Towards an Infrastructure for Temporal Databases

Report of an Invitational ARPA/NSF Workshop

Niki Pissinou, Richard T. Snodgrass, Ramez Elmasri, Inderpal S. Mumick, M. Tamer Özsu, Barbara Pernici, Arie Segev, Babis Theodoulidis and Umeshwar Dayal

TR 94-01

March 1, 1994

Executive Summary

Temporal databases has been an active area of research for the the last fifteen years, with a corpus nearing 800 papers. While most applications need to store time-varying data, there are no widely used commercial temporal databases. A primary reason for the absence of technology transfer from research to practice is the lack of a commonly accepted consensus data model or query language upon which to base research and development. Even the terminology is inconsistent.

On June 14–16, 1993, the International Workshop on an Infrastructure for Temporal Databases was held in Arlington, TX. Forty-four prominent researchers from ten countries and from ten industrial labs and companies and from many universities participated in this workshop. Much of the workshop comprised intensive meetings of four working groups. Additionally, there was substantial effort both before the meeting, to prepare infrastructure proposals for debate, and after the meeting, to build on the insights that emerged from the discussion.

The first working group discussed characteristics of the various application areas that could benefit from temporal databases, and strove to identify essential features to be provided by a temporal database system in order to be usable by these different communities as well as features and functionality for which additional research is needed. This group emphasized the need for (a) identifying core functionality to be provided by the temporal database management system (TDBMS), and the associated removal of such features such as temporal reasoning from this infrastructure, (b) ways to bridge the "conceptual gap" between the TDBMS research community and the potential user community, and (c) an open architecture, so that handling for time could be integrated into existing tools, instead of requiring the wholesale incorporation of a TDBMS.

The second group considered the much narrower topic of how SQL-92 could be extended to support time. The primary realization was that there were three fundamental, and contradictory, viewpoints on how time should be incorporated in SQL: (a) the required additional support is primarily in the algebraic operators and in the syntax; the underlying data model need not be changed; (b) efforts should be directed solely towards adding time to the SQL3 proposal; and (c) a two-pronged approach should be adopted in which parallel efforts would consider adding time to SQL-92 and to SQL3. Whatever the approach, it was agreed that the temporal data model underlying the language be designed solely in terms of its semantic properties, with distinct and possibly multiple data models being employed for representation and presentation.

The third working group attempted to identify infrastructure for the next generation of temporal database concepts, including extensions of the relational data model as well as the adoption of concepts from the semantic, object-oriented, and active data models. The participants felt that the object-oriented data models, particularly in conjunction with SQL3, provide the most appropriate basis for future work. However, research into temporal object bases and temporal active databases is still in its very early stages, and so the infrastructure discussed here entails primarily a core set of concepts and terminology.

The final working group attempted to classify representational models and system implementation techniques and to propose a TDBMS reference architecture. It was decided that the reference architecture should not differ markedly from those of non-temporal DBMSs, and that extensibility at all levels is a vital characteristic of the architecture, if diverse storage models, temporal algebras, execution algorithms, and query optimization strategies are to be employed.

Common themes ran through the discussions of all four groups. The infrastructure must be based on a base set of desired features, so that most temporal applications receive at least some support from the temporal DBMS. Terminology is critical. Aspects of the conceptual model must be separated from concerns of the representation. The baseline architecture must be extensible, and should identify what is different about a temporal DBMS, and what can vary between TDBMS implementations.

Contents

List of Figures

1 Introduction

Temporal databases has been an active area of research for the last fifteen years, with a corpus nearing 700 papers [KLINE93]. Most database conferences include at least one paper on temporal databases (TDB). Temporal databases are now discussed in several undergraduate database textbooks. There are perhaps one hundred researchers actively studying temporal databases.

During that time an astonishing diversity of temporal data models and query languages has arisen. Most applications, whether business, engineering, medical, or scientific, need to store historical data.

Surprisingly, in spite of both this substantial activity and this pressing requirements from the user community, there are no widely used commercial temporal database management systems (TDBMS). One view is that there is an embarrassment of riches in the TDB literature: with so many alternative approaches from which to choose, it is safer for a DBMS vendor not to choose than to choose an approach that ultimately yields to a competing alternative. The same phenomenon may be occurring in TDB research. In contrast to the flurry of activity in query languages and data models, there is a dearth of results in temporal database design and temporal query optimization, in part because there is no commonly accepted consensus data model or query language upon which to base research and development. At a more fundamental level, even the terminology is highly nonstandard. As an example, the terms intrinsic time, logical time, real-world time, and valid time have all been used for the core concept of the time at which something happened.

It was decided in early 1992 that a meeting should be held with the objective of identifying a common infrastructure to provide a foundation for implementation and standardization as well as for further research into temporal databases. Subsequently, on June 14–16, 1993, the International Workshop on an Infrastructure for Temporal Databases was held in Arlington, Texas. Forty-four prominent researchers from ten countries (Belgium, Canada, Denmark, Israel, Italy, Germany, Greece, Switzerland, the UK, and the US) and from ten industrial labs and companies (Beckman Institute, Bell Communications Research, AT&T Bell Labs, Digital Systems Research, the European Computer-Industry Research Centre, Honeywell, IBM, Lawrence Berkeley Labs, Tandem, and Texas Instruments) and from many universities participated in the workshop (see Appendix A).

The workshop consisted of plenary sessions (invited talks one day and discussions the other two days) as well as group sessions where four working groups addressed specific issues amenable to infrastructure. Group A surveyed existing and anticipated applications of temporal databases, and gathered requirements of these applications for temporal databases. Group B considered the fairly narrow question of extending SQL-92 to support time in a comprehensive fashion. Group C considered extensions to a wider range of data models, including advanced relational models (e.g., that underlying SQL3), object-oriented data models, and active data models. Finally, Group D considered the definition of an architecture for temporal database management systems.

There was also substantial effort both before the meeting, to prepare infrastructure proposals for debate, and after the meeting, to build on the insights that emerged from the discussion. Specifically, an initial glossary of temporal database concepts and a test suite of temporal queries were distributed before the workshop. Both of these documents were amended based on the analysis and critique of the workshop. A language design committee was constituted after the workshop to develop a consensus temporal query language extension to SQL-92; this design also benefited from the discussion at the workshop.

This report documents the discussions and consensus reached at the workshop. The report reflects the conclusions reached at the workshop in June, 1993 and further discussions amongst the group participants through electronic mail. In preparing this report, each group coordinator assembled ideas and prepared an initial draft, which was then reviewed by all the workshop participants.

The record of the deliberations of these four groups, in the following four sections, forms the bulk of this report. Each of these sections begins with the group's charter and a brief snapshot of the status of the field and ends with a list of follow-on efforts. Appendices identify the workshop participants, provide the abstracts of the position papers that appear in the proceedings of the workshop, and give the original workshop call for position papers.

2 Group A: Special Requirements and Approaches

2.1 Introduction and Group Charter

Temporal databases can be used in a variety of applications. In addition to conventional applications handling administrative data of various types, other applications such as of logistics, scientific applications, and artificial intelligence present new requirements to the temporal database community.

The working group on Special Requirements and Approaches, consisting of G. Ariav, M. Baudinet, M. Boddy, C. Dyreson, M. Egenhofer, P. Hayes, F. Olken, B. Pernici, and S. Sripada, with the collaboration of occasional visitors, discussed characteristics of the different application areas, attempted to relate terminology differing from that generally used in the temporal database community, and strived to identify essential features to be provided by a temporal database system in order to be usable by these different communities as well as features and functionality for which additional research is needed.

In the following, the basic characteristics of the different application areas are briefly outlined, followed by an analysis of features to be supported by a temporal database management system (TDBMS).

2.2 Current Status of the Field

G. Ariav started the discussion with a presentation based on a set of problems from case studies illustrated in his position paper at the workshop [Ariav93]. Other application domains were also considered, specifically geographical information systems (GIS), scientific applications, and planning and scheduling applications.

The group isolated two needs not met in current TDBMS proposals.

1. *Multiple time lines*

   Traditionally, only one time line has been associated to a given entity, termed *valid time*. However, there are cases in which several time lines seems to be needed. Let us consider, for instance, a check with an associated payment date. It has at least three dates associated with it: the date in which it is issued, the date at which it can be paid, and the date at which it is actually paid. In this case, three validity times seem to be needed.

2. *An undo operation*

   In several applications there is the need of undoing previous operations. A rollback operation in this case is not appropriate, since the undo should be limited only to a given instance, not extended to all the database. Typical cases in which this operation is needed are in the domain of CAD databases.

A common feature of these applications is that temporal data are mostly imprecise and concern relative times. Ordering relationships between events are often more frequent than a precise location on the time axis of the events.

An additional common feature, is that there is a need for merging different databases, possibly defined on different time granularities.

2.3 Discussion of Glossary Terms

The term which was widely debated in this working group is *chronon*. This concept is essential in the discussed cases, since the data are on time lines that differ from those common in administrative data. For instance, scientific databases storing geological data need to store data in terms of geological eras. Distinguishing between chronons and their representation at the physical level is important. Since different application domains have such differing requirements concerning needed chronons, the user must be allowed define the appropriate chronon in the data definition language. Even within the same database, several chronons may be used for different data sets (e.g. in different relations), so appropriate support must be provided.

Another important discussion, related to the problem of chronons, is the need to distinguish between *instants* and chronons. In scientific applications, it is usual to have ordering relationships between events, rather than precise times associated to them. Different events, although not simultaneous, and for which a relative order is known, could occur within the same chronon.

2.4 Features to be Supported by a TDBMS

2.4.1 General Discussion

In general, there seems to be a gap between the goals assumed by the temporal database community and the needs discussed in the working group on special requirements.

To define the needed requirements, the first, important consideration is the definition of the boundary between the functionality to be supported by the temporal database system and that provided by applications working with the data stored in the database.

It initially seems desirable that the TDBMS provide not only basic functions, but also advanced features useful for specific applications. For instance, a classical problem found in scientific and logistic applications is the "shortest path" problem. In practice, it seems that in order to provide reasonably efficient access to data, it might be important to provide some support also for this type of computation. A similar related problem is that of providing recursive queries support in relational databases.

However, the group, decided that, at least for the short term, only basic functionality should be considered for the purpose of establishing an infrastructure, provided that the system does not become an obstacle for users to retrieve their data.

An important consideration concerns the nature of temporal data in a temporal database. The request is for particular forms of support for data defined as temporal. This support should be analogous to that provided in classical databases for predefined attribute domains. For instance, if an attribute is defined on the integer domain, it is not possible to insert characters as values. Obviously, more sophisticated types of consistency checks can be defined. Analogously, there is a need for providing true temporal attributes with appropriate consistency checks, to justify the definition of a temporal attribute. For instance, let us suppose that Name functionally determines Salary. The TDBMS should ensure that the same nameis not associated with tow different salaries at the same time. Such integrity constraints may be defined on attribute values that have been specified as valid time attributes. If the temporal attribute is treated simply as any other attribute, there seems to be no need to define it specifically as "temporal."

2.4.2 Basic Functionality

A fundamental need of most users is for support of time values at many different granularities. Appropriate operations must be supplied to perform conversions of time values between the different granularities, and to formulate queries and present results in an appropriate form.

A related feature is the need for a *merge* operation in order to be able to work with data coming from different databases (or relations) defined at different granularity levels.

Concerning times, in scientific databases and in planning and scheduling databases it is essential to provide support not only for times based on the time line (absolute times) but also for time which are relative to other times. To this purpose, the use of *time variables*, both for time points and for time entities, has been suggested (alternative name: symbolic time points/interval).

Accordingly, appropriate temporal relationships (possibly imprecise) have to be defined and supported in the TDBMS (point-point relations, interval-point relations, interval-interval relations). However, only storage support should be provided: reasoning on these times is considered outside the scope of the temporal database.

Several features related to queries have been identified.

- Support for relative times (temporally ordered attributes);
- Support for aggregation operators over time; and
- Support for temporal joins.

In our discussions, the term "relative time" was used in several senses: times specified with respect to an unanchored rather than an anchored ordering, but also times that were "variable" (e.g., we would consider "Easter" as a variable time, dependent on context for disambiguation/grounding, and "after A but before B" as a relative time specified as a position in an ordering that is unanchored to a time-line);

The issue of query result presentation is also important, in particular concerning the presentation of approximate answers, and in providing answers sorted according to a specified time dimensions, to increase readability (re-arranging in time).

2.4.3 Additional Needs

A number of possible extensions to the above basic functionalities can be considered.

- Definition of time-varying data types (with interpolation functions, with associated probabilities, and the like) and of associated operations;
- Definition of composite events;
- Associating disjoint intervals with data values; and
- Support of periodic data.

2.5 Conclusion and Follow-on Efforts

The result of this working group can be summarized by considering the following question: "Can the diverse needs of the user community be served by temporal database technology?"

Characterizing the commonalities of the user community is an enterprise doomed to fail, because users are many and their needs are diverse. Nonetheless, it is an effort we attempted in our group, gathering potential users from a variety of disciplines. Although the group was not representative of the user community as a whole, those that were present were knowledgeable potential users familiar with, by and large, database technology, and their target applications also involved time at a fundamental level. Yet these were only *potential* users of temporal database technology, primarily because of two factors.

First, no common infrastructure for temporal database research exists. This lack of common infrastructure is detrimental, not only from a research perspective, but more importantly for this group, from a pure salesmanship perspective. Users could not say what a temporal database is, nor even begin to comprehend how it could be of service to their applications. Towards this end, the glossary was important, yet at the same time confusing. The glossary was couched in the language of temporal database researchers. But as researchers in other disciplines have their own (implicit) glossaries for time related concepts; the "conceptual gap" between the two glossaries was difficult to bridge. Also, the lack of an infrastructure document led users to look for such in the glossary, but the glossary was not written nor designed for such a purpose. Consequently, basic concepts such as chronon and event remained permanently baffling, primarily because there exists no "road map" to provide users an understanding of how these terms fit together.

This raises the issue of whether the glossary should serve as a document for researchers active in the field or provide a gentle introduction to temporal databases to researchers and users from other communities. The consensus of the group seemed to be that the glossary can only be written for researchers in the field and that some other form or document should present the infrastructure and advertise the utility of temporal databases. The glossary should provide a backdrop to the infrastructure, only giving meaning to words that are unfamiliar to the reader.

The second factor is related to the first. The users in our group have developed tools to meet their needs (e.g., the geographer in our group developed a GIS). By and large, users want to "extend" their tools to include support for time values. The key characteristic of this extension is its ad-hoc nature. The tools exist and a great deal of effort and research has been invested in creating them. By and large these users are only interested in providing better support for time values or time-related processes, rather than replacing these tools with a temporal database. In order for temporal database technology to serve these users, that technology must provide a platform on which these tools can reside, without requiring substantial modification of the tools themselves. Perhaps one could characterize this need by saying that users are very strongly in favor of an "open" architecture.

Because time is considered an "add-on" in these systems, many of the issues discussed by other working groups for inclusion in the infrastructure did not emerge as user concerns. In particular, we did not discuss nor even raise questions as to whether SQL-92 or SQL3 should serve as a platform for an infrastructure query language and data model. Users have their own "high-level" query language targeted for their applications (e.g., temporal reasoner, GIS, human genome project) and any language that is capable of extracting time-related information (in a very primitive way) would suffice. This is not a criticism of the laudable goals of these other groups, only an observation that none of the users in our group currently base their tools on SQL-like interfaces or databases. The consensus TSQL2/3 effort is certainly of importance to the large community of actual database users. But almost unanimously, the users in our group were uninterested in the differences between these alternatives because they already have their own query languages and data models. It is an open question as to whether the users wanted a temporal abstract data type (ADT) or something more complex.

In essence, it is a matter of timing. The temporal database community is somewhat late to the game due to a lack of common infrastructure and working temporal databases. Consequently other players have already taken the field. If we are to have any impact on the game, as a practical matter, the question of how to integrate with existing tools is of primary importance and runs deeper than SQL-integration.

Some specific user needs did emerge, aside from the "open" architecture requirement. Let us consider relative time. Can it be supported by a temporal database? That depends on what is meant by "supported". Certainly, a temporal database can store such times and their associated constraints. Interpreting these constraints however is another matter, and lies in the sphere of general temporal reasoners rather than temporal databases. But embedding a temporal database within a temporal reasoner is exactly what the users in our group desired. Our users had temporal reasoners. They were interested in knowing whether and how their reasoners could be seamlessly coupled with a temporal database. They did not expect the temporal database to interpret the relative times, that would be done by the reasoner. At first glance, it would seem that the only requirement for a temporal database to "support" relative times is the capability of storing such times (as valid times) and passing "uninterpreted" times to a higher level. Since these times are uninterpreted, they should remain inert in determining temporal keys and normal forms.

This is not to suggest that all user requirements can be accommodated as easily; a user, say, who desires continuous times may be somewhat harder to satisfy. Rather it suggests that the functionality of a temporal database must be clearly and distinctly articulated before integration can take place. The above example places relative times outside the sphere of temporal databases (a widely-held viewpoint within the community?). The line must be drawn everywhere on exactly what is and what is not supported by a temporal database, perhaps further dividing that support into core and optional functionality. The consensus of our group was that user needs are diverse, consequently core functionality should be minimized and ease of extensibility should be maximized.

3 Group B: Extending SQL-92

3.1 Introduction and Group Charter

The working group, consisting of I. Ahn, J. Clifford, F. Grandi, C.S. Jensen, W. Käfer, K. Kulkarni, N. Lorentzos, R. Snodgrass, A. Tansel, with occasional visitors, addressed a fairly narrow but complex topic: how should SQL-92 be extended to support time in a comprehensive fashion. The ultimate goal is to produce a concrete language definition that can be used by temporal DBMS researchers as an infrastructure, incorporated into legacy (relational) DBMS products, and considered by the SQL standards committee. The (quite ambitious) goals of this working group were to put into place a structure for such a language definition, and to reduce the number of possibilities to a small set that can be further evaluated in the coming months.

3.2 Current Status of the Field

Researchers have been prolific in developing temporal data models and query languages, in an attempt to find the right tradeoffs among a set of irreconcilable constraints [MS91a]. Over the last fifteen years of work, a total of over two dozen temporal extensions of the relational data model have been proposed. Approximately half of these models support only valid time; three models support only transaction time; and the remaining seven or so support bitemporal relations. The temporal data models may be compared by asking four basic questions: how is valid time represented (alternatives include event, interval or temporal element stamping of individual attributes or tuples), how is transaction time represented (alternatives include event, interval, three events, or temporal element stamping of individual attributes, tuples, or sets of tuples), how are attribute values represented (alternatives range from atomic valued, to ordered pairs, to triplet valued, to set-triplet valued), and is the model homogeneous and coalesced (all four alternatives are represented).

