakamura, and K. Oomachi, "Local and global query optimization mechanisms for relational databases," A. Pirotte and Y. Vassiliou, eds., *Proc. Int'l Conf. Very Large Databases*, pp. 405–417, Stockholm, Sweden, Aug. 1985.

[172]E. Sciore, "Using annotations to support multiple kinds of versioning in an object-oriented database system," *ACM Trans. Database Systems*, vol. 16, no. 3, pp. 417–438, Sept. 1991.

[173]E. Sciore, "Versioning and configuration management in an object-oriented data model," *VLDB J.*, to appear in 1995.

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

[175]A. Segev and A. Shoshani, "Logical modeling of temporal data," U. Dayal and I. Traiger, eds., *Proc. ACM SIGMOD Int'l Conf. Management Data*, pp. 454–466, San Francisco, May 1987.

[176]A. Segev and R. Chandra, "A data model for time-series analysis," *Advanced Database Systems*, N.R. Adam and B.K. Bhargava, eds., Springer-Verlag, pp. 191–212, 1993.

[177]T.K. Sellis, "Global query optimization," C. Zaniolo, ed., *Proc. ACM Int'l Conf. Management Data*, pp. 191–205, Washington, DC, May 1986.

[178]P. Seshadri, M. Livny, and R. Ramakrishnan, "Sequence query processing," *Proc. ACM SIGMOD Int'l Conf. Management Data*, R.T. Snodgrass and M. Winslett, eds., pp. 430–441, Minneapolis, May 1994.

[179]L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: an approach to real-time synchronization," *IEEE Trans. Computers*, vol. 39, pp. 1,175–1,185, 1990.

[180]L. Sha, R. Rajkumar, S.H. Son, and C. Chang, "A real-time locking protocol," *IEEE Trans. Computers*, vol. 40, 7, pp. 782–800, July 1991.

[181]Y. Shahar, S.W. Tu, and M.A. Musen, "Knowledge acquisition for temporal-abstraction mechanisms," *Knowedge Acquisition*, vol. 4, no. 2, pp. 217-236, June 1992.

[182]Y. Shahar and M.A. Musen, "RESUME: A temporal-abstraction system for patient monitoring," *Computers and Biomedical Research*, vol. 26, no. 3, pp. 255–273, June 1993.

[183]S. Shenoy and G. Özsoyoğlu, Z., "Design and implementation of a semantic query optimizer," *IEEE Trans. Data and Knowledge Eng.* vol. 1, no. 3, pp. 344–361, Sept. 1989.

[184]K.P. Smith and J.W.S. Liu, "Monotonically improving approximate answers to relational algebra queries," *Proc. COMPSAC 89 Conf.*, Orlando, Fla., Sept. 1989.

[185]R.T. Snodgrass and I. Ahn, "Temporal databases," *Computer*, vol. 19, no. 9, pp. 35–42, Sept. 1986.

[186]R.T. Snodgrass, "The temporal query language TQuel," *ACM Trans. Database Systems*, vol. 12, no. 2, pp. 247–298, June 1987.

[187]R.T. Snodgrass, "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 Computer Science 639, pp. 22–64, Sept.1992.

[188]R.T. Snodgrass, I. Ahn, G. Ariav, D.S. Batory, J. Clifford, C.E. Dyreson, R. Elmasri, F. Grandi, C.S. Jensen, W. Käfer, N. Kline, K. Kulkanri, C.Y.T. Leung, N. Lorentzos, J.F. Roddick, A. Segev, M.D. Soo, and S.M. Sripada, "TSQL2 language specification," *ACM SIGMOD Record*, vol. 23, no. 1, pp. 65–86, Mar. 1994.

[189]R.T. Snodgrass, "Temporal object-oriented databases: A critical comparison," chap. 19, *Modern Database Systems: The Object Model, Interoperability and Beyond*, W. Kim, ed., Addison-Wesley/ACM Press, pp. 386–408, 1995.

[190]"Real-time database systems: Issues and approaches," S.H. Son, ed., *ACM SIGMOD Record*, vol. 17, no. 1, special issue on real-time database systems, Mar. 1988.

[191]S.H. Son, J. Lee, and H. Kang, "Approaches to design of real-time database systems," *Proc. Symp. Database Systems for Advanced Applications*, Korea, pp. 274–281, Apr. 1989.

[192]S.H Son and S. Kouloumbis, "Replication control for distributed real-time database systems," *Proc. 12th Distributed Computing System Conf.*, 1992.

[193]S.H. Son, S. Park, and Y. Lin, "An integrated real-time locking protocol," *Proc. Int'l Conf. Data Eng.*, pp. 527–534, 1992.

