An Extensible Notation for Spatiotemporal Index Queries

Vassilis J. Tsotras
Dept. of Computer Science
University of California
Riverside, CA 92521
tsotras@cs.ucr.edu

Christian S. Jensen
Dept.of Computer Science
Aalborg University
DK-9220, Aalborg Øst, Denmark
csj@cs.auc.dk

Richard T. Snodgrass
Dept. of Computer Science
University of Arizona
Tucson, AZ 85721
rts@cs.arizona.edu

Temporal, spatial and spatiotemporal queries are inherently multidimensional, combining predicates on explicit attributes with predicates on time dimension(s) and spatial dimension(s). Much confusion has prevailed in the literature on access methods because no consistent notation exists for referring to such queries. As a contribution towards eliminating this problem, we propose a new and simple notation for spatiotemporal queries. The notation aims to address the selection-based spatiotemporal queries commonly studied in the literature of access methods. The notation is extensible and can be applied to more general multidimensional, selection-based queries.

1. Introduction

Data in spatiotemporal databases is associated with multiple dimensions. We will assume that there are three spatial dimensions, namely the x- y- and z-dimension, and two time dimensions, the valid-time and the transaction-time dimension [J+94]. Spatial and temporal data are simply special cases without the temporal and spatial dimensions, respectively. We say that each of these dimensions is recorded by an *implicit* attribute. Next, each piece of data has also a number of regular attributes, termed *explicit* attributes.

Temporal, spatial and spatiotemporal queries are inherently multidimensional as they combine predicates on explicit attributes with predicates on time dimension(s) and/or spatial dimension(s). In its simplest form, a temporal query may combine the employee salary attribute with the valid and transaction time dimensions, as in: “find all employees with salary 50K on January 1st, 1997 as known on March 15th, 1995.” Various temporal query combinations are possible. For example, a query may specify a range of salaries and a single time interval, or simply an interval on each time-dimension. Even more combinations are possible if spatial dimensions are added to form a spatiotemporal query.

The evaluation of a query is greatly affected by the query type and by the presence of various storage structures and indexes. Hence most papers on these topics have focused on a few specific query types.

In the past there has not been any unifying approach to refer to spatiotemporal queries. Rather, individual works used ad hoc approaches to refer to subsets of such queries, sometimes employing conflicting or counter-intuitive terminology, leading to considerable confusion. Some of the most common spatiotemporal queries are *selection*-based. These are queries applied on a single data set (a relation), asking for all data objects (tuples) that satisfy the given query predicate. Selections are the fundamental queries supported by indexes. Even when indexes are employed for joins they are frequently used for making selections. It was actually in indexing papers, we first observed the terminological confusion that subsequently motivated this work. In an attempt to eliminate the problem, we propose a simple notation for selection-based spatiotemporal queries. For simplicity we initially assume a single explicit attribute; however, our notation allows multiple explicit attributes via a simple extension.

We note that there are other basic spatiotemporal queries, like joins, unions, projections, aggregates, constraints and differences, which our notation does not cover. Our aim is not to address all possible kinds of spatiotemporal queries (which would probably create a very complex, hard-to-use notation resembling the creation of a new query language!), but to provide an easy-to-use notation for the most common queries we encountered. Our formalism is at a very high level. We classify spatiotemporal queries based on what the query predicates supply. We do not specify how a query should be expressed, and we do not classify queries based on their expressive power or time/space complexity. Finally, granularity issues are outside the scope of the proposed notation; instead each data value is categorized based on a certain, given granularity of its data domain. While being simple, our notation is also extensible and can be applied to multidimensional queries in general.

Section 2 discusses the many previous approaches to temporal query terminology, and Section 3 presents the “core” of the proposed notation. Section 4 provides examples of various queries to illustrate the use of the notation. Section 5 discusses possible extensions to the core and Section 6 concludes the paper.

2. Previous Terminology

The following discussion and examples focus on terms that have been used in the past to refer to temporal queries. Such terms have been found either as keywords in defining query expressions [OS95] (for example: “as_of,” “intersect,” “timeslice,” “when,” “from_time,” “to_time,” “valid_at,” “valid_during”) or to describe a temporal query supported by some temporal index.

[AS88] compares the performance of various access methods using a benchmark of sixteen temporal queries on four databases (snapshot, valid-time, transaction-time, and bitemporal). All benchmark queries are specified by a declarative (select-from-where) query expression. A similar approach is taken in [AS89].