As examples of the diversity of data models, we briefly examine some of those proposed by members of the working group.

- In the *Historical Database Model*, an additional, chronon-valued attribute, STATE, is part of each relation schema. A boolean attribute, EXISTS, is also added to indicate whether the particular tuple exists for that state [Cli82, CW83]. Hence, this model timestamps tuples with valid-time events.

- The *Temporal Relational Model* [LJ88, Lor88] was the first to support nested specification of timestamps using values of different granularity and to support periodic events. This model associates timestamps with individual attribute values rather than with tuples. In this approach time is treated in all respects like any other domain type. As a consequence, there is no need for time to be treated as a means to stamp data. Subsequently, there is no need to distinguish between user defined time and stamping time. Furthermore, no implicit or mandatory timestamp attributes are assumed. Timestamps are simply explicit, numeric-valued attributes, to be viewed and updated directly by the user. They represent either the chronon during which one or more attribute values are valid or a *boundary point* of the interval of validity for one or more attribute values. Since valid time is recorded in an explicit attribute, no distinction has to be made between user-defined and valid time. Several timestamp attributes of nested granularity may also be used together in a specification of a chronon. The model can also handle intervals of types other than time.

- In differentiating valid and transaction time, a four-dimensional data model was used [SA85, SA86]. Relational instances were illustrated as a sequence, stamped with individual transaction times, of three-dimensional volumes, where one of the dimensions was valid time (tuples were stamped with intervals).

- In the data model associated with TQuel, four implicit attributes were added to each relation: the transaction time of the transaction inserting the tuple, the transaction time of the transaction logically deleting the tuple, the time that the tuple started being valid in reality, and the time that the tuple stopped being valid in reality [Sno87, SGM93].

- Tansel's model ([Tan86], Tansel in [CT85]) was designed to support the calculus-based query language HQuel [TA86] and, later, the Time-by-Example language [TAO89]. The model allows only one type of object: the valid-time relation. However, four types of attributes are supported: Attributes may be either non-time-varying or time-varying, and they may be either atomic-valued or set-valued. The attributes of a relation need not be the same type, and attribute values in a given tuple need not be homogeneous. The value of a time-varying, atomic-valued attribute is represented as a triplet containing an element from the attribute's value domain and the boundary points of its interval of existence while the value of a time-varying, set-valued attribute is simply a set of such triplets. Tansel generalizes this model to nested (N1NF) relations and provides formal definitions of a temporal relational algebra and a temporal relational calculus which are shown to be equivalent [Tan91b]. Normal forms for nested temporal temporal relations have also been developed [TG89]

- The *Historical Relational Data Model* [CC87], a refinement of the model associated with a valid-time algebra (Clifford in [CT85], [Cli82]), is unique in that it associates timestamps with both individual tuples and with individual attribute values of the tuples. The data model allows two types of objects: a set of chronons, termed a *lifespan*, and a valid-time relation, where each attribute in the relation schema and each tuple in the relation is assigned a lifespan. A relation schema in the Historical Relational Data Model is an ordered four-tuple containing a set of attributes, a set of key attributes, a function that maps attributes to their lifespans, and a function that maps attributes to their value domains. A tuple is an ordered pair containing the tuple's value and its lifespan. Attributes are not atomic; rather, an attribute's value in a given tuple is a partial function from a domain of chronons onto the attribute's value domain. The domain of chronons for each such function-valued attribute is defined as the the intersection of the lifespans of the particular attribute and tuple.

- Gadia's homogeneous model [Gad88] allows two types of objects: valid-time elements [GV85] and valid-time relations. Valid-time elements, which were first proposed by Shashi Gadia, have the nice property that they are closed under union, difference, and complementation, unlike intervals.

- Gadia's multihomogeneous model [GY88] is an extension of the homogeneous model. It lifts the restriction that all attribute values in a tuple be functions on the same temporal element, in part to be able to perform Cartesian product without loss of temporal information caused by merging two timestamps into one. In this data model, temporal elements may be multi-dimensional to model different aspects of time (e.g., valid time and transaction time). Attribute values are still functions from temporal elements onto attribute value domains, but attribute values need not be functions on the same temporal element.

- The association of facts with time is implicit in the transaction-time data model DM/T [JMR91]. Instead, DM/T contains a special system-generated and maintained transaction-time relation, termed a backlog, for each user-defined transaction-time relation. This log-like backlog contains the full, timestamped change history of the associated user-defined relation. Backlog tuples, change requests, are stamped with a single transaction-time value and an attribute with values that indicate whether an insertion, deletion, or modification is requested.

- The "history-oriented" data model uses a unique key/surrogate as an object identifier and interval tuple time-stamping, with only start-time stored if versions are ordered along time [GST91].

- Clifford and Tuzhilin in [TC90] define a temporal algebra **TA** that is applicable to any temporal relational data model supporting discrete linear bounded time. This algebra has the five basic relational algebra operators extended to the temporal domain and a new operator of *linear recursion*. They prove that this algebra has the expressive power of a safe temporal calculus based on the predicate temporal logic with the **until** and **since** temporal operators, the language **TC** which was first proposed as a query language for valid-time databases in [Tuzh89]. Clifford, Croker and Tuzhilin present the calculus L_h for a temporally grouped data model which they call $M_T G$ [CCT93]. The model $M_T G$ extends the traditional relational model by defining the domains of each attribute to be functions from a temporal domain into an ordinary domain, and thus each tuple contains the entire "history" of some real-world object. L_h is a many-sorted extension of Codd's relational calculus, the extensions being the inclusion of additional sorts of variables, so that it includes variables ranging not only over ordinary values, but also over historical values and over times. Although the language L_h was not proposed as an end-user query language, a more user-friendly, SQL-like syntactic variant, similar to the HOT-SQL presented by Fabio Grandi at the workshop (a variant of HoTQuel in [GST91]), could easily be designed.

Most temporal data models are paired with a temporal query language proposal. Some two dozen temporal relational query languages have been proposed, including seven extending the relational algebra, five extending Quel, seven extending SQL, and a few being based on other formalisms. We list a few examples, proposed by members of the working group.

- The semantics of the algebra defined on the Historical Relational Data Model extends the relational operators union, difference, intersection, projection, and Cartesian product to handle lifespans directly [CC87]. Temporal variations of the joins are defined using intersection semantics, and several new time-oriented operations are introduced.

- The *Structured Query Language plus Time (SQL+T)* is based on the tuple-timestamped model, where each tuple is associated with four timestamps, to valid time and to transaction time [Ahn93]. It augments SQL with `valid`, `when`, and `as of` clauses.

- *TempSQL* is based on an attribute timestamped, homogeneous data model, where temporal elements are used as the timestamps [Gad92]. The language augments SQL with a `while` clause, in which *temporal expressions* are used to specify temporal selection.

- *TQuel* is based on a tuple timestamping bitemporal data model [Sno87, Sno93]. It is quite similar semantically to TSQL+T, and uses similar additional syntactic constructs. It is supported procedurally with an attribute-timestamped temporal algebra [MS91b].

- *HoTQuel* is based on a tuple timestamped valid-time data model, and provides two types of range variables: history variables which have histories as values and denote objects and tuple variables which have tuples as values and denote object versions [GST91]. A more recent proposal showed how history variables could be added to SQL [GST93].

- *HQuel* is based on an attributed time-stamped valid-time data model (Tansel in [CT85], [TA86, TG89, Tan91a]). The data model has four different kinds of attributes: elementary (atomic), set-valued, triplet-valued, and set triplet-valued. The first two are non time-varying; the other two are time-varying attributes. HQuel supports methods for referencing members of a set, for indicating time-stamps and value of a triplet, and for comparing attribute values. There is an equivalent algebra for this language; it includes restructuring and temporal operations in addition to regular algebraic operations. In addition to conventional algebraic operations, it includes operations to restructure temporal relations, to synchronize the time associated with attributes, and to form and decompose timestamped values. There is also an equivalent calculus for HQuel.

- *Time-by-Example (TBE)* [TAO89] is a graphical query language similar to QBE that follows hierarchically arranged subqueries of the Summary-Table-by-Example (STBE) [OO84] database query language. For query processing and optimization TBE queries are converted to equivalent algebraic operations [Tan86].

- SQL^T is based on nested relations that use attribute timestamping and allow inhomogeneous tuples [Tan93]. It augments the SELECT statement with temporal features and allows access to individual temporal values.

- *IXSQL* is based on an interval-extended relational algebra [LM93]. In addition to the five primitive operations on the conventional relational model, it supports two more relational algebra operations, new comparison operators and new scalar functions. Again, no distinction is made between user-defined and valid time, or between explicit and implicit attributes.

Support for time in conventional data base systems (e.g., [TC83, OC87]) is entirely at the level of user-defined time (i.e., attribute values drawn from a temporal domain). These implementations are limited in scope and are, in general, unsystematic in their design [Dat88, DW90]. The standards bodies (e.g., ANSI) are somewhat behind the curve, in that SQL includes no time support. Date and time support very similar to that in DB2 is included in the SQL-92 standard [MS93]. SQL-92 corrects some of the inconsistencies in the time support provided by DB2 but inherits its basic design limitations [SS92]. The SQL3 draft proposal contains no additional temporal support over SQL-92.

Many within the temporal database research community perceive that the time has come to consolidate approaches to temporal data models and algebra- and calculus-based query languages, to achieve a consensus query language and associated data model upon which future research can be based. Within the broad diversity of language and modeling constructs, common themes keep emerging. However, the community is quite fragmented, with each research project being based on a particular and different set of assumptions and approaches. Often these assumptions are not germane to the research *per se*, but are made simply because the research required a data model or query language with certain characteristics, with the particular one chosen rather arbitrarily. For example, research in query optimization must assume some data model and some query language, but the details are often not critical. It would be better in such circumstances for research projects to choose the *same* language. Unfortunately, no existing language has attracted a following large enough to become the one of choice.

3.3 Level of Language Support

The primary realization to come out of the workshop was that there were three fundamental viewpoints on how time should be incorporated into SQL. In the following, we present each of these viewpoints, along with some of their supporting arguments.

The first viewpoint argues that the SQL data model is already quite close to having the support required by temporal applications. The additional support that is necessary is primarily in the algebraic operators and to the syntax of the language. A concrete realization of this viewpoint is the IXSQL proposal, which extends SQL with a *generic interval* data type (of course, the focus here is on intervals of *time*). The data model is identical to that of SQL, with the addition of `DATEINTERVAL`. The algebra for this language is an extension of the relational algebra, retaining the traditional operators in an unmodified form, and adding two new operators. **Unfold** converts an interval into a set of time points, with the remaining attributes duplicated for each time point. **Fold** is the inverse operator. In terms of the SQL syntax, new predicates on intervals are defined, and two clauses are added, a `REFORMAT` clause, to support **Fold** and **Unfold**, and a `NORMALISE` clause, which can be simulated with the `REFORMAT` clause.

Several advantages accrue from this approach.

- Since intervals are generic, and can thus be defined over any metric, this approach naturally supports spatial and spatiotemporal databases.
- Since the extensions to SQL are minimal, especially compared with other approaches, implementation is less difficult, and acceptance by the user community may be easier to attain.
- Multiple time (and other metric) intervals may easily be incorporated.
- Every snapshot relation is also a valid valid-time relation.

The second and third viewpoints share the belief that time is a basic aspect of data, and therefore should be incorporated in a fundamental way into the data model and query language. The two viewpoints differ on the timing of the language definitions. The second viewpoint holds that, with SQL-92 an accepted standard and SQL3 being actively designed, there is no sense in extending SQL-92. Instead, efforts should be directed towards adding time to the SQL3 proposal, yielding perhaps a temporal query language standard in the 1995–1996 time frame.

Several advantages have been stated of a single SQL3 extension.

- SQL-92 is frozen, so any extensions based on SQL-92 will be rendered meaningless when SQL3 is accepted.
- SQL3 has several data modeling constructs, including object orientation, which can aid in the development of temporal extensions. For example, it allows nested relations to be simulated, thereby accommodating more temporal data models than SQL-92.
- A two-pronged approach would be difficult to coordinate, and could easily result in incompatible proposals.
- Research is active in temporal databases, with new ideas appearing all the time. It is important to do the design "right," or we will be saddled with a poor design, with no opportunity to change it (we get only one chance).

The third viewpoint favors a two-pronged approach, in which parallel efforts would consider adding time to SQL-92 and to SQL3. The rationale is that time will be added to the SQL standard only when there is implementation experience available, and that won't occur unless there is a consensus extension of SQL that admits a straightforward implementation without requiring SQL3 constructs.

Proponents of the third viewpoint counter with advantages of their approach.

- SQL-92 provides a stable basis on which to do language design; SQL3 is constantly changing.
- Designing an SQL3 extension will not impact actual applications before 1997, when the first implementations supporting SQL3 may start to appear.
- Should extension of only SQL3 proceed, there is the chance that a vendor will go ahead and implement temporal support now, in an ad hoc fashion, which the SQL3 standard will be forced to incorporate (as happened with user-defined time).
- The SQL3 standards bodies will not be interested in fundamental time support until users clamor for it, and until a commercial, relational DBMS (or perhaps two such systems) supports time.

These three viewpoints are clearly in conflict. However, each has its vocal proponents, marshaling strong technical arguments to advocate their position. The disparity between these distinct viewpoints offers one explanation as to why there has not been greater consensus in the field. Clearly, it is difficult to arrive at a single data model when there are fundamental disagreements concerning even the extent to which time should be incorporated in the model.

3.4 Desired Functionality

Much of the discussion of the working group was devoted to determining the functionality that is desired in a temporal query language. We now list the aspects discussed. We refer to an extension of SQL (-92 or 3) as *TSQL*, for convenience, keeping in mind the diversity of opinion listed in the previous section.

There have been three types of time that may be used in a temporal database: *user-defined time*, *valid time*, and *transaction time* [SA86]. User-defined time has garnered support in most commercial DBMS's (and is present in the SQL-92 standard), and transaction time is supported in some object-oriented databases (as version identifiers) and one relational database (Montage). However, the range of applications that could use valid-time support (most applications, in fact), as well as those that could use support of all three kinds of time (which is not the majority of applications, but certainly a sizable portion), dictates that a bitemporal extension of SQL is warranted.

At the same time, it is important to support legacy applications. Hence, temporal support should be optional in both the schema and in the query language. This requirement translates into the ability to specify snapshot relations (for which no temporal support is required), as well as valid-time, transaction-time, and bitemporal relations in the `CREATE TABLE` statement. It also implies that queries should be able to include multiple types of relations in the `FROM` clause, and evaluate to multiple types of relations. For example, it should be possible to compute via `SELECT` a snapshot relation from a bitemporal relation.

The extension to SQL should be upward compatible. Existing SQL queries should remain valid in TSQL. Query language reducibility is also important: an SQL query, evaluated on a temporal database as a TSQL query, should result in a temporal relation, which, when timesliced at a particular time, yields the same snapshot relation that results when this same query is evaluated as an SQL query on the timeslice of the temporal database at the same time. This property ensures that user's intuition concerning SQL will transfer over wholesale to TSQL.

Some data model proposals require that the underlying valid time domain extend only to now. Other data models support future time. (Note that transaction time, on the other hand, is never allowed to extend past now: it is impossible to know entirely accurately what will be stored in the future in the database). Planning applications require future time support; hence, it should be present in the data model for TSQL.

The controversy of discrete versus continuous time surfaced in several working group discussions, as well as the workshop plenary sessions. As this topic has been discussed for literally thousands of years by philosophers, mathematicians and physicists, it is understandable that the intricacies of this dichotomy would not be fully resolved in this workshop. Nevertheless, this working group agreed that the *representation* (as opposed to the conceptual model) should be discrete. Gio Wiederhold invoked a useful analogy of real numbers and their representations. While floating point numbers in computer programs can be conceptualized by programmers as real (i.e., continuous) numbers, their representation must necessarily be discrete. The same should hold for timestamps in TSQL's data model.

A more restricted incarnation of this issue is the distinction between open and closed intervals. An *open interval*, generally denoted as $[a,\ b)$, where a and b are timestamps, contains the time between a and b, as well as the time instant a, but *not* the time instant b. Conversely, the *closed interval* $[a,\ b]$ contains the instant b. In a discrete representation, $[a,\ b{+}1) \equiv [a,\ b]$; in a continuous model of time, there is no successor to b, and so the two are not comparable. At the representation level of TSQL, which uses discrete time, the distinction is not important. At the language level, which the user can pretend is based on continuous time, the language should support both open and closed intervals in the presentation (input and output) of temporal values.

The final issue discussed at length was that of *ungrouped* versus *grouped completeness.* These terms were presented by James Clifford at the workshop, based upon his previous research [CCT93]. In this work the authors attempt to contrast those models which employ tuple-time-stamping, which they term temporally ungrouped, and those which employ complex attribute values bearing the temporal dimension, which they term temporally grouped. After defining canonical versions of these two types of data models, called M_{TU} and M_{TG}, respectively, they present logic-based query languages for each of them and propose them as standards for measuring the expressive power of query languages for such models. They further demonstrate that the grouped models are more expressive than the ungrouped models, but define a precise, though cumbersome, technique for extending a temporally ungrouped model, by means of a group surrogate, in such a way as to extend its expressive power to that of the temporally grouped complete models. In surveying some (but by no means all) of the models that have appeared in the literature, they demonstrate the following: (i) several algebras and calculus-based query languages are ungrouped complete, (ii) the calculus L_h is, by their definition, grouped complete, and (iii) to their knowledge no algebra has been shown to be grouped complete.

While the expressive power of ungrouped complete was generally accepted as a desirable property for TSQL, there was considerable discussion concerning grouped complete. The benefit of grouped complete is that it supports a rather strong notion of the "history of an attribute," called a *history* in the Glossary. For example, one can talk about "John's salary history" as a single object, and ask to see it, or define constraints over it, etc. If the data model and query language are not grouped complete, then the salary history will be lost unless the key (here, the name) is always retained, which places a burden on the user. Ungrouped models also generally require some kind of time-invariant key to identify entities in the miniworld being modeled by the database, whereas "histories" are supported directly in grouped models without any need for time-invariant keys.

The primary concern raised by some members of the working group was one of implementability. It was pointed out that no implementation of a grouped model exists, but this was countered by the observation that few of the proposed models of any ilk have been implemented. A formal mapping of a grouped complete data model onto an ungrouped complete model, via system-maintained surrogates, has been given. However, it was pointed out that this mapping has never been implemented, and the concern was raised that implementing joins in this approach appears to some to be difficult.