[194]S.H. Son and J. Lee, "Scheduling real-time transactions in distributed database systems," *Proc. Seventh IEEE Workshop Real-Time Operating Systems and Software*, Charlottesville,Va., pp. 39–43, May 1990.

[195]S.H. Son, J. Lee, and Y. Lin, "Hybrid protocols using dynamic adjustment of serialization order for real-time concurrency control," *Real-Time Systems J.*, pp. 269–276, Sept. 1992.

[196]X. Song and J. Liu, "How well can data temporal consistency be maintained?" *Proc. IEEE Symp. Computer-Aided Control System Design*, pp. 276–284, Mar. 1992.

[197]M.D. Soo, "Bibliography on temporal databases," *ACM SIGMOD Record*, vol. 20, no. 1, pp. 14–23, Mar. 1991.

[198]M.D. Soo, C.S. Jensen, and R.T. Snodgrass, "An algebra for TSQL2," *TSQL2 Commentary*, Sept. 1994.

[199] M.D. Soo, R.T. Snodgrass, and C.S. Jensen, "Efficient evaluation of the valid-time natural join," *Proc. Int'l Conf. Data Eng.*, pp. 282–292, Feb. 1994.

[200] S. Sripada, "A logical framework for temporal deductive databases," *Proc. Int'l Conf. Very Large Databases*, pp. 171–182, Los Angeles, 1988.

[201]R. Stam and R.T. Snodgrass, "A bibliography on temporal databases," *IEEE Database Eng.*, vol. 7, no. 4, pp. 231–239, Dec. 1988.

[202]M. Stefik, D. Bobrow, and K. Kahn, "Integrating access-oriented programming into a multiparadigm environment," *IEEE Software*, vol. 3, no. 1, pp. 10–18, Jan. 1986.

[203]J. Stankovic, "Misconceptions about real-time computing," *Computer*, vol. 21, no. 10, pp. 10–19, Oct. 1988.

[204]M. Stonebraker, "The design of the Postgres storage system," P. Hammersley, ed., *Proc. Int'l Conf. Very Large Databases*, pp. 289–300, Brighton, England, Sept. 1987.

[205]M. Stonebraker, L. Rowe, and M. Hirohama, "The implementation of Postgres," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, no. 1, pp. 125–142, Mar. 1990.

[206]S.Y.W. Su and H.M. Chen, "A temporal knowledge representation model OSAM*/T and its query language OQL/T," *Proc. Int'l Conf. Very Large Databases*, 1991.

[207]*Temporal Databases: Theory, Design, and Implementation*, A. Tansel,

J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R.T. Snodgrass, eds., Database Systems and Applications Series. Benjamin/Cummings, Redwood City, Calif., 1994.

[208] A.U. Tansel, "Adding time dimension to relational model and extending relational algebra," *Information Systems*, vol. 11, no. 4, pp. 343–355, 1986.

[209] A.U. Tansel, M.E. Arkun, and G. Özsolyoğlu, "Time-by-example query language for historical databases," *IEEE Trans. Software Eng.*, vol 15, no. 4, pp. 464–478, Apr. 1989.

[210] A.U. Tansel, "A historical query language," *Information Sciences*, vol. 53, pp. 101–133, 1991.

[211] P.A. Thompson, "A temporal data model based on accounting principles," PhD thesis, Dept. of Computer Science, Univ. of Calgary, Calgary, Alta., Canada, Mar. 1991.

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

[213] O. Ulusoy, "Research issues in real-time database systems," Tech. Report BU-CEIS-9432, Bilkent Univ., Ankara, Turkey, 1994.

[214] O. Ulusoy, "A study of two transaction processing architectures for distributed real-time database systems," *J. Systems and Software*, to appear in 1995.

[215] O. Ulusoy and G.G. Belford, "Real-time transaction scheduling in database systems," *Information Systems J.*, vol. 18, no. 8, 1993.

[216] O. Ulusoy and G.G. Belford, "Real-time lock based concurrency control in a distributed database system," *Proc. 12th Int'l Conf. Distributed Computing Systems*, pp. 136–143, 1992.

[217] S.D. Urban and L.M.L. Delcambre, "An analysis of the structural, dynamic, and temporal aspects of semantic data models," *Proc. Int'l Conf. Data Eng.*, IEEE CS. Order No. 655, pp. 382–389, Los Angeles, Feb. 1986.

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

[219] S. Vrbsky and K.J. Lin, "Recovering imprecise transaction with real-time constraints," *Proc. Seventh Symp. Reliable Distributed Systems*, pp. 185–193, Oct. 1988.