[LS89, LS93] describe temporal queries by a rectangular region in the (transaction) time-(surrogate) attribute space. The term *timeslice* was used to refer to a query that asks for data as of a given transaction time.

[KS89] discusses the performance of various methods that index interval data. Two queries are identified in the performance evaluation, one that asks for all objects whose time interval intersects a given interval, and another that in addition provides a range predicate on the explicit attribute dimension. In [KS91] queries are defined as random rectangles in the time-attribute space.

[EWK90, EWK93] discuss queries based on the WHEN operator (find all objects whose time interval intersects a given interval), aggregate functions at different time points, and *temporal selections* that specify a condition on the explicit attribute dimension and a time interval.

[DENS95] presents a benchmark for temporal databases that is based on seven valid-time queries, four transaction-time queries and twenty-eight bitemporal queries. These queries are based on various combinations of basic range-range, range-point and point-point operators ("range" refers to attribute ranges or time ranges, and "point" refers to single attribute values or time points).

[MK90, LM91] query *versions*, by specifying the appropriate surrogate range and a version number. Versions are closely related to transaction time. [BGO+96] concentrates on two queries: the *exact match* query which asks for the object with a given surrogate value in a given version, and the *range* query which asks for all objects with surrogates in a given range in a given version. [VV94] discusses snapshot (or timeslice) queries (as in "find all objects alive at some point in the past"), *time-range* queries which are snapshots over a time interval, *attribute-search* queries where an attribute value and a time point are specified, and *attribute-history* queries where an attribute value and a time interval are specified. In the special case where the attribute is a surrogate (i.e., an attribute that does not depend on time), the attribute queries are called *key-search* and *key-history* respectively. [VV97] discusses the *key-search*, *key-range* search ("find objects with surrogates in a given range at some given time point"), *key-history*, *snapshot* (timeslice) and *time-range view* ("find all objects alive during a time interval").

In [JMR91] queries based on *timeslices* are examined; queries are specified in a form of relational algebra. In addition to regular data, queries on data changes (as in "find the employees who changed between two given time instants") are examined.

[TGH95, TK95] present efficient ways to reconstruct past database states. The queries specify a past transaction time and find all objects alive at that time (snapshot or timeslice query). [TK95] also looks at the queries that find all objects alive during a time interval and queries that find the history of a given surrogate.

[GS93] classifies queries of interest into four categories: *ST* queries that query the object surrogate attribute and time, *AT* queries that query other object attributes and time, *T* queries where only time is specified, and *M* ("multidimensional") queries for arbitrary conjunctions of attributes.

In [ST94] the worst case performance of various temporal indexes is examined using three basic temporal queries: the *pure-timeslice* (find all objects whose intervals contain a given time point), the *range-timeslice* (as the pure-timeslice, but with an additional surrogate range predicate), and the *pure-key* query that finds all the time intervals associated with a given surrogate. A similar terminology was adopted in [R97], where in addition the term *pure-key-timeslice* was used for the query that specifies one surrogate and one time point.

Valid-time database queries are similar to queries in computational geometry. The simplest query that asks for a set of dynamic intervals that contain a given valid-time point is termed the *dynamic interval management* problem in [KRVV93]. The term *external dynamic 2-dimensional range* searching was also used. Valid-time queries are also discussed in [NDK96] where three kinds of interval queries are examined, namely the *containment*, *inclusion* and *intersection* queries that find all objects whose valid-time intervals contain, include or intersect a given interval.

[KTF95] addresses bitemporal queries and introduces the terms *bitemporal pure-timeslice* (which asks for all objects whose valid interval includes a valid-time point and transaction interval includes a transaction-time point), and *bitemporal range-timeslice* (where in addition a range on the attribute domain is specified). There is also discussion about the more complex bitemporal query where an interval is specified in both the valid and transaction-time dimensions and a range on the attribute dimension. [NDE96] calls this query the *general bitemporal query*. [KTF97] addresses various bitemporal queries and adopts the proposed terminology presented here.

It is clear that much confusion already exists! At times the same query has been given different names (like snapshot, timeslice, pure-timeslice) or the same term (e.g., timeslice) has been used to describe different queries depending on the application. A coherent terminology is thus needed.