The working group members fell into two camps. One position was that the lack of an existing SQL extension definition that was grouped complete, as well as the lack of any implementation experience with grouped complete data models, rendered this requirement of grouped complete too risky to incorporate into TSQL at this time. The other position held that the ultimate aim was to make life easier for the user, even if it complicated the implementation, and thus grouped complete should be a requirement of the model.

This discussion can be examined in light of the three viewpoints presented earlier. The first viewpoint, minimally adapt the data model to support time, favors neither ungrouped nor grouped complete, as both of these distort the original relational model to too great a degree. The second viewpoint, that only SQL3 should be extended, is comfortable with grouped complete. The third viewpoint, advocating definition of both TSQL2 and TSQL3, was generally comfortable with TSQL3 being grouped complete, but not so with TSQL2.

3.5 Separation of Concerns

As previously mentioned, there are now over two dozen temporal data models, each with one or more associated query languages. While such a diversity of approaches is a reflection of the excitement and ferment in the area of temporal databases, it also at some point may become counter-productive.

Focusing on data *semantics* (what is the meaning of the data stored in the data model), data *presentation* (how temporal data is displayed to the user), on data *storage* (what regular storage structures can be employed with temporal data), and on efficient *query evaluation*, has complicated the primary task of capturing the time-varying semantics. The result has been a plethora of incompatible data models and query languages, and a corresponding dearth of database design and implementation strategies that may be employed across these models.

The previously proposed data models arose from several considerations. They were all extensions of the conventional relational model that attempted to capture the time-varying semantics of both the enterprise being modeled and the state of the database. They attempted to retain the simplicity of the relational model; the tuple-timestamping models were perhaps most successful in this regard. They attempted to present all the information concerning an object in one tuple; the attribute-value timestamped models were perhaps best at that. And they attempted to ensure ease of implementation and query evaluation efficiency; the backlog representation may be advantageous here.

Most proposed models aim at being suitable for data presentation, for data storage, and for capturing the temporal semantics of data. Seen solely as means of capturing the temporal semantics, such models exhibit presentational and representational anomalies because they encode the temporal semantics in ways that are more complicated than necessary. Put differently, the time-varying semantics is obscured in the representation schemes by other considerations of presentation and implementation.

It is clear from the large number of proposed data models that meeting all goals simultaneously is a difficult, if not impossible, task. We therefore advocate a separation of concerns, i.e., adopting a very simple *conceptual* data model that captures the essential semantics of time-varying relations, but has no illusions of being suitable for presentation, storage, or query evaluation. Proposals for this conceptual data model were discussed, but a final choice was not made.

Figure 1 places the conceptual temporal data model with respect to the tasks of logical and physical database design, storage representation, query optimization, and display. As the figure shows, logical database design produces the conceptual relation schemas, which are then refined into relation schemas in some *representational* data model(s) during physical database design. The query language itself would be based on the conceptual data model. Query optimization may be performed on the logical algebra, parameterized by the cost models of the representation(s) chosen for the stored data, and in the algebra of the representational model. Finally, display presentation should be decoupled from the storage representation, and should be capable of exploiting the several existing data models having convenient display formats.

Figure 1: Interaction of Conceptual and Representational Data Models

Note that this arrangement hinges on the semantic equivalence of the various data models. It must be possible to map between the conceptual model and the various representational models. An appropriate conceptual data model would allow equivalences to be demonstrated with many of the representational models thus far proposed. This equivalence should be based on *snapshot equivalence*, which says that two relation instances are equivalent if all their snapshots, taken at all times (valid and transaction), are identical. Snapshot equivalence provides one means of comparing rather disparate representations. However, it can be demonstrated that a grouped relation can be snapshot equivalent to a large number of ungrouped relations, only one of which carries the same information content. Some argued that the notion of strong equivalence [CCT93], somewhat (but not entirely) captured by the the term "history equivalence" in the glossary, provides a more appropriate means of comparing disparate representations.

3.6 Unresolved Issues

Finally, there were a myriad of issues that were discussed briefly, or that were listed as important but not discussed.

- *Homogeneity*—In some data models, all the attributes of a tuple are defined over the same interval(s) of time, termed homogeneity. Others allow attributes to be defined over differing intervals of time, in part to permit Cartesian product.
- *Coalescing*—Some data models require that value-equivalent tuples (those with identical explicit attribute values), be coalesced if they overlap in time, that is, combined into one tuple. Others allow non-coalesced value-equivalent tuples to be present. Note that the need for coalescing only arises in ungrouped models.
- *Attribute-value versus tuple timestamping*—The field is split on this aspect, with very roughly half the data models adopting the former approach, which is viewed by some as being more "object-oriented," and half adopting the latter approach, which is viewed by some as being more efficient.
- *Levels of Nesting*—Some attribute-value timestamped models, notably Tansel's data model [Tan86], can be viewed as an extension of nested relational models. One can then ask if the nesting should be restricted to one level, or if multiple levels of nesting should be allowed.
- *Vacuuming*—When transaction time is supported, all updates, including (logical) deletion, are implemented as physical insertions, leading to ever-growing databases. So, unlike in the standard relational model, an additional notion of deletion is necessary in order to control the size and contents of the database. *Vacuuming* is this notion and denotes the flexible, *physical* deletion, consistent with the semantics of transaction time, of data in a temporal database supporting transaction time.
- *Periodic Time*—Lorentzos' data model is notable in its support for periodic events, such as an airline flight every day at a certain time [Lor88]. There has also been substantial theoretical advances in this area.
- *"Manufactured" Time*—Some advocate *intersection* semantics, where the period of validity of a tuple resulting from Cartesian product is the intersection of the period of validities of the two underlying tuples, and characterize any time outside of this time as "manufactured," and hence in some way artificial. Others feel that there are applications for which other semantics, such as union semantics, are entirely natural.
- *Restructuring* allows one to "group" on identified attributes, without being restricted to the grouping that may be imposed by the schema.

Each of these issues must be examined in the development of a conceptual temporal data model.

3.7 Conclusion and Follow-on Efforts

As mentioned in Section 3.3, there were conflicting viewpoints on a temporal extension of SQL. They can be summarized as (a) with the addition of an interval data type, there will be sufficient support in SQL2/3's data model to support applications using temporal data; (b) a two-pronged effort should be initiated, the first being a short-term effort to define a temporal extension to SQL-92 and the second being a long-term effort to define a comprehensive extension to SQL3, and (c) temporal support should be added, but only SQL3 should be extended. Whatever the approach, it was agreed that the temporal data model underlying the language be designed solely in terms of its semantic properties, with distinct and possibly multiple data models being employed for representation and presentation.

4 Group C: Advanced Relational and Non-Relational Temporal Databases

4.1 Introduction and Group Charter

The overall objective of the discussions in working group C, consisting of A. Buchmann, S. Chakravarthy, T.-S. Cheng, K. Dittrich, S.K. Gadia, T. Lawson, I.S. Mumick, M.T. Özsu, N. Pissinou, K. Ramamritham, A. Segev, M. Soo, S. Su, B. Theodoulidis, and G. Wuu, was to identify the common infrastructure for the next generation of temporal database concepts including extensions of the relational data models as well as the adoption of concepts from the semantic and object-oriented data models.

While it is true that the majority of the work on temporal databases has been in the context of the relational data model, a number of approaches based on semantic data models, such as the entity-relationship, infological and object data models, have appeared in the literature. The motivation behind all of these approaches is that the relational data model is considered to be insufficiently expressive for complex database applications such as multimedia, executive information systems, computer-aided design (CAD), computer integrated manufacturing (CIM), and geographical information systems (GIS). These applications have strong requirements to model the temporal or spatio-temporal relationships between objects. Therefore, "temporality" is an important (even if not integral) part of the next generation of database systems. Another trend that started in the 1980's is the incorporation of constraints, triggers, and rules in relational and object-oriented databases. Work in this area is concerned with active temporal databases and was considered at the group discussions.

In view of this, the overall objective of the discussions in group C was to identify a common infrastructure for the next generation of temporal databases, including extensions to the relational data model and object-based models. These issues were discussed in two subgroups, then integrated in plenary sessions of group C. Subgroup C1, consisting of S.K. Gadia, T. Lawson, M.T. Öszu, N. Pissinou, S. Su, B. Theodoulidis and G. Wuu, addressed data modeling concepts, time concepts and the incorporation of time into the next generation of temporal databases, with an emphasis on object based models. Subgroup C2, consisting of A. Buchmann, S. Chakravarthy, K. Dittrich, I.S. Mumick, K. Ramamritham, and A. Segev addressed the area of active temporal databases with particular reference to the notion of temporal rules.

The following sections elaborate on the consensus reached for the infrastructure and the open issues and future work that need to be carried out in order to complete the infrastructure for the next generation of temporal databases.

4.2 Current Status of the Field

4.2.1 Temporal Object Based Modeling

The two most prominent models that provided the basis for the development of temporal conceptual models are the entity-relationship (ER) model [Chen76] and the object based model. The ER model deals with the structural component and is founded on the notions of entity and relationship. The object model deals with both the structural and behavioral components and is founded on the notions of object, structure and behavior (method). Furthermore, a number of approaches include notions like Event-Condition-Action (ECA) rules that deal with the constraint component. Several approaches of introducing time into an object based data model were discussed. The group isolated three main approaches.

1. To extend the semantics of a preexisting snapshot model to incorporate time directly (built-in).

2. To base the new model on a snapshot model with time appearing as an additional attribute(s).

3. To move in an independent direction, developing entirely new approaches.

As examples of the diversity of data models, based on these three approaches, we briefly examine some of the proposed models:

- The *Infological data model* [LAN73, LS75], is one of the earliest of the conceptual data models that recognized time to be indeed a distinguished entity. It is based on the natural human perception of elementary facts (e-facts). Specifically, the concepts of an object, a property (or relation), and time are associated to form an atomic e-fact, which is assumed to be the "building block" of human knowledge. Storing such an e-fact in the information system results in an elementary message (e-message), which includes the references to an object, a property (attribute or relation), and a time point or period.

- The *Time-extended Entity Relationship Model (TERM)* [KLO81] focuses on a on a set of modeling primitives based on the constructs of the ER model. The notion of history structure introduced in the model, augments the basic ER constructs to create new constructs such as attribute-history and role-history.

- The *Temporal EER model* [EWK93] is an extended ER model which accommodates temporal information for entities, relationships, superclasses/subclasses and attributes.

- The *Entity-Relationship-Time (ERT) model* [TWL90, TLW91, TAL92, Theo93] is an ER-based formalism which makes a clear distinction between objects and relationships. On this basis, the ERT model accommodates the explicit modeling of time, taxonomic hierarchies and complex objects. The ERT model together with the Conceptual Rule Language and the Process Interaction Diagrams provides a uniform formalism for developing temporal database applications.

- The *ERAE data model* [DUB+86, DH87] is an attempt to extend the semantics of the ER model with a distinguished type Event as one of its basic constructs. The Conceptual Modelling Language (CML) which is based on the TAXIS model [GBM83] includes time as a primitive notion.

Other semantic data models that have addressed the introduction of time include the RM/T [CODD79], which is an extension of the traditional relational model to handle among others sequencing of events, and the Functional Data Model [SHI81].

One of the earliest attempts to introduce time into the object-oriented data model is reported in [CC88], which proposed certain terminology and concepts. Other research in extending object based models to incorporate the semantics of time or the roles of temporal objects include the following.

- The *Time In Object Databases* work [P90, P91, PM92, PM93b, PM94b, PM94c] addresses temporal problems at the finest level of granularity, viz., the object level, explores the semantics of time in the context of object databases, and identifies the temporal aspects of objects and temporal inter/intra-object relationships and changes to existing notions of temporal data that are necessary because of the transition from the relational to the object model. The work specifically involves the design and development of a model that integrates time with object databases and demonstrates how the constructs and operations of this model may be used as the basis for the stepwise refinement of other models of increasing complexity.

- *TOODM* [RS91] is a temporal object-oriented data model that supports multiple time lines and schema histories and incorporates a temporal object-oriented algebra [RS93a, RS92] and a temporal object-oriented SQL [RS93b].

- The *OSAM*/T model* model and its associated query language called OQL/T [SC91, SKL89], is based on notion of time-varying association.

- The *OODAPLEX model* [DW92, WD93] accommodates temporal data modeling through the notion of function.

- The *OOTempDBM model* [CG93] captures the temporal semantics of objects through type inheritance.

- The [KRS90] extends a complex object model by adding transaction and valid times to tuples.

- The *TIGUKAT* [GO93] model incorporates time as an abstract type. TIGUKAT models everything in the system as a first-class object. Therefore, it is only natural to model time in the same manner. TIGUKAT associates temporality with individual objects. An object is time varying (temporal) if it has at least one time varying (temporal) behavior. A behavior is time varying if it is an instance of type TemporalBehavior. Thus, temporal behaviors are represented by a type in the type lattice. Since a behavior is itself an object, temporality is built bottom up and since everything is an object, there is actually no differentiation between the attribute versioning and object versioning approaches in TIGUKAT. Applications that require the functionality of object management systems also require an extensible type system. Applications built on top of TIGUKAT may have different type semantics. Consequently, a rich and extensible set of types are provided to support various models of time (more specifically, structural models of time and the density of these structural models). Temporal constraints are introduced to represent relationships between objects, more specifically between objects in a class and those existing in their (immediate) superclass.

4.2.2 Active & Real-Time Databases

Active databases evaluate conditions and execute actions in response to event occurrences (either primitive or complex) according to the semantics of rule processing in active databases. Incorporation of active capability has typically been addressed with respect to a snapshot (i.e., non-temporal) database. A limited notion of time is used in events (e.g., temporal events and composite events) and for specifying deadlines (e.g., complete action prior to a given time). A large body of work exists on the specification of rules, its execution semantics, modeling of events, and incorporating active capability into object-oriented paradigms [AMC93, Cha92, CM93, BM91, DBB+88, DHL91, DG93, GJ91, GJS92b, GJS92a, JMS92, JMS93, MD89, SPAM91, SC93b, WF90].

Although rules have been used in temporal databases, there is no agreement on when a rule itself, used in a temporal or non-temporal database, may be considered to be temporal. A related issue is, when does a rule require temporal support for its activation? Further, rules often are not modeled in the same way as data; rules should be treated as first class objects, and so rules must be subject to the same temporal semantics as data. The working group addressed this new aspect of temporal rules in addition to defining rule structure for active databases.

Work on integrating temporal and active database features has started appearing in the literature only recently. Examples of such works include [CK93, EGS92, EGS93a, DG93, GJMS93]. Some related issues have also been discussed in the context of real-time databases [BB93, Rama93a].

4.3 Next Generation Temporal Data Modeling Concepts

The purpose of this discussion was to identify the key concepts of the next generation of temporal data models and languages. It was agreed that the next generation of temporal data models should be an extended model, rather than extensible with respect to the current generation. This implies the design of the next generation of temporal databases should not be limited to current solutions and approaches to temporal modeling, nor should be an "extensible relational approach."

An important issue, rising from the isolation of the three approaches for next generation temporal data modeling proposed in Section 4.2, concerns the role of a temporal data model. Without clarifying this issue, it is difficult to extend the object model to include temporality. The role of a temporal data model is to visualize and structure temporal data, temporal information and the temporal relationship of objects. In general, one of the main ways of structuring and visualizing temporal data is through the use of abstraction at various levels of granularity. To do so, a temporal model can be discussed in terms of three distinct parts: structures, operations and constraints. The structural component of a data model, deals with objects and their relationships while the operational/behavioral component deals with the manipulation of objects. The constraint component deals with rules for the integrity of the object structure and manipulation over time.

In line with existing data modeling design principles, the basic concepts and components identified by the group for the next generation of temporal data models were classified into three broad categories: Temporal Structural properties, Temporal Operational/behavioral properties, and Temporal Constraint properties. Temporal structural properties describe the objects of the application domain in terms of their properties and their relationships with other objects (inter/intra object relationships) with respect to time. Temporal operational properties describe the behaviour of objects over time, as reflected through changes in their properties. Finally, temporal constraint properties describe conditions the object properties must satisfy during the object lifespan. More specifically,

1. A *temporal object* is defined as a set of one or more temporal properties. The temporal properties describe structural, operational and constraint characteristics of objects over time.

2. A *temporal constraint* or rule is a database rule that includes also its validity period and is divided into three parts namely *event*, *condition* and *action* part. All these parts may refer to time points but at least the event or condition part do so in order for the rule to be characterized as a temporal rule.

Based on these definitions, and after many hours of discussions the following consensus were achieved.

- The design of the temporal query language associated with the above concepts should take into consideration the traditional language design issues such as ease of use, optimizability, expressiveness and implementability.
- Schema evolution should be supported in a way that will accommodate object persistence across schema changes. The issue of dynamic schema evolution is very important in object-oriented databases and time support can provide approaches to deal with this issue.
- The participants agreed that the modeling of time should be independent of the particular choice for the data model. This means that irrespective of the data modeling concepts, time has an ontology by itself that needs to be defined and agreed upon.
- Besides the basic infrastructure concepts of the next generation of temporal data models, it is possible to define additional constructs for the declaration of conditions/constraints (e.g., homogeneity) which may be beneficial in application development.

The next generation of temporal databases should explicitly support a rich set of time concepts. The workshop participants identified the following concepts as the minimum set of concepts to be incorporated: bitemporal interval, bitemporal span, bitemporal chronon, bitemporal element, bitemporal time point, and operations on intervals, spans and time points as well as conversion facilities between them.

Although all the above concepts are necessary for a comprehensive treatment of time in databases, the participants singled out the notions of interval, span and time point as the key concepts upon which the other concepts can be defined.

The discussion on how to incorporate time within individual models was not conclusive. The participants felt that in the first instance, the infrastructure should identify concepts rather than techniques. The identification of concepts should point out what needs to be expressed explicitly in terms of time within an object model. The particular technique to be followed cannot be agreed within the community at this stage. Consequently, it "does not matter" how one includes time as long as there is the capability to incorporate it.

4.4 Active and Temporal Database Concepts

To reach consensus on the subject of Active Temporal Databases, there first has to be a shared understanding of the structure of rules and the definition of event. Accordingly, we first provide consensus definitions of active rules and events (in the context of rule definition). We limit the discussion to active rules (ignoring, for instance, deductive rules).

An active rule is an Event-Condition-Action (E-C-A) rule, where

Event E: is a *basic* or *composite* event, as defined below.

Condition C: is either (1) a boolean expression, or (2) a query in the database query language that results in a TRUE/FALSE answer. The query must be side-effect free.