[220] S.V. Vrbsky, J.W.S. Liu, and K.P. Smith, "An object-oriented query processor that produces monotonically improving approximate answers," *Proc. IEEE Conf. Data Eng.*, Kobe, Japan, 1991.

[221] X. Wang, S. Jajodia, and V. Subrahamanian, "Temporal modules: An approach toward temporal databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, R.T. Snodgrass and M. Winslett, eds., Minneapolis, pp. 227–236, May 1994.

[222] G. Wiederhold, J.F. Fries, and S. Weyl, "Structured organization of clinical data bases," *Proc. Nat'l Computing Conf.*, pp. 479–485, 1975.

[223] G. Wiederhold, S. Jajodia, and W. Litwin, "Dealing with granularity of time in temporal databases," *Proc. Third Nordic Conf. Advanced Information Systems Eng.*, Trondheim, Norway, May 1991.

[224] K. Wilkinson, P. Lyngbæk, and W. Hasan, "The IRIS architecture and implementation," *IEEE Trans. Knowledge and Data Eng.*, vol. 2, no. 1, pp. 63–75, Mar. 1990.

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

[226] K.L. Wu, P.S. Yu, and C. Pu,, "Divergence control for epsilon serializability," *Proc. Int'l Conf. Data Eng.*, pp. 2–11, 1992.

[227] G.T.J. Wuu, "SERQL: An ER query language supporting temporal data retrieval," *Proc. 10th Int'l Phoenix Conf. Computers and Comm.*, pp. 272–279, Mar. 1991.

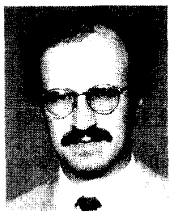
[228] G.T.J. Wuu and U. Dayal, "A uniform model for temporal object-oriented databases," *Proc. Int'l Conf. Data Eng.*, pp. 584–593, Tempe, Ariz., Feb. 1992.

[229] C. Yau and G.S.W. Chat, "TempSQL–A language interface to a temporal relational model," *Information Science and Technology*, pp. 44–60, Oct. 1991.

[230] P.S. Yu, H.-U. Heiss, and D.M. Dias, "Modeling and analysis of a time-stamp history based certification protocol for concurrency control," *IEEE Trans. Knowledge Eng.*, vol. 3, no. 4, pp. 525–537, Dec. 1991.

[231] P.S. Yu, K-L. Wu, K-J. Lin, and S.H. Son, "On real-time databases: concurrency control and scheduling," *Proc. IEEE*, vol. 82, no. 1, 1994.

[232] M. Zloof, "Query by example," *Proc. Nat'l Computer Conf.*, vol. 44, 1975.



Gultekin G. Özsoyoğlu received a BA degree in electrical engineering and an MS degree in computer science from the Middle East Technical University, Ankara, Turkey, in 1972 and 1974, respectively. He received a PhD degree in computing science from the University of Alberta, Canada, in 1980. He is a professor in the Department of Computer Engineering and Science at Case Western Reserve University in Cleveland, Ohio.

Dr. Özsoyoğlu's research interests include real-time databases, multimedia databases, scientific and statistical databases, and graphical user interfaces. He is the author of 60 papers for database conferences and journals such as *ACM Transactions on Database Systems*, *IEEE Transactions on Software Engineering*, *IEEE Transactions on Knowledge and Data Engineering*, and the *Journal of Computer and System Sciences*.

He has served on program committees and panels of major database conferences. He was an ACM national lecturer, program chair of the Third Statistical and Scientific Database Conference, workshop general chair at the CIKM'94 Conference, and research prototypes chair of ACM SIGMOD'94. He is an associate editor of the *Journal of Database Administration* and has served on National Science Foundation, National Reserach Council, and Ford Foundation panels.

Richard T. Snodgrass received a BA degree in physics from Carleton College in 1977 and a PhD degree in computer science from Carnegie Mellon University in 1982. In 1989, he joined the faculty of the University of Arizona in Tucson, where he is now an associate professor in the Department of Computer Science.

His research interests include temporal databases, database query languages, data models, and computer-aided software engineering environments and databases. He has written or coedited three books, including *Temporal Databases: Theory, Design and Implementation*, and approximately 40 journal and conference papers. He is an associate editor of *ACM Transactions on Database Systems* and is on the editorial board of the *International Journal of Computer and Software Engineering*.

Dr. Snodgrass chaired the program committees for the 1994 SIGMOD Conference and the International Workshop on an Infrastructure for Temporal Databases. In addition, he has served as a vice chair or member of many program committees—seven in the past three years—as well as several National Science Foundation panels. He chaired the TSQL2 Language Design Committee and is now working closely with the ANSI and ISO SQL3 committees to add temporal support to that language.