3. The Proposed Query Notation

We assume that each piece of spatiotemporal data has associated a number of explicit attributes, a valid-time attribute, a transaction-time attribute, and three spatial attributes. We propose the following basic ("core") notation to classify spatiotemporal queries: *Expl_attr // X_dimension/ Y_dimension/Z_dimension // Valid/Transaction*. Double slashes ("//"'s) separate the explicit attribute(s), the spatial attributes and the temporal attributes. The idea to use separate entries per explicit/implicit attribute was inspired by the notation used for queueing system models, e.g., (*M/M/*1 systems) [K75].

3.1 The Core Notation

A BNF for the proposed notation follows:

```
<spatiotemporal query> ::= <expl. attr> "//" <space qualifier> "//"
                                            <time qualifier>

<spatial query>     ::=  <expl. attr> "//" <space qualifier>

<temporal query>    ::=  <expl. attr> "//" <time qualifier>

<expl. attr>        ::=  <qualifier> | <qualifier> "/" <expl. attr>

<space qualifier>   ::=  <x qualifier> "/"<y qualifier> "/"<z qualifier>
<x qualifier>       ::=  <qualifier>
<y qualifier>       ::=  <qualifier>
<z qualifier>       ::=  <qualifier>

<time qualifier>    ::=  <valid> "/" <transaction>
<valid>             ::=  <qualifier>
<transaction>       ::=  <qualifier>

<qualifier>         ::=  V          (single Value: point)
                         | R        (Range: time interval)
                         | S        (Set: set of intervals)
                         | *        (any value)
                         | –        (not applicable)
```

The above notation specifies which entries are involved in the query and in what way. The various values a qualifier can take are explained below.

V : In an explicit attribute entry, *V* represents a single attribute value provided by the query. In the valid and transaction entries it corresponds to a single time instant (slice). In the spatial entries it corresponds to the values of a point's spatial coordinates. *V* stands for "single Value."

R : In the explicit attribute entry, *R* corresponds to a specified range of attribute values. In a temporal entry it represents a continuous time interval. In a spatial entry it represents an interval on the spatial coordinate. *R* is an abbreviation for "Range."

S : In an explicit attribute entry, *S* corresponds to a set of ranges of attribute values given in the query. In a temporal entry it represents a set of time intervals (or equivalently, a set of specific time instants). The letter "*S*" comes from "Set of intervals." In a spatial entry it represents a set of intervals on the spatial coordinate.

* : The occurrence of "*" as a qualifier value indicates a trivially true predicate, i.e., that any value is accepted in this entry to satisfy the query (equivalently, it indicates the absence of a selection predicate on this attribute).

– : The occurrence of "–" as a qualifier value indicates that the data does not include the associated attribute, making the entry inapplicable.

3.2 Discussion

In an effort to keep the core notation simple, we have chosen to include only notation for the types of queries that are considered frequently. However, it is extensible. Additional aspects of temporal, spatial or spatiotemporal queries can be covered through extensions that preserve the core notation as a special case. Section 5 provides examples of some of the possible extensions.

The core notation considers only conjunctive queries, i.e., the slash-separated terms are ANDed together. It is possible to extend the notation to also allow explicit disjunctions and negations.

Query predicate qualification on the explicit attribute uses exact value match or value containment. Thus, if a point is specified on the explicit attribute entry, the returned data should have their attribute matching the given value. If an explicit attribute range is instead specified, the returned data should have attribute values inside the given range. If a set of ranges is specified, the results should have attribute values inside this set of ranges.

Qualification for a temporal or spatial dimension can be more general here because data is associated with an interval and not a single value. The core notation makes the common assumption that query qualification on a temporal (spatial) dimension is based on time-instant (point) containment and interval (interval) intersection. For example, a query that specifies a time instant on the transaction-time entry and a time interval on the valid-time entry asks for data whose transaction-time interval contains the given time instant and whose valid-time interval intersects the given time interval. If the query specifies a set of intervals in the valid-time entry, it asks for data whose valid-time interval intersects any of the intervals in the set. It is possible to also introduce extensions supporting any other of the variety of existing predicates involving periods, points, or both, e.g., CONTAINS, CONTAINED_IN, PRECEDES, and MEETS (for example, "find all objects with intervals before a query time point/ interval" or "find all objects with intervals contained in a given interval" [BO95, NDK96]).