Action A: is an execution of a database operation or an arbitrary application program.

Definition 4.1 Basic Event: a pair *(event occurrence, time instant)*. The event occurrence is represented by some symbol e and is mapped to a time instant t on the system clock. The basic event (e, t) is said to occur at time t. □

It should be noted that an event occurs at a point in time. Examples of basic events include `begin transaction`, `after commit`, and `before read`. In fact, basic events can be obtained from most database operations by adding the modifiers `before` or `after` to the name of the database operation. External signals, and time events, such as *11:00am*, are permitted as basic events. A time event is represented by the pair *(time name, time instant)*, where time name is the symbol representing the event occurrence. `(''11:00am'', 11:00am on July 20, 1993)` is an example of a basic event.

Events may have attributes. For instance, an `after insert` event has information regarding the specific relation updated as well as the specific tuple inserted. Such information is an attribute of the event. In addition, event attributes may include system-level information such as transaction id, user id, and time.

Definition 4.2 Composite Event: can be created from basic and other composite events through the use of a closed algebra. A composite event occurs at a time instant, as specified by the closed algebra in terms of the time instants of component basic events. □

Several algebras for composite events have been proposed, e.g., Snoop [CM93], SAMOS [DG93], [Chom92], and ODE [GJS92b, GJS92a, JMS92, GJMS93]. We permit simple conditions, such as `X > 10` on attribute `X` of an event, or a boolean predicate on attributes of events, to be included in the algebra for composite events. Note that the boolean predicates in a composite event do not refer to items stored in the database, and can be evaluated from the given event, without querying the database. Permitting boolean expressions allows for easy specification and efficient implementation of events such as

`every` n^{th} `trade of IBM stock at price > 50`

that would otherwise require complex temporal queries in the *Condition* part. A more complex algebra may also permit predicates that refer to database items, We assume that an optimizer that can move such complex predicates into the condition part would be provided, thereby making the algebra equivalent to the one we consider.

Both basic and composite events are usually referred to by event names or event identifiers. Depending upon the event description, and the type of database, a rule may get associated with one or more relations, views, or objects. Rules can typically be inserted, deleted, updated, activated, and deactivated by the user as well as by the system.

To summarize, for the purpose of this report, we will assume that the event part of a rule is based on an algebra, that the algebra will permit certain conditions to be included in the event part, and that there will be a separate *Condition* part in the rule. For high-level syntax, we will express a rule as "`WHEN` event `IF` condition `THEN` action"; different syntax may be used and defaults assumed in actual implementations. It is desirable of course to standardize on a rule language.

4.5 Temporal Rules

Following the above definitions of rules and events, we focus on the definition of temporal rules, and distinguish between two cases: temporal rules in non-temporal databases and temporal rules in temporal databases.

Definition 4.3 Temporal Active Rule: An active rule is said to be temporal if (1) the event is a composite event that refers to basic events occurring at time points other than the time when the rule is fired, or (2) the event refers to explicit time basic events, or (3) the condition contains a temporal database query that cannot be expressed in a non-temporal query language that can reference the basic event (or the last basic event in the composite event that caused the rule to fire), operating over a database that does not maintain a temporal history. □

Note that the action is not mentioned in the definition of a temporal active rule. The action is an arbitrary procedure, and we will not attempt to characterize a rule based on the behaviour of the action.

In the above definition of a temporal active rule, conditions (1) and (2) may be seen as the definitions of a *temporal event*, and condition (3) as the definition of a *temporal condition*.

4.5.1 Temporal Rules in a Non-temporal Database

In a non-temporal database, the query language is non-temporal, so the condition of a rule cannot contain a temporal query. Hence, active rules in a non-temporal database are temporal if and only if (1) the event is a composite event that refers to basic events occurring at time points other than the time when the rule is fired, or (2) the event refers to explicit time basic events.

We consider both these cases:

Composite events ⇒ Temporal Rules Composite event algebras enable one to relate basic events occurring at different points in time. One can specify simple patterns of such events that are of interest, in much the same way as a temporal query can specify patterns of values in successive versions of relations. Composite events thus represent simple forms of temporal queries, and can provide simple temporal features. Active rules using such event algebras must therefore be considered to be *temporal rules*. The composite algebras can be used in non-temporal databases, provided mechanisms to recognize these event patterns are provided [CM93, DG93, Chom92, GJS92a, JMS92]. While we will not discuss any particular algebra in this paper, we will illustrate their relationship to temporal databases through a representative syntax.

EXAMPLE 4.1 Consider an inventory database in a store. There is an `inventory(item, amount)` relation storing the amount of each item in stock. Further, at the end of every month, sales statistics for the month are computed. One of the statistics is the average price and quantity sold for each item in the store during the last month.

We want to label an item as *high-tech* if it sold in low quantities and at high prices for three consecutive months some time in the past, and has been selling in high quantities and at low prices for the last three consecutive months. Clearly, in a temporal database, a temporal query can be used to identify the *high-tech* items. We show how a composite algebra can be used to define a temporal rule that labels items as *high-tech*.

Let u be an item carried by the store, and let `u_sale(Q, P)` be an event representing the insertion of the monthly statistic "item u sold in quantity Q at average price P during the last month".

We first define derived events `u_losale` and `u_hisale` that represent the facts that (1) item u sold in low quantities at high prices, and (2) item u sold in high quantities at low prices, respectively, in the last month. Assume `LO_QTY`, `LO_PRICE`, `HI_QTY`, and `HI_PRICE` are system constants defined elsewhere.

```
(r1 ):  #define u_losale = u_sale(Q, P) && Q < LO_QTY&&  P > HI_PRICE;
        #define u_hisale = u_sale(Q, P) && Q > HI_QTY && P < LO_PRICE ;
```

Suppose that, when we identify an item to be *high-tech*, we want to check if its current stock is greater than `HI_QTY`, and if not, we want to place an order for an amount = `HI_QTY` less the currently stocked quantity of the item. One may chose to write this as follows (the syntax below is for purpose of illustration only, we are not promoting this syntax. For instance, in a real language, one would have higher order constructs to represent repetition).

```
(r2 ):  WHEN sequence(u_losale, u_losale, u_losale)
             followedby
             sequence(u_hisale, u_hisale, u_hisale)
        IF (SELECT amount
            FROM inventory
            WHERE item = 'u' AND amount < HI_QTY)
        THEN order('u', (HI_QTY - amount));
```

The WHEN part of the active rule uses two composite algebra operators: `sequence` and `followedby`. `sequence(u_losale, u_losale, u_losale)` is a composite event that occurs when a sequence of three consecutive `u_losale` events occur, at the point when the last `u_losale` event in the sequence occurs. The composite event (a `followedby` b) occurs if the event b occurs sometime after event a has occurred, at the point in time when event b occurs. So, the composite event in the WHEN clause occurs at the point in time when the third month's high sale figure is reported, and at some time in the past, three consecutive low sale figures were posted. The IF part of the rule is a condition that checks if the given SQL query returns a nonempty answer. In case it does, an order is placed for the difference between `HI_QTY` and the `amount` returned by the SQL query. □

The active rule $r2$ is considered to be a temporal rule because it can be mapped to an equivalent rule where the WHEN clause contains basic events, and the IF clause contains a temporal query, providing such a query was permissible in the system. For example, if the WHEN part was limited to basic events, then it would contain the event `u_hisale`, and the IF part would need a temporal query that refers to old versions of a sales relation. We assume here that the sales relation only keeps the average price and total quantities for the last one month. Since an active rule that has a temporal query in the condition part and a basic event in the event part would definitely be called a temporal rule, we must also consider the equivalent rules of type $r2$ as temporal.

One may want to enhance Example 4.1 to:

1. Require that the sequence of `u_losale` events be followed by the sequence of `u_hisale` events within one year.
2. Check whether the current inventory amount is less than the sum of the last two months sales, and if so, to order the difference between the sum of the last two months sales and the current amount in the inventory.
3. Rather than writing a separate rule for each item that one wants to track in a similar fashion, write one active rule to track all such items.

These enhancements require that events have attributes, and that attributes of events be passed across time, to other events, and into the condition and action part [GJMS93]. The resulting rule may be written as follows.

EXAMPLE 4.2 Let `sale(I, Q, P)` be an event representing the insertion of the monthly statistic "item `I` sold in quantity `Q` at an average price of `P` during the last month".

The derived events `losale(I, T)` and `hisale(I, Q, T)` have attributes, and represent the facts that (1) item `I` sold in low quantities at high prices in month `T`, and (2) item `I` sold in high quantity `Q` at low prices in month `T`, respectively.

```
(r3 ):  #define losale(I, T) = sale(I, Q, P) && Q < LO_QTY &&  P > HI_PRICE && T = time();
        #define hisale(I, Q, T) = sale(I, Q, P) && Q > HI_QTY && P < LO_PRICE && T = time();
```

`time()` is a function that returns the current time at time of invocation. The `time()` function is invoked at the time of occurrence of event `sale(I, Q, P)`, and thus captures the month when the sales figures are posted.

The active rule may now be written as follows.

```
(r4 ):  WHEN sequence(losale(I, T1), losale(I, _), losale(I, _))
             followedby
             sequence(hisale(I, _, _), hisale(I, Q1, _), hisale(I, Q2, T2))
             && (T2 - T1 ≤ 12)
        IF (SELECT amount
            FROM inventory
            WHERE item = I AND (amount < (Q1+Q2)))
        THEN order(I, (Q1 + Q2 - amount)) ;
```

The symbol '_' is used for an argument position whose value is not important in specifying the rule. The WHEN part of the active rule includes a simple condition `(T2 - T1` $\leq$ `12)` to ensure that the current hisale event is within 12 months of the first losale event being considered to satisfy the composite event. Using the same variable `I` in all events ensures that all sales events are about the same item. The item `I` and the quantities `Q1` and `Q2` are then available, and used, in the condition and action part. □

Explicit Time Events ⇒ Temporal Rules The event can contain explicit reference to times, such as at "11:00am on Jan 26, 1950". A *calendar algebra* can be defined to refer to time events at a higher level, such as "3rd Friday of a month". In either of the above cases, we say that the event refers to basic time events, so the active rule is temporal. Such basic time events can be used like any other basic events in defining composite events using a composite algebra.

An example of a temporal rule using basic time events is, "`WHEN every 10 Minutes IF` condition `THEN Evaluate(Portfolio)`" where `Evaluate` is a user-defined procedure that is applied to the named object (condition may be any condition on the database states).

In many business applications, there is a need to reference times which are not standard Gregorian dates, e.g. "business days" or "trading days". Several proposals have been introduced recently of a database support for different calendric systems [SSD92, CS93, CSS93]. In the context of this position paper, a calendar is a semantic collection of time interval possibly with a cyclical structures. Calendars can be operated upon via an algebra (calendar expressions) and the result is also a calendar. The interpretation of the expression is a function of the operand calendars. So "`Every Fri in 1993`" will yield a different result if applied to the Gregorian calendar than if applied to the business days calendars. The collection of those Fridays is also a calendar. There is one base calendar (e.g., the Unix time system) to which all other defined calendars are mapped. It should be noted that such an algebra is more powerful than functions (such as DATE functions).

We use Chandra's calendar algebra [CSS93] to illustrate the idea (a different language could be used). Assume that the calendars `WEEKS`, `DAYS`, `Expiration_Month`, and `AM_BUS_DAYS` (American Business Days) were already defined. The following code specifies a basic time event that can be included as the event of a rule associated with trading stock options; it specifies "third Friday of the expiration month if a business day, else the preceding business day".

```
{
        Fridays = [5]/DAYS:during:WEEKS;
```

```
        temp1   = [3]/Fridays:overlaps:Expiration_Month;
        /* 3rd Friday of the expiration month where expiration
           month is a predefined calendar */

        if (temp1:intersects:holidays) /* if holiday */

                return([n]/AM_BUS_DAYS:<:temp1);
                /* last business day before 3rd friday of expiration month */

        else
                return(temp1);
}
```

The above example illustrates that calendars can be defined as part of the event specification and previously defined calendars can also be referenced. If such an algebra is used it has to be part of the event composition algebra mentioned before. Incorporating such calendars expressions into rules require the design of parsers for such scripts that create an efficient evaluation plan. Chandra also describes the implementation of such temporal rules in POSTGRES [CSS93].

4.5.2 Temporal Rules in Temporal Databases

From the definition of temporal rules, it follows that an active rule in a temporal database is nontemporal if the event is basic and the condition contains a non-temporal query that could be expressed in a non-temporal query language. All other rules in a temporal database are called *temporal rules*. Thus, rules that would be considered temporal in a non-temporal database, are also considered temporal in a temporal database.

Further, both temporal and non-temporal active rules can be be viewed as first class database objects. This means that the history of rules should be kept. Each rule is associated with transaction and valid times. Transaction time is the time when the rule was recorded in the database. Valid time represent the time point(s) when the rule is applicable, i.e., checked for activation. The valid time of a rule can be specified explicitly as a temporal element, or implicitly in terms of data condition(s) or the occurrence of some event(s). Activation and deactivation of rules is achieved by changing their valid time.

There are two basic alternatives for modeling the history of rules. In the first way, the rule is considered as a single unit, and thus, a change to one of the components is regarded as deletion (note: in the temporal database case, deletion of a rule amounts to indefinite deactivation) of the rule and the addition of a new rule. In the second way, a rule is considered to be a complex object and the history of the individual components is maintained, that is, we can represent different versions of the same rule.

4.5.3 Actions of Rules in Temporal Databases

When a rule is activated, and the condition evaluates to true, the action part of the rule gets executed. In a temporal database, the action can include any update to the database, including updates to past or future valid times of data items. Such updates are called retro-active and pro-active updates, respectively.

If proactive or retroactive updates are allowed in a temporal database, rules can effect data in several ways. We will elaborate on the retroactive case only; dealing with proactive updates is similar. In [EGS93a] the following characterization of retroactive effects is given. In the following definitions, "past" is measured relative to the time of the operation or triggering of a rule:

Definition 4.4 Retroactive Update: an update operation that modifies past values of data elements. □

Definition 4.5 A Retroactive Rule: a rule whose action includes a retroactive update. □

Definition 4.6 Retroactive Rule Activation: the application of a rule to past snapshots. □

These definitions indicate that rules can effect data *retroactively* in two main ways: due to retroactive rules, or due to retroactive activation of rules. The latter case can occur for two reasons: 1) a rule is introduced in the system with a valid time that includes past time interval(s), or 2) a retroactive update occurred, and the updated data element(s) trigger a rule which was valid at that past time.

4.6 Temporal Consistency

Generally speaking, the consistency of a database is measured relative to the effect of a serial execution of a set of transactions on a state that is assumed to be consistent (i.e., the serializability condition), and relative to a set of constraints that limit the space of legal database states. In the rest of this section we assume that the serializability condition is satisfied, and therefore, consistency is in the context of constraints only. Constraints can be non-temporal, i.e., they refer to any valid time snapshot, or temporal, i.e., they refer to particular snapshot(s). It is assumed that constraints can be compiled into rules that enforce them. Note that these rules can also derive data items.

In order to characterize the actions of rules in a temporal database, there is a need to distinguish between a database state and a snapshot. The following definitions refer to a *bitemporal* database, i.e., a database that supports both transaction and valid times. We use the term *system time* to refer to the time values generated by the system clock. It is assumed that these time values are used as the domain of transaction time. *Observation time* refers to the reference point in the system time line from which the database state is observed. In conventional databases the observation point is always NOW. In temporal databases the observation point can be less than or equal to NOW. Only data objects with transaction time less than the observation time can be seen by a query (or a transaction).

We define a few concepts needed to understand temporal consistency.

Definition 4.7 Database State(t): all the values of data objects committed by system time t. □

Since we assume no overwriting of data, each state contains the complete database evolution up to time t. Moreover, the history of database states is kept as well, and therefore is a history of histories (or a sequence of sequences). Note that database states are ordered by system time.

EXAMPLE 4.3 Assume that the values of data object A are changed by transactions in the following way (the values of A are denoted as a_1, a_2, etc. a_i is the new value inserted by transaction TR_i):

Transaction	TR_1	TR_2	TR_3
Value	a_1	a_2	a_3
Transaction Time	1	3	11
Valid Time	$[1, 4]$	$[3, 10]$	$[9, 15]$

The database state at time 2 contains the information in the first column, at time 5 the information in the first two columns, etc. □

Definition 4.8 Transaction Time Database Snapshot(t): the database state at system time t. It is assumed that this snapshot is the same for any observation time greater than t. □

Note that Transaction Time Database Snapshot is the same as the database state at the specified time. The reason the two definitions are given is that those two terms are used by many people with different meaning. It is convenient in some contexts to use the term database state and in others to use transaction time snapshot. However, these are synonyms.

Definition 4.9 Valid Time Database Snapshot(t_1, t_2): the world's state (as inferred from the database states) at valid time t_1 as observed from system time t_2. t_1 can be either greater than or less than or equal to t_2 (greater than implies that the data values are predictions). □

For valid time snaphots, t_2 is necessary for the following reasons. At the presence of retroactive or proactive updates, a snapshot characterization requires the specification of an observation point, i.e., the snapshot values can be different for different observation points. In the above example, the database snapshot at valid time 2 will give the value v_1; at valid time 4, however, there are two values, v_1 and v_2 inserted by transactions TR_1 and TR_2 respectively. If we assume that the later commit is the right value, the snapshot value will be v_2. There are situations where the snapshot value has to be determined based on more complex inference [EGS92]. It should be noted that also without retroactive or proactive updates t_2 is necessary (of course one can default it to NOW); for example, if A had a value a with valid and transaction times 2, then a valid time snapshot requested at system time t_2 for valid time 2 may have different values for different observation times t_2; the value corresponding to $t_2 = 4$ is a, while the value corresponding to $t_2 = 1$ is "unknown."

Thus the value of a data object at a given valid time is a function of the observation time. Consequently, the consistency of the database has to be determined relative to a chosen observation time.

Definition 4.10 Temporal Consistency at Time t: An active temporal database is consistent at system time t if for all valid time instants t_v, a valid time snapshot at time t_v as seen from time t (which will include all the data objects whose valid time intervals include t_v when observed from time t) satisfies all the rules that are valid at time t_v. □

Note that if rules can also change retroactively, then determining which rule is valid at t_v in the above definition is not trivial. That is, when observing the database state from observation time t, then for a particular t_v and a data object D, there may be two or more rules which constrain D (possibly in a contradictory way). For example, at time $t = 1$ the constraint $D < 10$ was inserted with valid time interval [1,5]; this constraint was changed to $D < 8$ at $t = 5$ with valid time [3,10]; then at $t = 10$ a transaction changed the value of D at valid time 4 from 7 to 9. How should consistency be enforced? if we evaluate it against the rules that existed at time 4 as we see them from observation time 10, then two rules exist and if we assume that the latest is correct then the new value of D should be rejected. However, if the consistency is evaluated relative to the rules that were valid at time 4 as we saw them at *that* time, then the new value should be accepted.