The *S* qualifier is clearly more powerful than *V* or *R* (for example a set can be specified as a union of ranges) and can thus be used in place of the other two. However, the notation includes all three qualifiers because of their frequent use in spatiotemporal queries.

The proposed notation distinguishes among three kinds of attributes, namely explicit, temporal and spatial attributes. We followed this approach in order to identify the basic parts in a spatiotemporal query, i.e., the temporal (valid time, transaction time), spatial (latitude, longitude and elevation) and explicit attributes. A similar approach can also be applied to more general multidimensional queries. However, note that while temporal dimensions have a deep conceptual theory distinguishing them from other dimensions, in a general multidimensional query there may not be a firm delineation between spatial and explicit dimensions. If spatial dimensions are of importance (for example if special spatial predicates are used in the query), the space qualifier part should be included in the query notation (and maybe extended to include the new predicates). Non-spatial dimensions can then be represented as multiple explicit attributes. The core notation allows for adding explicit attributes; they should be separated by a "/" (single slash) and precede any time or spatial dimensions. If spatial dimensions are not treated differently (or if they do not exist in the query of interest) then all non-temporal attributes should be represented as regular explicit attributes (and the space qualifier is dropped from the notation). For example, if queries are based on two explicit attributes, Name and Salary, a temporal *V/R//V/** query indicates a specific name, a range of salaries, a specific valid time and any transaction time.

Finally, granularity issues are not part of the notation. Instead, we assume a certain, given granularity for each of the involved domains of the underlying data; any value that is coarser than that granularity denotes a range, and any value that is the same (or of finer) granularity denotes a single value or point.

4. Examples

We first illustrate the proposed notation with various temporal query examples on a bitemporal data set. Examples for the spatiotemporal domain follow. The time-instant (point) containment and interval intersection qualifications are assumed.

4.1 Temporal Queries

The queries are over the relation Employee(Name, Salary) and are based on a single explicit attribute (the surrogate attribute, Name) and the temporal dimensions. Thus three entries are needed: *Expl_attr//Valid/Transaction*. Each query is followed by its representation in the proposed notation and a discussion.

Query 1: What salary does Bob make? (Equivalently, what salary does Bob currently make, as best known?)

Notation: *V//V/V*

The explicit attribute qualifier is a single value (Bob); the valid-time qualifier is also a single value (now); and the transaction-time qualifier is a single value (now). The result will include all objects with a given name and whose valid-time interval contains the given valid time (now) and whose transaction-time interval contains the current time (now). On a conventional relation we would have: *V//–/–* because the relation includes neither valid nor transaction time. On a valid-time relation: *V//V/–,* while on a transaction-time relation: *V//–/V*.

Query 2: List the current employees whose name starts with "B".

Notation: *R//V/V*

This query uses a range qualifier on the explicit attribute.

Query 3: List the current employees.

Notation: **//V/V*

All values are allowed for the explicit attribute. All objects that contain the given valid time instant (now) as of the given transaction-time instant (now) will be returned. This is an example of the *bitemporal pure-timeslice* of [KTF95]. On a transaction-time database, a **//–/V* query is an example of a *pure-timeslice* query [ST94, TK95] or the *snapshot* of [VV97].

Query 4: List the history of employees whose name starts with "B".

Notation: *R//*/V*

The transaction-time qualifier is a single value (now, as best known), but all values of the valid time are relevant. Note that this is a different query than *R//–/V,* which is on a transaction-time database (no valid-time exists). The latter is an example of the *range-timeslice* query [ST94].

Query 5: What was Bob's salary history for 1990–1996?

Notation: *V//R/V*

Here an interval of valid time is requested. The transaction-time qualifier is again a single value (now, as best known). Similarly, we may ask for Bob's salary history for 1990–1996 as best known on (transaction time) Jan. 1, 1997.

Query 6: What was Bob's salary for 1990–1992 and 1994–1996?

Notation: *V//S/V*

Information over a set of intervals, or a temporal element [GV85], is requested.The transaction time is again a single value (now, as best known).

Query 7: When was Bob's salary recorded (perhaps erroneously)?

Notation: *V//V/**

This is a single-value query over the explicit attribute (Bob) and valid time (now); all transaction times are requested. Note that this is a different query than *V//V/–* as the former is on a bitemporal database (and any transaction-time qualifies), while the latter is for a valid-time database (where no transaction-time exists and hence there are no past database states to be queried).