Enforcing consistency is necessary when new data values are modified, and when rules are activated. The complexity of the task is a function of which updates and rules are allowed. A classification of temporal active database systems based on that level of complexity is useful. Factors that will affect it include the following.

- Are retroactive/proactive updates to data permissible?

- Are retroactive/proactive updates to rules
- Are retroactive/proactive rules permissible?
- Is retroactive/proactive activation of rules permissible?

4.7 Real Time Constraints

The ECA model allows one to capture the condition corresponding to the lack of completion by a deadline but not much more. Consider the rule, "`At` deadline `if` A not completed `do` recovery action." There is no way to say "complete A within deadline" so that the system tries proactively to meet the deadline and only if it fails it takes recovery action.

It is not difficult to see that active databases provide a good model for the *arrival* (i.e., triggering) of periodic/aperiodic activities based on events and conditions. Even though the ECA model implies that an active database can be made to react to timeouts, time constraints are not *explicitly* considered by the underlying transaction processing mechanism.

However, the primary goal of real-time database systems is to *complete* transactions on time [Rama93b]. One can thus state the main deficiency in active databases in relation to what is required for them to deal with time constraints on the completion of transactions: time constraints must be *actively* taken into consideration.

Consider a system that controls the landing of an aircraft. Ideally, we would like to ensure that once the decision is made to prepare for landing, necessary steps, for example, to lower the wheels, to begin deceleration, and to reduce altitude, are completed within a given duration, say 10 seconds. Here the steps may depend on the landing path, the constraints specific to the airport, and the type of aircraft, and hence may involve access to a database containing the relevant information. In those situations where the necessary steps have not been completed in time, we would like to abort the landing within a given deadline, say within 5 seconds; the abort must be *completed* within the deadline, presumably because that is the "cushion" available to the system to take alternative actions. This requirement can be expressed as follows.

```
ON (10 seconds after initiating landing preparations)
  IF (steps not completed)
    DO (within 5 seconds Abort landing).
```

Thus, while active databases possess the necessary features to deal with many aspects of real-time database systems, the crucial missing ingredient is the active pursuit of the timely processing of actions.

4.8 Conclusion and Follow-on Efforts

The overall objective of the discussions in group C was to identify the common infrastructure for the next generation of temporal database concepts including extensions of the relational data models as well as the adoption of concepts from object based data models. The emphasis of the discussions was on object based models since the participants felt that this is the most likely way forward. Research work in the areas of temporal object bases and temporal active databases is quite preliminary and consequently, the infrastructure is less well developed here as compared with that for temporal relational databases.

The participants of group C1 discussed in some detail the future directions of the work in this area. Although, it is too early to consolidate, the participants felt that any future work on infrastructure should be linked with work on SQL3. The main reason for that is that SQL3 is still open for negotiation and in addition, incorporation of time semantics into it will certainly have a major impact in the community unlike the work in temporal relational databases and extensions of SQL-89 and SQL-92.

Future work in this area should also be linked with the findings and conclusions reported in the next section, in order to provide the required performance levels necessary for the wide acceptability of temporal object database technology. This link was not investigated in any detail during this workshop but it will certainly be a major issue for discussion at a future infrastructure workshop.

Finally, work on the glossary should incorporate concepts from any proposed extensions to SQL3 and agreed infrastructure for temporal object databases.

5 Group D: Implementation

5.1 Introduction and Group Charter

Working group D, consisting of J. Blakeley, R. Elmasri, S. Jajodia, V. Kouramajian, K. Makki, D. Peuquet, V. Tsotras, and D. Wells, was concerned with the definition of system implementation techniques and an architecture for temporal databases. The identification of any similarities and differences between an architecture for a temporal DBMS and that for a non-temporal DBMS was one of our goals. Proposing a reference architecture was one of the goals of the group.

A second goal was to identify a suitable model, or models, for representing temporal databases at the storage level. Such a model would be needed for discussions on several system modules, such as query optimization, it was argued. As it turned out, our discussions led us to change this opinion.

A third goal was to identify which aspects of a temporal database, if any, need to be identifiable at the storage level; for example, whether such time dimensions as valid and transaction time would need to be explicitly represented at the storage level.

The fourth goal was to identify a number of system modules (query optimization, concurrency control, security, etc.) and to determine preliminary requirements for each of these modules.

Finally, we wanted to discuss what are typically temporal database applications in preparation for the specification of performance benchmarks. These benchmarks would be used to compare proposed indexing and storage structures for temporal databases.

5.2 Current Status of the Field

Only a few generalized temporal database management systems have been implemented. The TQuel prototype [AS86] is perhaps the best known. However, because many applications of databases are inherently temporal, there have been countless implementations of ad-hoc temporal databases that either utilize existing commercial DBMSs or that build temporal databases over file systems. In these implementations, the meaning and interpretation of time is implemented by the user application programs rather than being understood by the DBMS software itself, which is the goal of a generalized temporal DBMS.

A number of indexing techniques have been proposed that claim to improve the performance of search based on temporal conditions. Some are extensions of techniques that were originally proposed for spatial indexing, whereas others were explicitly designed for temporal databases.

A crucial aspect of a temporal DBMS architecture is to define a standard algebra that is well accepted for temporal database operations. This would correspond to the well-accepted relational algebra operations for representing non-temporal database requests. In addition, a set of update operations for temporal databases would be useful. These operations would be the target internal representation for temporal queries and updates, and would serve as a basis for such system modules as query processing and optimization. Unfortunately, there is no well-accepted temporal database algebra (there is, however, no shortage of candidates [MS91a]).

There has been little research in areas such as identifying concurrency control, recovery, security, and other techniques that would take advantage of temporal database features, such as the availability of the history of database changes.

5.3 Baseline Architecture

After heated discussion, there was general agreement that the architecture of a temporal DBMS should not differ drastically from that for a non-temporal DBMS. In particular, it seems that at the physical storage level (i.e. disk pages), data can be stored as byte streams as for non-temporal databases. However, we did not have enough time to discuss the impact of time on concurrency control, recovery, and security mechanisms. We mainly were considering the system modules for query processing and optimization.

Our baseline architecture consists of four main modules. At the lowest level, a storage system exists, which stores persistent data in disk pages. This level could use existing storage systems, such as EXODUS or the UNIX file system. Stored objects are retrieved as byte streams, and interpreted as objects at a higher level of the system based on the information stored in the system catalog. We could not agree whether the basic storage model should be based on tuple versioning, attribute versioning, the use of deltas, or a combination of these techniques. At the storage level, data records can be clustered for efficient access. For example, the partitioning of storage into a current store and history store could be used. Indexes to locate related data or to search based on time conditions, attribute values, or a combination of both, could be built.

Above the storage level, there would be a number of higher level modules. One module would include an extensible library of available execution algorithms is proposed. Another module that contains a library of available index structures would be accessible by some of the execution algorithms. The execution algorithms would be various implementations of the high-level temporal database operations. A query optimization module would create an execution strategy for a temporal query by choosing the appropriate options from the execution algorithms library. With reference to the conceptual architecture shown in Figure 1, these two lower levels (storage system, query optimization and evaluation) correspond to the representational side of that figure.

Standard modules such as query parser would create an internal query representation, which would then be optimized by the query optimizer (query parsing and logical query optimization corresponds to the right middle of Figure 1, concerned with the (single) conceptual data model on which the temporal query language is based.

The issue of which set of formal operations to use in representing and optimizing temporal queries was not resolved. There were two main points of view. The first was that we should extend the standard relational algebra operations with the interval algebra [All83] so that we can proceed with prototype implementations and analysis of various optimization methods, indexing techniques, and execution algorithms. The second point of view was that we do not fully understand how temporal databases differ from non-temporal ones, and that we should examine new algebras developed explicitly for temporal databases. The conclusion was that we should proceed in both directions, with some researchers taking the first shorter-term approach, while others pursue a possible long-term better solution.

The indexing module should be extensible. Thus, new indexing methods would be added to the library as they become implemented. For each indexing method, the specification of the storage model that it is compatible with, as well as the execution modules that can use it, and cost estimate functions for use by the optimizer, must be given when it is added to the index library.

5.4 Performance Benchmarks

Performance benchmarks to compare various proposals for temporal index structures and search techniques are needed. We agreed that there is probably no typical temporal database application, so that it would be necessary to create a number of benchmarks for different applications. The following characteristics should be considered when designing a temporal benchmark.

- *Database size (volume of data):* This could be identified by the average number of snapshot objects, and the total number of stored objects.
- *Frequency of updates:* The parameters that could specify this include the average version interval (lifespan), average number of versions per (non-temporal) object, percentage of very longed lived objects, and other parameters.
- *The purging or archiving characteristics of the application.*
- *The presence of retrospective updates.*
- *Query characteristics:* These include whether the prevailing queries are purely temporal, purely attribute based, or a combination. For temporal queries, a further distinction into point versus interval queries is possible. An initial matrix to characterize query characteristics at a low level was proposed.

The metrics to be measured by a benchmark include the space consumption by indexing and storage structures, the update time, the archiving/migration time, and access times for different types of queries.

5.5 Extensible Query Processing Architecture

We further discussed the query processing architecture for temporal databases and agreed that it should support extensibility at various levels. At the algebra level, the architecture should support the use of different algebras; for example, a relational algebra extended with Allen's interval algebra, or one of the many algebras proposed by temporal database researchers. The set of execution algorithms in the execution algorithms library should also be extensible. New search techniques can be incorporated by adding their implementations and descriptions to the library, along with cost estimation formulas to be used by the optimizer. The optimization algorithms themselves may be changed by basing them on different paradigms, such as dynamic programming or branch and bound.

This organization is consistent with the trend towards open architectures.

5.6 Conclusion and Follow-on Efforts

In summary, our group recommends that the architecture for temporal databases be based on similar architectures for non-temporal databases. The emphasis is on flexibility so that various query optimizers, execution algorithms, index techniques, and storage models could be supported. We recommend two levels of future investigation: short term and long term.

For the short term, we should examine in more detail the suggested framework for temporal query processing and optimization. The proposed system modules and their interactions should be further specified. We recommend that an algebra for internal representation of temporal queries be developed based on extending the existing relational algebra with temporal operations. Research to optimize temporal queries based on this algebra would then proceed. A library of execution algorithms and a library of indexing methods that can be used to implement the operations of this algebra would be needed. The characterization of a number of benchmarks for various temporal database applications is needed to be able to compare various optimization techniques and indexing methods.

For the longer term, we recommend that research continue in identifying whether or not there are more significant differences between temporal and non-temporal databases. The use of proposed temporal algebras that are more independent from the relational algebra as a basis for system implementation should be investigated. The impact of temporal databases on concurrency control, recovery, security, and other system modules should be investigated.

6 Conclusions

This workshop was the first opportunity for those active in temporal databases to meet and discuss the common aspects of extant ideas and proposals. The workshop was unusual in that the topics of discussion were not new results of research, nor recommendations for future research, but rather which results of previous research could be identified as common infrastructure.

The preceding four sections each enumerate specific contributions to an infrastructure for temporal databases. There were several common threads that ran through many of these individual group discussions.

- *The infrastructure must be based on a core set of desired features, so that most temporal applications receive at least some support from the temporal DBMS.*

  Applications demand a wide variety of temporal database features, from storage of timestamps through temporal joins through support for relative time (in which only the relative ordering of events is known), through full-fledged temporal reasoning. Because of this diversity of requirements, the infrastructure should only include those aspects that support a significant fraction of the applications, and that are fairly well understood.

- *Terminology is critical.*

  As time is such a prevalent aspect of data, and indeed of life in general, it is natural that different spheres of activity would come up with different terms for the same concept (e.g., an airplane trip from New York to Paris is a "macro-event" to some and an "interval" to others) and identical terms for different concepts (e.g., an "event" to some is simply a position on a time line, whereas to others it is an occurrence of something interesting). Much effort was invested to develop a well-defined glossary of relevant terms.

- *Aspects of the conceptual model must be separated from concerns of the representation.*

  This separation proved to be beneficial in several of the discussions: it enabled the issue of performance to be separated from the issue of semantic integrity. Particularly in databases, performance is seen as all-important, with other issues subjugated to a lesser status. Several of the groups made explicit distinction between the semantics of the data, as expressed in the conceptual model, from the encoding of the data, as expressed in a representational model.

- *The baseline architecture must be extensible, and must identify what is different about a temporal DBMS, and what can vary between TDBMS implementations.*

  The distinction of conceptual versus representational is incorporated into the architecture. Extensibility of the storage model and index library ensures that different representational models can be employed, thereby achieving high performance through the use of storage models and temporal indexes appropriate to the application.

In addition to this report, several other components of an infrastructure for temporal databases have recently been completed.

Substantial effort over the two years preceding the workshop generated an initial glossary that was published in the *ACM SIGMOD Record* [JCG+92], and its impact on standardizing terminology is now being felt. Christian S. Jensen headed an editorial board to complete the glossary. The glossary, containing 87 terms and their definitions, will appear in the March, 1994 issue of the *ACM SIGMOD Record*.

Also, over the six months prior to the workshop a fairly exhausting consensus effort generated an initial draft of a "language benchmark" intended to be an aid in evaluating the user-friendliness of proposals for temporal query languages. Christian S. Jensen spearheaded the effort to complete "test suite of temporal query languages", as it is now called. This document is focused on SQL language extensions.

Several other efforts contemporaneous with the planning of the workshop also contribute to an infrastructure for temporal databases. The first book on temporal databases [TCG+93] is a comprehensive volume covering modeling, languages, and implementation aspects of temporal databases. The book consists of 23 chapters that report the research results of leading researchers in temporal databases. The fifth in a series of bibliographies on temporal databases appeared in the December, 1993 issue of *ACM SIGMOD Record*, a bibliography on spatiotemporal databases appeared in the March, 1993 issue of *ACM SIGMOD Record*, and an extended version will appear in the *International Journal of Geographical Information Systems*.

Several consensus efforts were started as a result of the discussions at the workshop. The glossary is continuing, and new terms will be added as temporal databases and their diverse applications are better understood. The TSQL2 and TSQL3 language design efforts are ongoing. In particular, the TSQL2 language design committee has released an initial language specification.

In sum, the last two years have seen a blossoming of consensus efforts within the temporal database community. The temporal database book, the glossary, the temporal query language test suite, the TSQL2 language definition, and this report are tangible artifacts of the community's new-found willingness to work together to produce a realistic, articulated, and coherent infrastructure on which future research and industrial development can proceed.

7 Acknowledgements

Curtis Dyreson contributed to the preparation of Section 2. Fabio Grandi's notes were helpful in the preparation of Section 3.

Gio Wiederhold has pushed for an infrastructure for temporal databases for many years now, and this workshop is one result of his efforts. Maria Zemankova was vital in shepherding the initial proposal through an NSF review. Without their diligence this workshop would never have occurred.

This workshop was supported by ARPA and the National Science Foundation under grant IRI-9304091. Support was also provided by Bell Communications Research and Tandem Corporation. However, the findings expressed in this report do not necessarily reflect the official position of any of the sponsoring agencies.

A Group Members

Participant	*Group*	*Affiliation*
Ilsoo Ahn	B	AT&T Bell Laboratories
Gad Ariav	A	Tel Aviv University, Israel
Marianne Baudinet	A	Universite Libre de Bruxelles, Belgium
José Blakeley	D	Texas Instruments
Mark Boddy	A	Honeywell Systems and Research Center
Alex Buchmann	C.2	Technisch Hochschule Darmstadt, Germany
Sharma Chakravarthy	C.2	University of Florida
Su-Shing Chen		National Science Foundation
Tsz-Shing Cheng	C.1's rapporteur	Iowa State University
James Clifford	B	New York University
Klaus Dittrich	C.2	Universitat Zurich, Switzerland
Curtis Dyreson	A's rapporteur	University of Arizona
Ramez Elmasri	D's coordinator	University of Texas at Arlington
Max Egenhoffer	A	University of Maine
Shashi K. Gadia	C.1; visited B	Iowa State University
Fabio Grandi	B's rapporteur	Università di Bologna, Italy
Pat Hayes	A	Beckman Institute
Sushil Jajodia	D	George Mason University
Christian S. Jensen	B	Aalborg Universitetscenter, Denmark
Wolfgang Käfer	B	Universitaet Kaiserslautern, Germany
Vram Kouramajian	D's rapporteur	University of Texas at Arlington
Krishna Kulkarni	B	Tandem Corporation
Ted Lawson	C.1	University of Wales College of Cardiff, U.K.
Nikos Lorentzos	B	Agricultural University of Athens, Greece
Kia Makki	D	University of Nevada
Inderpal Singh Mumick	C.2	AT&T Bell Laboratories
Frank Olken	A	Lawrence Berkeley Lab
M. Tamer Öszu	C.1	University of Alberta, Canada
Barbara Pernici	A's coordinator	Politecnico di Milano, Italy
Donna Peuquet	D	Pennsylvania State University
Niki Pissinou	C.1	University of Southwestern Louisiana
Krithi Ramamritham	C.2	University of Massachusetts
Scott Relan		Digital Systems Research
Arie Segev	C.2's coordinator	University of California at Berkeley
Richard Snodgrass	B's coordinator	University of Arizona
Michael Soo	C.1's rapporteur	University of Arizona
Sury Sripada	A	ECRC, Munich, Germany
Stanley Su	C.1	University of Florida
Abdullah Uz Tansel	B	City University of New York
Babis Theodoulidis	C.1's coordinator	UMIST, Manchester, U.K.
Vassilis Tsotras	D	Polytechnic University
David Wells	D	Texas Instruments
Gio Wiederhold		ARPA/SISTO
Gene Wuu	C.1	Bell Communications Research

B Proceedings Abstracts

As a prelude to the workshop, the invited participants prepared position papers for distribution about a month before the workshop. The participants were encouraged to read these position papers, to inform the discussions at the workshop.

The following is a list of the position papers, with their abstracts. The entire proceedings is available at cost by sending a request to Robyn Austin, Department of Computer Science, University of Arizona, Tucson, AZ 85721, or robyn@cs.arizona.edu.

Proposed Temporal Database Concepts–May 1993
Christian S. Jensen (editor), et al.