Query 8: What changes were made during 1996 to Bob's salary for May 11, 1997?

Notation: *V//V/R*

An interval of transaction time is requested. The *V* on the valid entry is the single value of current time.

Query 9: What changes were made during 1996 to Bob's salary history?

Notation: *V//*/R*

The word "history" implies that we are interested in all valid times, with transaction time restricted to the interval of 1996.

4.2 Spatiotemporal Queries

To illustrate spatiotemporal queries we use a bitemporal two-dimensional spatial relation Precipitation(Kind, Amount) with a surrogate of Kind. Queries are based on the surrogate, temporal and spatial dimensions. Hence six entries are used: *Expl_attr//X_dimension/Y_dimension/Z_dimension//Valid/ Transaction*. However since there are only two spatial dimensions in this database, the third spatial entry (*Z_dimension*) is occupied by "–".

Query 10: What is the rainfall at Bob's house today?

Notation: *V//V/V/–//V/V*

Here "rainfall" is a single-valued qualifier on the explicit attribute (Kind). "Bob's house" is a single-valued qualifier in each of the x and y dimensions; the z dimension (altitude above sea level) is not relevant, since the relation records only two dimensions. The valid-time and transaction-time qualifiers are restricted to the current time.

Query 11: What is the amount of precipitation at Bob's house today?

Notation: **//V/V/–//V/V*

The precipitation can be of any Kind, hence any value is accepted on the explicit attribute.

Query 12: What is Tucson's rainfall today?

Notation: *V//R/R/–//V/V*

"Tucson" indicates a range in both the x and y axes, i.e., a rectangular region in two-space; "rainfall" indicates a single value for the explicit attribute. The word "today" implies single-valued qualifiers in both valid and transaction time.

Query 13: What is the history of Tucson's rainfall this year?

Notation: *V//R/R/–//R/V*

The terms "history" and "this year" imply a range qualifier in valid time.

Query 14: What is the rainfall today for the North American and European continents?

Notation: *V//S/S/–//V/V*

Two continents imply qualifying over a set of two-dimensional spatial rectangles, hence element is indicated.

Query 15: What changes were made to the history of Tucson's precipitation, and when were they made?

Notation: **//R/R/–//*/**

Here "changes" implies all of transaction time; "history" implies all of valid times and "precipitation" implies all explicit attribute values.

4.3 Comparison with Previous Terminology

The table in the next page presents the correspondence between some of the previous temporal query terms and the new notation.

5. Extensibility Issues

The core notation can be extended in various ways to address additional aspects of queries in the specialized settings where this is necessary. One such extension/generalization of the core notation addresses queries with general temporal predicates (i.e., not only with point containment and interval intersection). In particular, the valid and transaction time qualifiers of the core notation can be extended to:

<valid> ::= <temporal_predicate> <qualifier> | <qualifier>

<transaction> ::= <temporal_predicate> <qualifier> | <qualifier>

Various combinations of temporal_predicates and qualifiers are possible. For example, Allen's [A83] well-known predicates or derivations of these (that often are more useful in practice) can be allowed in place of temporal_predicate. Doing this, we associate a single temporal predicate with each temporal qualifier in a query. As an example, assume the CONTAINS temporal_predicate is denoted by *c* and the FOLLOWS temporal_predicate by *f*. Then consider the query:

Query 16: "What was Bob's salary history after 1990?"

Notation: *V//fV/cV*.

Here a single value is given for valid time and for the transaction-time (as best known now is assumed).

Depending on the selection criteria, other more complex predicates can also be incorporated [V82, L86].

Another possible extension is the inclusion of explicit disjunctions and negations. In a similar way, we can add before each qualifier a predicate that specifies whether the qualifier is to be used in the query specification with an AND, OR or NOT.

Other extensions could be envisioned, for operations such as joins, and for specifying the result desired (e.g., all attributes or just a few attributes). Another interesting extension is to allow for constraints of variables with respect to other variables.