This document contains the complete set of glossary entries proposed by members of the temporal database community from Spring 1992 until May 1993. It is part of an initiative aimed at establishing an infrastructure for temporal databases. As such, the proposed concepts will be discussed during the "International Workshop on an Infrastructure for Temporal Databases," in Arlington, TX, on June 1993, with the specific purpose of defining a consensus glossary of temporal database concepts and names.

Earlier status documents appeared in March 1993 and December 1992 and included terms proposed after an initial glossary appeared in *ACM SIGMOD Record* in September 1992. This document subsumes all the previous documents. Additional information related to the initiative may be found at `ftp.cs.arizona.edu` in the `tsql` directory, accessible via anonymous ftp.

SQL+T: a Temporal Query Language
Ilsoo Ahn

Many temporal languages have been proposed over the years, but none of them has emerged as the standard. The absence of a standard language is one of the obstacles that slow the progress of the temporal database technology. It would be worthwhile to design and agree upon a temporal query language that encompass desirable characteristics of various proposals in a consistent framework. In this paper, we propose a new temporal query language, Structured Query Language plus Time (SQL+T). SQL+T is an intuitive and straightforward extension of SQL2, built on the existing features available in commercial relational DBMSs and the new SQL92 standard. And it is powerful enough to support the temporal requirements in real applications, and flexible enough to handle all the different types of time-oriented applications with varying degrees of temporal support. We define the semantics of the new language in SQL, demonstrating that they vary greatly depending on the types of temporal support. We also describe how to simulate SQL+T, mapped to SQL, on a conventional relational DBMS. We believe SQL+T includes many useful concepts that can be considered for the standard temporal language.

Tools for Managing Temporally Oriented Data: Are They *Really* Practically Irrelevant?
Gad Ariav

The discussion of infrastructure for temporal databases rests on the premise that the widespread use of tools for temporally oriented data management is inhibited by the lack of coherent, standardized view of this discipline. Alas, a decade of fairly intensive study of the various aspects temporally oriented information systems (TOIS) has created so far only limited interest outside the respective academic communities. Is the concept practically irrelevant?

In this paper we report the initial findings of a study that attempted to develop some deeper sense of the value and need for TOIS in information systems practice. As indicated by the study, the actual reference to temporally oriented data management tools and concepts is indeed abundant, and thus the inhibiting factors have to be sought elsewhere. We then speculate about possible difficulties in the practical implementation of TOIS and suggest related research questions.

Temporal Databases: Beyond Finite Extensions
Marianne Baudinet, Jan Chomicki, and Pierre Wolper

We argue that temporal databases should not be restricted to relations with finite extensions. Many temporal events are periodic and have no natural bounds. Moreover, such events have a more compact representation when allowed to be unbounded. We present two formalisms for representing and querying possibly infinite periodic data and discuss some of their properties, including expressiveness and query evaluation complexity. Finally, we turn to implementation issues and argue that significant extensions to existing database systems are necessary in order to implement the frameworks we describe.

Challenges for Research on Temporal Databases
José A. Blakeley

This paper poses five challenges to the temporal database research community. (1) To unify the support of time-varying data with other notions of change in databases such as schema evolution, versioning, and change management under a common theme of *database change.* (2) To develop a core, reusable set of temporal constructs that can be used to extend any data model with time-varying semantics. (3) To design new database query languages in such a way that they can accommodate any extensions to the data model including time-varying, distribution, and persistence extensions. (4) To develop an extensible, temporal query processing framework that allows the temporal database research community to experiment with newly discovered algebraic operators, equivalence transformations, execution algorithms (e.g., temporal sort-merge join), cost and selectivity estimations, and state space search strategies to find the most effective combination of ideas. (5) To develop a standard benchmark for temporal databases that includes both functionality and performance benchmarks similar to the TPC-X benchmarks for relational databases. The paper also proposes the Open Object-Oriented Database Management System (*Open OODB*) developed by Texas Instruments as part of the ARPA Persistent Object Bases program as an example system that can provide the basic infrastructure required by the temporal database community to enable experimentation of new ideas and effective transfer of temporal database technology.

On Combining Temporal and Real-Time Databases
A.P. Buchmann and H. Branding

This position statement introduces three issues that are relevant in combining temporal and real-time databases. First, it introduces the specification of validity intervals and the derivation of timing constraints from them. Next it states the need to consider future time in temporal databases, and the need to unify the representation of time in temporal and active databases. Finally, it addresses the handling of contingency plans and the need for branching time.

Semantics of Time-varying Information and Resolution of Time Concepts in Temporal Databases
Sharma Chakravarthy and Seung-Kyum Kim

In this paper, we first identify a number of temporal database issues that need to be addressed in order to establish an infrastructure for temporal database research and systems development. These issues are: non-availability of commercial temporal databases, number and semantics of time concepts required, query language extensions, possible layered architecture for supporting time-varying information, to name a few. This paper specifically elaborates on the semantics of time concepts in temporal databases - one of the issues identified earlier. We present a formal definition of temporal validity. The notion of temporal validity is extended to the interpretation-based validity, with which the confusion among various time concepts introduced earlier for temporal databases can be dispelled. Then, we discuss the problem of preserving multiple past states of a temporal database, which leads to the identification of a maximal set of time concepts. It is shown that the time concept, event time is needed to properly model retroactive and proactive updates, as it is not possible to model them using only the valid and transaction times as thought earlier. We also show the adequacy of three time concepts for completely preserving different past states generated by retroactive and proactive updates, error corrections, and delayed updates.

Grouped and Ungrouped Historical Data Models: Expressive Power and Completeness
James Clifford, Albert Croker, and Alexander Tuzhilin

Numerous proposals for extending the relational data model to incorporate the temporal dimension of data have appeared in the past several years. These proposals have differed considerably in the way that the temporal dimension has been incorporated both into the structure of the extended relations of these temporal models, and consequently into the extended relational algebra or calculus that they define. Because of these differences it has been difficult to compare the proposed models and to make judgments as to which of them might in some sense be equivalent or even better. In this paper we define the notions of temporally grouped and temporally ungrouped historical data models and propose two notions of historical relational completeness, analogous to Codd's notion of relational completeness, one for each type of model. We show that the temporally ungrouped models are less expressive than the grouped models, but demonstrate a technique for extending the ungrouped models with a grouping mechanism to capture the additional semantic power of temporal grouping. For the ungrouped models we define three different languages, a temporal logic, a logic with explicit reference to time, and a temporal algebra, and show that under certain assumptions all three are equivalent in power. For the grouped models we define a many-sorted logic with variables over ordinary values, historical values, and times. Finally, we demonstrate the equivalence of this grouped calculus and the ungrouped calculus extended with a grouping mechanism. We believe the classification of historical data models into grouped and ungrouped provides a useful framework for the comparison of models in the literature, and furthermore the exposition of equivalent languages for each type provides reasonable standards for common, and minimal, notions of historical relational completeness.

On the Semantics of Transaction Time and Valid Time in Bitemporal Databases
James Clifford and Tomás Isakowitz

Numerous proposals for extending the relational data model to incorporate the temporal dimension of data have appeared in the past several years. While most of these have been historical databases, incorporating in some fashion a valid time dimension to the data model and the query languages, others have been rollback databases, incorporating a transaction time dimension, or bitemporal databases, incorporating both of these temporal dimensions. In this paper we address an issue that has been lacking in many of these papers, namely, a formal specification of the precise semantics of these temporal dimensions of data. We introduce the notion of reference time - the time that any operation is applied to the database state - and provide a logical analysis of the interrelationships among these three temporal dimensions. We also provide an analysis of the meaning of various variables such as *now* and ∞ which have been used in many of these models without a complete specification of their semantics.

Extending Existing DBMSs to Manage Temporal Data: An Object-Oriented Approach
Umeshwar Dayal and Gene T.J. Wuu

In this paper we propose an object-oriented approach to support temporal data modeling and manipulation. We rely on the rich object-oriented type system to model temporal information, and use an existing database language to express complex temporal queries. There are two major benefits to this approach: (1) temporal and non-temporal queries can be expressed uniformly using the same language, and (2) existing DBMSs and their query processors can be smoothly extended to support the temporal database features. To show the general applicability of our approach and to have maximum impact on the industry, we chose to illustrate our approach using the object extension of the current SQL3 draft.

Time Issues in Active Database Systems
Klaus R. Dittrich and Stella Gatziu

Active mechanisms based on event-condition-action rules will play an important role in next-generation database management systems. As an event, in its most general form, is essentially a point in time, it is obvious that an appropriate concept of time is needed for the specification of events. However, there are also other aspects related to time that need to be considered in active database systems, and which should tie in with the general concept of time in case the active database is also a temporal one. This position paper gives a brief account of where time issues arise in active database systems, and especially demonstrates various options for powerful event specification features.

A Proposed Timestamp Format
Curtis E. Dyreson

Many database management systems and operating systems provide support for time values. At the physical level time values are known as timestamps. A timestamp has a physical realization and a temporal interpretation. The physical realization is a pattern of bits while the temporal interpretation is the meaning of each bit pattern, that is, the time each pattern represents. We propose a timestamp physical realization for events, intervals and spans, based on three basic timestamp formats. These formats can represent all of time to the granularity of a second, and all of recorded history to a finer granularity of a microsecond. Our proposed formats were designed to be more space and time efficient than existing timestamps. We compare our timestamp formats with those used in common operating systems and database management systems. We recommend that these formats be used for user-defined, transaction-time, and valid-time timestamps.

Indexing, Searching and Archiving Issues in Temporal Databases
Ramez Elmasri and Vram Kouramajian

In this paper, we identify many of the key issues related with storing, searching and archiving of temporal data, and outline an infrastructure for a temporal storage model. The infrastructure encompasses an indexing mechanism, physical storage design, search algorithms and archiving strategies that take into account the characteristics of temporal data.

Our index is suitable for the majority of temporal search operations. The archiving model assumes a two-level magnetic-optical storage system, and includes strategies for dealing with very long lived versions. A variation of the time index, the monotonic B+-tree (MBT), is also described. The MBT is very efficient for append only databases that exhibit monotonic growth over the time dimension.

An object-oriented model for temporal databases
Tsz S. Cheng and Shashi K. Gadia

We propose an Object-Oriented Temporal Database Model (OOTempDBM) and a query language OOTempSQL for handling temporal data. Our model captures the semantics of temporal objects through type inheritance. Our underlying principles for handling temporal data are independent of the choice of a data model. In past, these principles have been applied to our relational model. To further our work, we show how our principles can be applied in object-oriented environment. As some infrastructure is already available in OODAPLEX to suit our needs, we have chosen it as a base to build the rest of our model. We also compare OOTempSQL to the query language in OODAPLEX, and argue that the former is higher level than later. In addition OOTempSQL has many desirable features not found in other models for temporal data.

Updates and incremental recomputation of active relational expressions in temporal databases
Madhuri L. Edara and Shashi K. Gadia

An active expression is one whose result is explicitly stored by the system. A technique for incremental recomputation of active relational expressions for relations in the classical database setting has been proposed by Qian and Wiederhold. We extend their technique to a temporal database model. Because temporal tuples can be very large relative to incremental changes, we introduce the notion of horizontal tuple fragments, and propagate them in incremental recomputation. We present propagation rules for a certain restructuring operator in the chosen temporal database model. We show that the propagation of fragments works nicely in case of weakly invariant expressions in the temporal model. We enumerate problems in propagating tuple fragments in non weakly invariant expressions and observe that full tuples need to be propagated in their recomputation.

SQL-like seamless query of temporal data

Shashi K. Gadia and Gautum Bhargava

In this paper we present our model and a query language TempSQL for seamless integration of static, snapshot and temporal data. In our model the compatibility of TempSQL with SQL is achieved through the concept of a user. Our one dimensional model is generic in the sense it can be used to query valid or transaction time data, and also readily extends to the bitemporal case. When our model is applied to transaction time we obtain a capability of querying updates and queries. In the bitemporal case our model yields a capability to query errors in every day record keeping. It is also argued that in the bitemporal case the storage requirements can be reduced to that for storing one dimensional valid time database plus the amount of error in the database.

An SQL-like seamless query language for spatio-temporal data
Shashi K. Gadia, Vimal Chopra, and U. Sunday Tim

There is a need for query languages in spatio-temporal databases that treat spatial and temporal dimensions uniformly. The users should be given a simple view of data and freed of the worry of how it is physically represented. This is even more important because physical implementation of spatial data is expected to be a topic of study for quite some time to come. We present a model and an SQL-like query language called ParaSQL for spatial data. Without tying ourselves down to any particular implementation, we propose certain closure properties for spatio-temporal regions to make ParaSQL seamless. Space and time coexist seamlessly providing unrestricted navigation in SpaSQL queries. To illustrate the modeling and querying capability in our framework, we present a case study of an application in agriculture systems and environmental management.

Incomplete information in relational temporal databases
Shashi K. Gadia, Sunil S. Nair and Yiu-Cheong Poon

For the conventional relational model there has been considerable research in the area of incomplete information. On the other hand, research in temporal databases has concentrated on models in which complete historical information is needed. However, the likelihood of missing information in temporal databases is greater because of the vast amount of information. Hence, a mechanism must be provided to store and query incomplete temporal information. In this paper we present a model for incomplete information in temporal databases. The model generalizes our previous model for complete temporal information. It is shown that our relational operators produce results that are reliable. We also show, with some exceptions, that if the definitions of the operators were strengthed to give more information, we may obtain results that are not reliable.

Temporal Mediators as a way to Support Multiple Temporal Representations
X. Sean Wang and Sushil Jajodia

In temporal databases, each object is associated with some timestamped data, i.e., time-varying attribute values together with timestamps that give periods of validity to these values. Timestamped data often exhibit more complicated syntax and semantics than conventional (non-temporal) data. This leads to at least two problems: (1) Users have to understand possibly different logical and physical structures of the underlying data in these databases, and (2) user queries must be in terms of the time units (e.g., day) on which the timestamps are based. These two problems may prevent a user from using the databases effectively. In order to solve these two problems, we introduce the concept of a temporal mediator.

A temporal mediator consists of three components: (i) a reservoir for windowing functions and conversion functions, (ii) a time unit thesaurus and (iii) a query interpreter. There are two types of windowing functions, one associates time points to sets of tuples, and the other associates tuples to sets of time points. A conversion function transforms information in terms of one time unit into that in terms of some other time unit. The time unit thesaurus stores the knowledge about time units (e.g., names of time units and relationships among them). Users pose queries using the windowing functions and desired time units. (A query language, which can be used to form such queries, is given in the paper.) To answer such a user query, the query interpreter first employs the windowing functions together with the time unit thesaurus to access the temporal data from the underlying databases and then uses the time unit thesaurus to select suitable conversion functions which convert the responses to the desired time units. Thus, a temporal mediator provides a simple interface that supports multiple temporal representations.

The concept of the temporal mediator leads the authors to propose to the workshop the following research directions: (1) Continue the current investigation into logical and physical aspects of different temporal data models, and (2) initiate an investigation into the issues related to the user interfaces and the interaction between these user interfaces and the temporal data models.

Three Proposals for a Third-Generation Temporal Data Model
Christian S. Jensen and Richard Snodgrass

We present three general proposals for a next-generation temporal data model. Each of these proposals express a synthesis of a variety of contributions from diverse sources within temporal databases. We believe that the proposals may aid in bringing consensus to the area of temporal data models.

The current plethora of diverse and incompatible temporal data models has an impeding effect on the design of a consensus temporal data model. A single data model is highly desirable, both to the temporal database community and to the database user community at large. It is our contention that the simultaneous foci on the modeling, presentation, representation, and querying of temporal data have been a major cause of the proliferation of models. We advocate instead a separation of concerns.

As the next step, we propose a data model for the single, central task of temporal data modeling. In this model, tuples are stamped with bitemporal elements, i.e., sets of pairs of valid and transaction time chronons. This model has no intention of being suitable for the other tasks, where existing models may perhaps be more appropriate. However, this model does capture time-varying data in a natural way.

Finally, we argue that the flexible support for physical deletion is needed in bitemporal databases. Physical deletion requires special attention in order not to compromise the correctness of query processing.

Temporal Selection, Temporal Projection, and Temporal Join Revised
Wolfgang Käfer

This paper presents an innovative approach to temporal relational operations, in particular to the temporal join operation. However, before this operation can be discussed the more basic operations, such as temporal selection, temporal projection and temporal Cartesian Product have to be introduced. These operations are part of a temporal relational data model which provides valid- time relations with a tuple-oriented time-stamping scheme. Our new approach to the temporal join operation stems from the observation that units of all temporal operations are histories - and not single tuples. A history or time sequence is a time-ordered list of tuples describing the same real-world entity. For example, in order to reflect a change concerning the entity, a new tuple representing the new state must be inserted into this list (not necessarily at the end). Thereby integrity constraints defined for the list must be checked, e.g. does the new data with its validity interval interfere or contradict with already stored data and validity intervals? Choosing time sequences as the units for our temporal operations offers new insights, e.g. into the temporal join operation which is traditionally defined on a tuple- oriented basis. Along with the discussion of the temporal operations we propose extensions of SQL to accommodate the new (temporal) functionality. Operations and associated language extensions are illustrated on a small sample database.

An Object-Oriented Approach to Temporal Modelling
Ted Lawson, Carmel Balthazaar, and Alex Gray

This paper discusses an object-oriented approach to temporal modelling which we believe to be relevant to the development of an infrastructure (or infrastructures) for temporal databases, and in particular to the development of conceptual temporal models, and the establishment of standard models. It is clear that there are many facets to the representation of time. For example, in a business application the time span between two events could be measured in Gregorian calendar days or in working days. We describe a collection of classes encoded in the Eiffel object-oriented language which together embody a multi-faceted temporal model. At present the classes exist only as executable object-oriented language code, but they could be incorporated into some future object-oriented DBMS. They fall into three groups and each group reflects a different principal facet of the model. The first is independent of any particular representation of time; the second reflects time represented in the Gregorian calendar; the third includes specialized extensions such as time-zone dependent time and "working day" durations. The collection is openly extendible, so that it could, for example, include a facet to reflect time represented using an Islamic calendar. The model can be seen as incorporating a conventional relational temporal model, and could be extended to incorporate a knowledge-based one. We suggest that such an extendible multi-faceted model could be the basis for a standard temporal model.

Axiomatic Generalization of the Relational Model to Support Valid Time Data
Nikos A. Lorentzos

In recent years a lot of research has been undertaken in valid time data modelling. Many extensions to the snapshot relational model have been proposed but researchers in the area have not come to a general agreement about the properties which a reference valid time relational model should satisfy. This paper aims at closing this gap, by proposing a set of properties subject to constructive discussion, in order to achieve a standardization of the properties of a reference valid time Database Management System. To this end, the properties are identified by which the snapshot relational model has to be enhanced, so as to handle valid time data. Two reference valid time relational models are specified. The first is a generalization of the snapshot relational model which satisfies First Normal Form. The second is a generalization of the nested snapshot relational model. It is the most general in that it can manage any piece of data which can be managed by any of the other models. It is then shown that all valid time extensions to the relational model which have been proposed, represent approaches towards the formalization of these two reference valid time data models and, therefore, they all have to be further extended.

Algebraic optimization in a relational model for temporal databases
Sunil S. Nair and Shashi K. Gadia

For the conventional relational model there has been considerable research in the area of query optimization. However, research in temporal databases has concentrated on the development of models to store and query historical information. The amount of data stored in a temporal database can be extremely large. Hence if temporal databases are to become viable in practice, it is important that optimization for temporal queries be studied. In this paper we develop a set of algebraic identities for a model for temporal databases. We present an algorithm that uses these identities to convert a given temporal query to another equivalent query that will execute more efficiently. We also give algorithm to compute the cross-product of two temporal relations, which yields substantial savings over the brute force method under certain conditions.

Research Perspective on Time in Object Databases
Niki Pissinou, Kia Makki, and Yelena Yesha

There are currently many data models that have powerful modeling constructs designed to make the tasks of database design, evolution, and manipulation simple and easy for database designers and users. In this paper, we look at the issues of dealing with temporal modeling in the context of object databases. In particular, we provide a survey of some important research achievements in temporal databases from the past two decades and significant contributions from related areas. We also examine a number of major objectives and areas of challenge which remain for researchers and implementors of temporal object database systems and discuss the time dimension in relation to object data modeling. We investigate the important issues that arise when attempting to integrate time with object databases and present our approach to temporal object modeling. Our main objective here, is to provide the necessary background and motivation to design and develop a model that integrates time with objects, thus supporting temporal temporal data and the temporal evolution of data in an object database framework. Such a work will present a significant step towards the synthesis of an integrated object data model with a high level of abstraction, that supports the temporal and dynamic aspects of data modeling in addition to structural and behavioral ones.

Temporal Extensions to a Uniform Behavioral Object Model
Iqbal A. Goralwalla and M. Tamer Özsu

We define temporal extensions to a uniform, behavioral and functional object model. This is on-going work and our proposals in this paper should be interpreted as an indication of our current thinking and the directions that we are following rather than as finalized research. We hope that these ideas will be topics of discussion at the workshop as part of the infrastructure for temporal databases.

Towards the development of a general Temporal Manager for Temporal Databases: a layered and modular approach
Luca Console, Barbara Pernici, and Paolo Terenziani

Two main aspects have been taken into account in the design of languages and systems for managing temporal information: the trade-off between expressive power and computational complexity of temporal reasoning and the desire to provide high level common-sense primitives for data manipulation and retrieval with a precise semantics. A layered and modular architecture facilitates the approach to the above mentioned problems: a temporal manager can be loosely coupled to existing DBMS to provide time-related functionalities; at the physical level, temporal information can be organized in such a way that, with a number of well defined limitations in the expressive power of the language, a correct and complete propagation of temporal constraints is tractable; access to temporal information can be provided with the traditional temporal primitives provided in the literature, by providing higher level representations, to facilitate users' understanding of performed operations. The paper presents LATER (LAyered TEmporal Reasoner), an architecture for the management of temporal information for Databases, developed according to the above principles. LATER is a modular (and extensible) architecture providing a set of primitives for data manipulation and query. Moreover, since in LATER temporal reasoning is performed during data manipulation (insertion and/or deletion of temporal information), queries can be answered in constant time, regardless of the contents and dimension of the database.

Towards a Temporal Logic Reconstruction of Temporal Databases
Angelo Montanari and Barbara Pernici

This position paper aims at contributing to a formal, consensus definition of the semantics of time-varying information in temporal databases. Such a formalization is necessary to precisely characterize their requirements and expressiveness. The paper proposes a temporal logic reconstruction of (bi) temporal databases that can be seen as the logical counterpart of the relational algebra approaches to temporal database semantics formalization. More precisely, we define a topological, bidimensional temporal logic, provided with a model-theoretic semantics and a sound axiomatization, that combines Rescher and Urquhart's topological temporal logic and Gabbay's two dimensional temporal logic. It allows us to formally deal with basic concepts as ordering and metric temporal relations, chronologically stable and unstable times, transaction and valid time. We also show how this formalism can be extended to provide a formal basis for advanced features such as time granularity, multiple versions, and variants. To illustrate the capabilities of the proposed formalism, we present in some detail the formalization of transaction and valid times. In the last section, we illustrate how it can be used for reasoning about stored temporal information at query time.

A Framework for the Representation of Spatiotemporal Processes in Geographic Information Systems
Donna J. Peuquet

The need for greater understanding of regional and global environmental processes and how man's activities are affecting the natural environment is being viewed with increasing urgency. Although the study of dynamics in space/ time is certainly not new, nor is it unique to a single field, addressing current human and environmental issues requires empirical examination from a much broader and integrated perspective than our current representational techniques will currently allow.

Current representational approaches for geographic data are based on a traditional cartographic paradigm, and are thus also geared toward representation of static situations. Cartography has traditionally focused on the visible representation of a portion of the world at a specific point in time. Although Geographic Information Systems (GIS) are intended to provide an integrated and flexible tool for analyzing large volumes of data, representations historically used in GIS are also geared toward a similar static view. Efforts to enhance the temporal capabilities of GIS have served to reveal many problems at a fundamental, conceptual level.

A representational framework that unifies temporal, as well as spatial and object (i.e., feature-related) aspects is described which incorporates concepts from Perceptual Psychology, Artificial Intelligence and other fields. It is the goal of this research to provide a drawing-together of many concepts and ideas, many of which are well known on an individual basis, so that not only can analytical tools be improved, but also to provide a more common ground among various fields.

Time for Real-Time Temporal Databases?
Krithi Ramamritham

Time related concepts appear in both temporal databases and real-time databases.In this position paper, we show that with minor changes to the recently compiled basic concepts of temporal databases, it is possible to cover the conceptual needs of real-time databases as well. We also discuss practical implementation-oriented issues that arise when real-time databases are married to temporal databases. The motivation behind this paper is to determine whether, as we attempt to standardize temporal database concepts, it is possible to take into account the specific characteristics of real-time databases as well so as to expand the applicability of temporal databases.

Temporal Active Databases
Opher Etzion, Avigdor Gal and Arie Segev

In recent years there has been an increasing effort to provide database support to complex new applications such as computer aided design, office information systems, manufacturing information systems, and scientific databases. Those efforts resulted in advanced data models and database management systems, such as extensible relational systems, object-oriented databases, and deductive databases. More recently there has been an effort to integrate the different non-traditional functionalities into single systems. We argue that a temporal database (or a data model) should be specified in the context of a non-temporal database (or data model). Consequently, existing data models should be augmented with temporal support. Obvious candidates are the relational model, extended relational models, object-oriented models, and statistical databases. Quite a few papers have been published on temporal relational databases, and presumably many will be included in this workshop. In this paper we present (via examples) functionality which is needed to support object-oriented features, rules and constraints, and temporal semantics including data and meta-data histories, multiple time types, and temporal transactions.

Multiple Calendar Support for Conventional Database Management Systems
Michael D. Soo

We describe a specific approach to supporting a time-stamp attribute domain in conventional relational database management systems. In contrast to existing proposals, which assume that a single interpretation of time is sufficient for all users and applications, we advocate a general solution that supports multiple interpretations of time. The main concept underlying this proposal is that the universal aspects of time are separated from the user dependent aspects, at both the query language and the architectural levels. The user dependent aspects are encapsulated in calendars and calendric systems, each of which are extendible by local site personnel. In this way, the available time support can be customized to local requirements. We briefly describe modifications to SQL2 that support multiple calendars and calendric systems. These modifications reduce the complexity of the language while simultaneously increasing its expressive power. Finally, we describe a set of tools that aid in the generation of calendars and calendric systems. This work can be viewed as a limited but practical application of research into extensible database management systems.

Design of the ChronoBase Temporal Deductive Database System
S.M. Sripada

The various considerations that have influenced the design of the conceptual model, query language, and architecture of ChronoBase, a temporal deductive database system being developed at ECRC, are briefly described in this position paper.

Modeling and Management of Temporal Data in Object-Oriented Knowledge Bases
Stanley Y.W Su and Hsin-Hsing M. Chen

There has been a considerable amount of work on object-oriented databases, active databases, and deductive databases. The common objective of these efforts is to produce highly intelligent and active systems for supporting the next generation of database applications. These future systems must be capable of capturing the concepts of time and managing not just temporal data but temporal knowledge expressed by knowledge rules. In this paper, we describe the work on a temporal object-oriented knowledge model OSAM*/T, its associated temporal query language OQL/T, an underlying temporal algebra TA-algebra, and some implementation techniques developed at the Database Systems Research and Development Center of the University of Florida. In addition to the characteristics of traditional object-oriented paradigm, the model is featured by its strong support of association types and its incorporation of temporal knowledge rules for specifying temporal and other types of semantic constraints associated with object classes and their temporal object instances. The query language is featured by its pattern-based specification of temporal object associations which allows complex queries with various time constraints to be formulated in a relatively simple way. The temporal algebra provides a set of primitive operators for manipulating homogeneous and/or heterogeneous patterns of temporal object associations, thus providing the needed mathematical foundation for processing and optimizing temporal queries. The implementation techniques include a Delta-Instance and Multi-Snapshot Storage Model, and data partitioning and clustering schemes for storage management of temporal knowledge bases. The purpose of this paper is to share with the research community the work accomplished and the on-going effort at UF with the hope to contribute to the workshop's objective of establishing "an infrastructure supporting both desirable practice and future research on temporal databases."

$\mathrm{SQL}^{\mathrm{T}}$: A Temporal Extension to SQL
Abdullah Uz Tansel

In this paper, we propose an extension to SQL, called SQLT for handling temporal data. The underlying data model allows sets as attribute values. Each attribute value can be a simple value or a temporal atom which consists of a temporal set, a set of time points and a value. A temporal atom asserts that the value is valid over the time period represented by the temporal set. SQLT allows set comparison, set operations on temporal sets and SELECT statement in the target list. Features of SQLT are illustrated by examples.

We believe that our extension to SQL will help to identify fundamental constructs needed for manipulating temporal data as well as the desired features of the underlying data model. Thus, it will contribute to establish a common foundation in temporal databases and the standardization efforts in query languages.

Towards the First Generation of Temporal Information Servers
Babis Theodoulidis

The development and installation of medium and large scale information systems are based, to a high extend, on Database Management Systems (DBMS). Today, there are many organisations which have to process historical data, using conventional commercial DBMSs. However, existing commercial DBMS technology do not provide support for the management of temporal data. As a consequence, large pieces of application dependent code have to be written in order to accommodate time dependent application requirements. This paper discusses the requirements for the first generation of Temporal Database Management Systems and proposes an overall component architecture for such systems. In addition, a Temporal Information Model (TIM) is described that addresses the needs for application independent representation of temporal information and a workplan for the consolidation of the work in this area is proposed.

Access Methods for Historical Data
Nickolaos Kangelaris and Vassilis J. Tsotras

In this paper we attempt a comparison of different indexing techniques that have been proposed for efficient access to historical data. The comparison is based on a collection of criteria that we believe are of importance in dealing with such indexing methods. As this is an on-going effort we do not claim that this is a complete comparison, either in covering all related indexing methods or in the criteria used. However, such a paper serves as a constructive summary of what has been achieved, to the best of our knowledge, in the area of access methods for temporal databases. Our purpose is to identify the difficult problems in accessing historical data and provide a good description of how the different methods aim to solve them. We have studied six such methods, namely, the Time-Split B-tree, the Time Index, the Segment R-tree, the fully Persistent B+-tree, the Append-only tree and the Snapshot index.

Exploring the Possibility of Time as a Seamless Database Extension
David L. Wells

The Open Object-Oriented Database (Open OODB) seamlessly adds database functionality such as persistence, transactions, distribution, views, versions, and automatic index maintenance to popular object-oriented programming languages such as C++ and CLOS. These extensions are supported in a uniform manner in the Open OODB by a common computational model and database management systems (DBMS) infrastructure.

If temporal semantics can be added to databases using the same computational model and mechanisms, programmers can be presented with an abstraction of temporal functionality that is compatible with other semantic extensions, and the developers of temporal DBMSs can use the Open OODB's infrastructure to implement their temporal extensions. To determine if this is feasible, this paper sketches Open OODB's computational model and infrastructure, and explores aspects of temporal extensions that appear amenable to the same model and mechanisms. It is hoped that this will allow experts in temporal databases to determine the feasibility of using the Open OODB architecture as an infrastructure component for temporal databases.

The Need to Harmonize Temporal Data Models
Gio Wiederhold

The objective of DARPA in sponsoring this workshop is to establish a common model for temporal databases at a higher level of abstraction than we have now. Having a common model will enhance the research of those subscribing to it, since now their results can be shared in the accepted scientific tradition. Researchers choosing alternative models can discuss their differences in a relative way, and their readers can clearly achsses the specific choices being made; just as now new database concepts are described relative to the relational model.

If the temporal database research community is to make a link with practice, then some common voice is needed. Such a voice warrants support for substantial research, even while alternative models and assessed and being proposed. A common model provides a harmonious base for this voice.

Temporal Queries for Active Database Support
Narain Gehani, H.V. Jagadish, Inderpal Singh Mumick and Oded Schmueli

An active database monitors events (such as access, insert, delete, and update of tuples/objects, and invocation of methods on objects). Each event potentially changes the database state. Applications may wish to react to sequences of events and database states satisfying certain properties. Specification of such sequences can be viewed as a formulation of temporal queries. Further, monitoring such sequences, or equivalently, the evaluation of such temporal queries, must be efficient, and must not require storage of the entire database history.

In this paper, we present a language for specifying temporal queries that are of interest in an active database, and for which we can hope to develop efficient evaluation techniques.

History and Tuple Variables for Temporal Query Languages

Fabio Grandi, Maria Rita Scalas and Paolo Tiberio

Standard relational query languages provide range variable whose values are tuples and which are used to denote objects (entities or relationships). Snapshot relational databases represent an object by means of a tuple, whereas temporal relational databases represent an object by means of a collection of time-stamped tuples with the same key value. Temporal extensions of query languages proposed so far are also provided with range variable whose values are still tuples and so can no longer denote objects, but time-stamped object versions.

In this paper we denote as history the set of all the tuples with the same key value in a relation and propose the introduction of two new kinds of range variables: history variables which have histories as values and denote objects and tuple variables which have tuples as values and denote object versions. We also illustrate temporal extensions to SQL with these two types of variables. Some examples show how history and tuple variables can improve clarity and ease of use of a temporal query language. Therefore we propose that the two types of variables should be part of a common infrastructure for a temporal SQL extension.

IXSQL: An Interval Extension to SQL
Nikos A. Lorentzos and Yannis G. Mitsopoulos

IXSQL, an extension of SQL, is proposed for the management of data of a generic interval data type. In addition to the five primitive operations on the conventional relational model, it supports two more relational algebra operations, new comparison operations and new scalar functions. Historical databases is one of the many application areas of IXSQL, in which an implementation is currently in progress.

The TSQL Benchmark
Christian S. Jensen (Editor)

This document presents the temporal database community with an extensive, consensus benchmark for temporal query languages. The benchmark is semantic in nature. It is intended to be helpful when evaluating the user-friendliness of temporal query languages, including proposals for the consensus temporal SQL that is currently being developed.

The benchmark consists of a database schema, an instance for the schema, and a set of queries on this database. The queries are classified according to a taxonomy, which is also part of the benchmark.

References

[Ahn93] I. Ahn. SQL+T: a Temporal Query Language. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[All83] J.F. Allen. Maintaining Knowledge about Temporal Intervals. CACM, 26(11):832–843, November 1983.

[AMC93] E. Anwar, L. Maugis, and S. Chakravarthy. A New Perspective on Rule Support for Object-Oriented Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 99–108, 1993.

[Ariav93] G. Ariav. Tools for managing temporally oriented data: are they really prectically relevant? In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[AS86] I. Ahn and R. Snodgrass. Performance Evaluation of a Temporal Database Management System. In *Proceedings of the SIGMOD International Conference*, Washington, DC, pp. 96–107, May 1986.

[BB93] A.P. Buchmann, H. Branding. On Combining Temporal and Real-Time Databases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[BM91] Catriel Beeri and Tova Milo. A model for active object oriented database. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, pages 337–349, Barcelona, Spain, September 1991.

[CC87] J. Clifford and A. Croker. The historical relational data model (hrdm) and algebra based on lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, CA, Feb 1987.

[CC88] J. Clifford and A. Croker. "Objects in Time," IEEE Database Engineering Bulletin, 11(4), December 1988.

[CCT93] J. Clifford, A. Croker, and A. Tuzhilin. On completeness of historical relational query languages. Technical report STERN IS-93-8, New York University Stern School of Business, 1993. (to appear in TODS).

[CG93] T.S. Cheng and S.K. Gadia. An Object-Oriented Model for Temporal Databases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[Cha92] S. Chakravarthy. Architectures and monitoring techniques for active databases: An evaluation. UF-CIS TR-92-041. (Submitted to Applied Data and Knowledge Engineering Journal).

[Chen76] P.P.-C. Chen. The Entity-Relationship Model-Toward a Unified View of Data. ACM TODS 1(1):9–36, March 1976.

[Chom92] Jan Chomicki. Real-Time Integrity Constraints In *Proceedings of the Eleventh Symposium on Principles of Database Systems*, pp. 274–281, San Diego, CA, June 1992.

[CK93] S. Chakravarthy and S.K. Kim. Semantics of Time-Varying Information and Resolution of Time Concepts in Temporal Databases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, Tx, June 1993.

[Cli82] J. Clifford. A model for historical databases. In *Proceedings of Workshop on Logical Bases for Data Bases*, Toulouse, France, December 1982.

[CM93] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. UF-CIS-TR-93-007, University of Florida, March 1993. (Revised and extended version of UF-CIS-TR-91-23).

[CODD79] E.F. Codd. Extending the Database Relational Model to Capture More Meaning. ACM TODS, 4:397–434, 1979.

[CS93] R. Chandra and A. Segev. Managing Temporal Financial Data in an Extensible Database. In *Proceedings of the 19th Int. Conf. on Very Large Databases*, Dublin, Ireland, September, 1993.

[CSS93] R. Chandra, A. Segev, and M. Stonebraker. Implementing Calendars and Temporal Rules in Next-Generation Databases. Technical Report LBL-34229, Lawrence Berkeley Laboratory, June 1993.

[CT85] J. Clifford and A.U. Tansel. On an algebra for historical relational databases: Two views. In *Proceedings of the International Conference on Management of Data*, pages 247–265, Austin, TX, May 1985.

[CW83] J. Clifford and D. S. Warren. Formal semantics for time in databases. ACM TODS, 8(2):214–254, June 1983.

[Dat88] C.J. Date. A proposal for adding date and time support to sql. *ACM SIGMOD Record*, 17(2):53–76, June 1988.

[DBB+88] U. Dayal, B. Blaustein, A. Buchmann, U. Chakravarthy, M. Hsu, R. Ladin, D.R. McCarthy, A. Rosenthal, S. Sarin, M.J. Carey, M. Livny, and R. Jauhari. The HIPAC project: Combining active databases and timing constraints. *ACM SIGMOD Record*, 17(1):51–70, March 1988.

[DG93] K.R. Dittrich and S. Gatziu. Time Issues in Active Database Systems. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[DH87] E. Dubois and J. Hagelstein. Reasoning on Formal Requirements: A Lift Control System. In *Proceedings on S/W Specification and Design*, 1987.

[DHL91] U. Dayal, M. Hsu, and R. Ladin. A transaction model for long-running activities. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, pp. 113–122, Barcelona, Spain, September 1991.

[DUB+86] E. Dubois, et al. The ERAE Model : A Case Study in Information Systems Design Methodologies : Improving the Practice (CRIS-3). T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart (eds), North-Holland, 1986.

[DW90] C. J. Date and C. J. White. *A Guide to DB2*, Volume 1, Third edition. Addison-Wesley, Reading, MA, September 1990.

[DW92] U. Dayal and G.T.J. Wuu. A Uniform Approach to Processing Temporal Queries. In *Proceedings of the International Conference on Very Large Databases*, Vancouver, Canada, August 1992.

[EGS92] O. Etzion, A. Gal, and A. Segev. Temporal Support in Active Databases. In *Proceedings of the Second International Workshop on Technologies and Systems*, Dallas, TX, December 1992.

[EGS93a] O. Etzion, A. Gal, and A. Segev. Retroactive and Proactive Database Processing. Technical Report LBL-34424, Lawrence Berkeley Laboratory, July 1993.

[EGS93b] O. Etzion, A. Gal, and A. Segev. Temporal Active Databases. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[EW90] R. Elmasri and G.T.J Wuu, A Temporal Model and Query Language for ER Databases. In Proceedings of the International Conference on Data Engineering, May 1990, pp. 76–83.

[EWK93] R. Elmasri, G. Wuu and V. Kouramajian. A Temporal Model and Query Language for EER Databases. Chapter 9, Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993, pp. 212–229.

[Gad88] S.K. Gadia. A homogeneous relational model and query languages for temporal databases. ACM TODS, 13(4):418–448, dec 1988.

[Gad92] S.K. Gadia. A seamless generic extension of SQL for querying temporal data. Technical Report TR-92-02, Computer Science Department, Iowa State University, May 1992.

[GBM83] S.J. Greenspan, A. Borgida and J. Mylopoulos. A Knowledge Representation Approach to Software Engineering: The TAXIS Project. In *Proceedings of the Canadian Information Processing Society*, Ottava, Ontario, May 1983.

[GJ91] N. Gehani and H.V. Jagadish. Ode as an active database: Constraints and triggers. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, pp. 327–336, Barcelona, Spain, September 1991.

[GJS92a] N. Gehani, H.V. Jagadish, and O. Shmueli. Composite event specification in active databases: Model and implementation. In *Proceedings of the Eighteenth International Conference on Very Large Databases*, pp. 327–338, Vancouver, Canada, August 1992.

[GJS92b] N. Gehani, H.V. Jagadish, and O. Shmueli. Event specification in an active object-oriented database. In *Proceedings of ACM SIGMOD 1992 International Conference on Management of Data*, pp. 81–90, San Diego, CA, June 1992.

[GJMS93] N. Gehani, H.V. Jagadish, I.S. Mumick, and O. Shmueli. Temporal Queries for Active Database Support. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[GM91] D. Gabbay and P. McBrien. Temporal logic and historical databases. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, p. 423–430, Barcelona, Spain, September 1991.

[GO93] I. Goralwalla and M.T. Özsu. Temporal Extensions to a Uniform Behavioral Object Model. In *Proceedings of the International Conference on Entity-Relationship Approach*, Dallas, June 1993.

[GST91] F. Grandi, M. R. Scalas, and P. Tiberio. A History-oriented Data View and Operation Semantics for Temporal Relational Databases. Technical Report CIOC-CNR N. 76, Università di Bologna, Italy, Revised April 1993.

[GST93] F. Grandi, M. R. Scalas, and P. Tiberio. History and Tuple Variables for Temporal Query Languages. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[GV85] S.K. Gadia and J.H. Vaishnav. A query language for a homogeneous temporal database. In *Proceedings of the Symposium of Principles of Database Systems*, pp. 51–56, March 1985.

[GY88] S.K. Gadia and C.S. Yeung. A generalized model for a relational temporal database. In *Proceedings of the ACM International Conference on Management of Data*, pp. 251–259, Chicago, IL, June 1988.

[JCG+92] C.S. Jensen, J. Clifford, S.K. Gadia, A. Segev, and R.T. Snodgrass. A glossary of temporal database concepts. ACM SIGMOD Record, 21(3):35–43, September 1992.

[JMR91] C.S. Jensen, L. Mark, and N. Roussopoulos. Incremental implementation model for relational databases with transaction time. IEEE Transactions on Knowledge and Data Engineering, 3(4):461–473, December 1991.

[JMS92] H.V. Jagadish, I.S. Mumick, and O. Shmueli. Events with attributes in an active database. Technical Report 921214-18-TM, AT&T Bell Laboratories, December 1992.

[JMS93] H.V. Jagadish, I.S. Mumick, and O. Shmueli. Sequences with attributes. Submitted for Publication, 1993.

[KLINE93] N. Kline. An Update of the Temporal Database Bibliography. SIGMOD Record, 22(4):66–80, December, 1993.

[KLO81] M.R. Klopprogge. TERM: An Approach to Include the Time Dimension in the Entity-Relationship Model. In *Proceedings of the Second International Conference on the Entity Relationship Approach*, Washington, DC, 1981.

[KRS90] W. Käfer, N. Ritter and H. Schoning. Support for Temporal Data by Complex Objects. In *Proceedings of the 16th International Conference on Very Large Data Bases*, Brisbane, Australia, 1990, pp. 24–35.

[LAN73] B. Langefors. Theoretical Analysis of Information Systems. Student Literature and Auerbach, Lund, Sweden, 1973.

[LJ88] N.A. Lorentzos and R.G. Johnson. Extending relational algebra to manipulate temporal data. Information Systems, 13(3):289–296, 1988.

[LLPS91] G.M. Lohman, B. Lindsay, H. Pirahesh, and K.B. Schiefer. Extensions to Starburst: Objects, types, functions, and rules. Communications of the ACM, 34(10):94–109, October 1991.

[Lor88] N.A. Lorentzos. A Formal Extension of the Relational Model for the Representation and Manipulation of Generic Intervals. Ph.D. thesis, Birkbeck College, University of London, 1988.

[LM93] N.A. Lorentzos and Y.G. Mitsopoulos. IXSQL: An Interval Extension to SQL. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[LS75] B. Langefors and B. Sundgren. Information Systems Architecture. Petrocelli/Charter, New York, 1975.

[MD89] D.R. McCarthy and U. Dayal. The architecture of an active database management system. In *Proceedings of ACM SIGMOD 1989 International Conference on Management of Data*, pp. 215–224, Portland, OR, May 1989.

[MS93] J. Melton and A.R. Simon. Understanding the New SQL: A Complete Guide. Morgan Kaufmann, 1993.

[MS91a] E. McKenzie and R. Snodgrass. An Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. ACM Computing Surveys, 23(4):501–543, December 1991.

[MS91b] E. McKenzie and R. T. Snodgrass. Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions. Technical Report TR–91–15, Department of Computer Science, University of Arizona, Tucson, AZ, August 1991.

[OC87] Oracle Computer, Inc. *ORACLE Terminal User's Guide.* Oracle Corporation, 1987.

[OO84] Z.M. Özsoyoğlu and G. Özsoyoğlu. Summary-table-by-example: A database query language for manipulating summary data. In *Proceedings of the International Conference on Data Engineering*, pp. 193–202, Los Angeles, CA, April 1984.

[P90] N. Pissinou. A Conceptual Framework For Time In Object Databases. Technical Report, Computer Research Institute, University of Southern California, Los Angeles, California, 1990.

[P91] N. Pissinou. Time In Object Databases. Ph.D. Thesis, Department of Computer Science, University of Southern California, December 1991.

[PM92] N. Pissinou and K. Makki. T-3DIS: An Approach to Temporal Object Databases. In *Proceedings of the International Conference on Information and Knowledge Management*, Baltimore, Maryland, November 1992.

[PM93a] N. Pissinou and K. Makki. A Framework for Temporal Object Databases. *Lecture Notes In Computer Science Series, Springer-Verlag*, Volume 752, 1993.

[PM93b] N. Pissinou and K. Makki. Separating Semantics From Representation in a Temporal Object Database Domain. In *Proceedings of the International Conference on Information and Knowledge Management*, Washington D.C., November 1993.

[PM94a] N. Pissinou and K. Makki. A Unified Model and Methodology for Temporal Object Databases. The International Journal on Intelligent and Cooperative Information Systems, 2(2), 1994.

[PM94b] N. Pissinou and K. Makki. A Coherent Architecture for a Temporal Object Database Management System. International Journal of Systems and Software, 23, 1994.

[PM94c] N. Pissinou and K. Makki. Separating Semantics From Representation in a Temporal Object Database Domain. The Journal of Computer Information Systems, Spring 1994.

[Rama93a] K. Ramamritham. Real-Time Databases. Journal of Distributed and Parallel Databases, 1(2):199–226, 1993.

[Rama93b] K. Ramamritham. Time for Real-Time Temporal Databases? In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[Ric92] J. Richardson. Supporting lists in a data model (a timely approach). In *Proceedings of the Eighteenth International Conference on Very Large Databases*, pp. 127–138, Vancouver, Canada, August 1992.

[RS91] E. Rose and A. Segev. TOODM - A Temporal Object-Oriented Data Model with Temporal Constraints In *Proceedings of the 10th International Conference on The Entity-Relationship Approach* October 1991, San Mateo, California, pp. 205–229.

[RS92] E. Rose and A. Segev. A Temporal Object-Oriented Algebra and Query Language. Lawrence Berkeley Laboratory technical report LBL-32013, 1992.

[RS93a] E. Rose and A. Segev. TOOA: A Temporal Object-Oriented Algebra. In *Proceedings of the 7th European Conference on Object-Oriented Programming*, Kaiserslautern Germany, July 1993, Lecture Notes in Computer Science, Springer-Verlag, Vol. 707, pp. 297–325.

[RS93b] E. Rose and A. Segev. TOOSQL – A Temporal Object-oriented Query Language. In *Proceedings of the 10th International Conference on The Entity-Relationship Approach*, Dallas, Texas, 1993.

[SA85] R. Snodgrass and I. Ahn. A taxonomy of time in databases. In *Proceedings of the ACM International Conference on Management of Data*, pp. 236–246, Austin, TX, May 1985.

[SA86] R.T. Snodgrass and I. Ahn. Temporal databases. IEEE Computer, 19(9):35–42, September 1986.

[SA91] S.Y.W. Su and A.M. Alashqur. A Pattern-based Constraint specification Language for Object-oriented Databases. In *Proceedings of IEEE Spring COMPCON 91*, San Francisco, CA, February 1991.

[SC91] S.Y.W. Su and H.H. Chen. A temporal Knowledge Representation Model OSAM*/T and Its Query Language OQL/T. In *Proceedings of the 17th International Conf. on Very Large Data Bases*, Barcelona, Spain, September 1991, pp. 431–442.

[SC93a] S.Y.W. Su and H.H. Chen. Modeling and Management of Temporal Data in Object-oriented Knowledge Bases. In *Proceedings of the Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[SC93b] S.Y.W. Su and H.H. Chen. Temporal Rule Specification and Management in Object-oriented Knowledge Bases. In *Proceedings of the First International Workshop on Rules in Database Systems*, Edingburgh, Scotland, August 1993.

[SGM93] R. Snodgrass, S. Gomez, and E. McKenzie. Aggregates in the Temporal Query Language TQuel. To appear in IEEE Transactions of Knowledge and Data Engineering, 1993.

[SHI81] D.W. Shipman. The Functional Data Model and the Data Language DAPLEX, ACM TODS 6(1):140–173, March 1981.

[SK91] M. Stonebraker and G. Kemnitz. The Postgres Next-generation Database Management System. Communications of the ACM, 34(10):78–93, October 1991.

[SKL89] S.Y.W. Su, V. Krishnamurphy and H. Lam. An Object-oriented Semantic Association Model (OSAM*) for Modeling CAD/CAM Databases. Chapter 17 in Artificial Intelligence: Manufacturing Theory and Practice, S.T. Kumara, A.L. Soyster, and R.L. Kashyap (eds.), Institute of Industrial Engineers, Industrial Engineering and Management Press, Norcross, GA., 1989, pp. 463–494.

[Sno87] R.T. Snodgrass. The Temporal Query Language TQuel. ACM Tods, 12(2):247–298, June 1987.

[Sno93] R.T. Snodgrass. An Overview of TQuel. Chapter 6. Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993, pp. 141–182.

[SPAM91] U. Schreier, H. Pirahesh, R. Agrawal, and C. Mohan. $\mathcal{A}$lert: An architecture for transforming a passive dbms into an active dbms. In *Proceedings of the Seventeenth International Conference on Very Large Databases*, pp. 469–478, Barcelona, Spain, September 1991.

[SS91] Y.M. Shyy and S.Y.W. Su. K: A High-level Knowledge Base Programming Language for Advanced Database Applications. In Proceedings of the ACM International Conference on Management of Data, Denver, CO., May 1991, pp. 338–347.

[SS92] M. Soo and R. Snodgrass. Mixed Calendar Query Language Support for Temporal Constants. TempIS Technical Report 29, Computer Science Department, University of Arizona, Tucson, Arizona, Revised May 1992.

[SSD92] M. Soo, R.T. Snodgrass, C. Dyreson, C.S. Jensen and N. Kline. Architectural Extensions to Support Multiple Calendars. TempIS Technical Report 32, Computer Science Department, University of Arizona, Revised May 1992.

[SSU91] A. Silberschatz, M. Stonebraker, and J.D. Ullman. Database systems: Achievements and opportunities. Communications of the ACM, 34(10):110–120, October 1991.

[TA86] A.U. Tansel and M.E. Arkun. Hquel, a query language for historical relational databases. In *Proceedings of the Third International Workshop on Statistical and Scientific Databases*, July 1986.

[Tan86] A.U. Tansel. Adding time dimension to relational model and extending relational algebra. Information Systems, 11(4):343–355, 1986.

[Tan91a] A.U. Tansel. A Historical Query Language. Information Sciences, 53:101–133, 1991.

[Tan91b] A.U. Tansel. Temporal relational data model. Technical report, Baruch College, CUNY, 1991.

[Tan93] A.U. Tansel. $\mathrm{SQL}^{\mathrm{T}}$: A Temporal Extension to SQL. In *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, Arlington, TX, June 1993.

[TAO89] A.U. Tansel, M.E. Arkun, and G. Özsoyoğlu. Time-by-example query language for historical databases. IEEE Transactions on Software Engineering, 15(4):464–478, April 1989.

[TC83] Tandem Computers. *ENFORM Reference Manual.* Cupertino, CA, 1983.

[TC90] A. Tuzhilin and J. Clifford. A Temporal Relational Algebra as a Basis for Temporal Relational Completeness. In *Proceedings of International Conference on Very Lanrge Databases*, Brisbane, Australia, August 1990.

[TG89] A. Tansel and L. Garnett. Nested historical relations. In *Proceedings of the ACM International Conference on Management of Data*, pp. 284–293, May 1989.

[TC90] A. Tuzhilin and J. Clifford. A temporal relational algebra as a basis for temporal relational completeness. In *Proceedings of the International Conference on Very Large Databases*, pp. 13–23, 1990.

[TCG+93] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass (eds.). *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA, 1993.

[Tuzh89] A. Tuzhilin. Using Relational Discrete Event Systems and Models for Prediction of Future Behavior of Databases. Ph.D. thesis, New York University, October 1989.

[TAL92] B. Theodoulidis, P. Alexakis and P. Loucopoulos. Verification and Validation of Temporal Business Rules. In *Proceeding of the 3rd International Workshop on the Deductive Approach to Information Systems and Databases*, Roses (Catalonia), Spain, September 1992.

[Theo93] C. Theodoulidis. Towards the First Generation of Temporal Information Servers. In *Proceedings of the 1st International Workshop on an Infrastructure for Temporal Databases*, Arlington, Texas, June 1993.

[TLW91] C. Theodoulidis, P. Loucopoulos and B. Wangler. The Entity Relationship Time Model and the Conceptual Rule Language. In *Proceedings of the 10th Conference on the Entity Relationship Approach*, San Mateo, CA, October 1991.

[TWL90] C. Theodoulidis, B. Wangler and p. Loucopoulos Requirements Specification in TEMPORA In *Proceedings of the 2nd Nordic Conference on Advanced Information Systems Engineering (CAiSE90)*, Kista, Sweden, 1990.

[WD92] G. Wuu and U. Dayal. A Uniform Model for Temporal Object-Oriented Databases. In *Proceedings of the International Conference on Data Engineering*, IEEE Computer Society, 1992.

[WD93] G. Wuu and U. Dayal. A Uniform Model for Temporal and Versioned Object-oriented Databases. Chapter 10 of Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993, pp. 230–247.

[WF90] J. Widom and S.J. Finkelstein. Set-oriented production rules in a relational database system. In *Proceedings of ACM SIGMOD 1990 International Conference on Management of Data*, pp. 259–270, Atlantic City, NJ, May 1990.

[WFW75] G. Wiederhold, J.F. Fries and S. Weyl. Structured Organization of Clinical Databases. In *Proceedings of the NCC*, AFIPS Press, Montvale, New Jersey, 1975.

Workshop Call for Position Papers