6. Conclusions

The absence of a common terminology for temporal, spatial and spatiotemporal queries has led to much confusion in the past. In an effort to eliminate this problem, we introduced a query notation that associates with each portion of the

Previous Term	*Citation*	*Proposed*
exact match, attribute-search, key-search	[BGO+96, VV94, VV97]	*V//–/V*
range, key-range, range-timeslice	[BGO+96, VV97, LS89, ST94]	*R//–/V*
snapshot, timeslice, pure-timeslice	[VV94, VV97, ST94, TK95, R97]	**//–/V*
attribute-history, key-history, pure-key	[VV94, VV97, ST94]	*V//–/R*
time-range, time-range view	[VV94, VV97, TK95]	**//–/R*
dynamic interval management, interval intersection	[KRVV93, NDK96]	**//V/–*
bitemporal pure-timeslice	[KTF95]	**//V/V*
bitemporal range-timeslice	[KTF95]	*R//V/V*
general bitemporal query	[KTF97, NDE96]	*R//R/R*

selection criteria (be it one or more explicit attribute values, spatial dimensions, or temporal dimensions) a descriptive entry. Query qualifications in the core notation included point containment and interval intersection. Our notation is simple, general and extensible, while it captures many of the distinctions made by previous terminology, which was inconsistent and ambiguous. It can also be easily applied to multidimensional queries in general.

Acknowledgments

We would like to acknowledge the many colleagues who commented on this notation. In particular we thank Leopoldo Bertossi, Alex Delis, Piero Fraternali, H.V. Jagadish, Mario Nascimento, Carme Martin, Bongki Moon, Dimitris Plexousakis, Sakti Pramanik, Betty Salzberg, Letizia Tanca, and Paolo Terenziani. This research was supported in part by NSF grants IRI-9509527 and IRI-9632569, by the Danish Technical Research Council through grant 9700780, and by the CHOROCHRONOS project, funded by the European Commission DG XII Science, contract no. FMRX-CT96-0056.

References:

[A83] J.F. Allen, “Maintaining Knowledge about Temporal Intervals,” *Commun. of the ACM*, Vol. 26, No. 11, pp 832-843, 1983.

[AS88] I. Ahn and R. T. Snodgrass, “Partitioned Storage for Temporal Databases,” *Information Systems*, Vol. 13, No. 4, pp 369-391, 1988.

[AS89] I. Ahn and R. T. Snodgrass, “Performance Analysis of Temporal Queries,” *Information Sciences*, Vol. 49, pp 103-146, 1989.

[BGO+96] B. Becker, S. Gschwind, T. Ohler, B. Seeger and P. Widmayer, “An Asymptotically Optimal Multiversion B-tree,” *Very Large Data Bases Journal*, Vol. 5, No. 4, pp 264-275, 1996.

[BO95] T. Bozkaya and M. Özsoyoglu, “Indexing Transaction-Time Databases,” Tech. Rep. CES-95-19, Case Western Reserve Univ., 1995.

[DENS95] M. H. Dunham, R. Elmasri, M. A. Nascimento and M. Sobol, “Benchmarking Temporal Databases-A Research Agenda,” Tech. Rep. CSE-95-20, Southern Methodist Univ., 1995.

[EWK90] R. Elmasri, G. Wuu and Y. Kim, “The Time Index: An Access Structure for Temporal Data,” *Proc.* 16*th VLDB Conf.*, pp 1-12, 1990.

[EWK93] R. Elmasri, G. Wuu and V. Kouramajian, “The Time Index and the Monotonic B^+-tree,” in A.Tansel, J.Clifford, S.K.Gadia, S.Jajodia, A.Segev, R.T.Snodgrass (eds.), *Temporal Databases: Theory, Design, and Implementation*, Benjamin/Cummings, pp 433-456, 1993.

[GV85] S.K. Gadia, J.H. Vaishnav, “A Query Language for a Homogeneous Temporal Database,” *Proc. ACM IGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pp 51-56, 1985.

[GS93] H. Gunadhi and A. Segev, “Efficient Indexing Methods for Temporal Relations,” *IEEE Trans. on Knowledge and Data Eng.*,Vol. 5, No.3, pp 496-509, 1993.

[JMR91] C. S. Jensen, L. Mark and N. Roussopoulos, “Incremental Implementation Model for Relational Databases with Transaction Time,” *IEEE Trans. on Knowledge and Data Eng.*,Vol. 3, No.4, 1991.

[J+94] C.S. Jensen, editor et. al., “A Consensus Glossary of Temporal Database Concepts,” *ACM SIGMOD Rec.*,Vol. 23, No.1, pp52-64, 1994.

[K75] L. Kleinrock, *Queueing Theory*, John Wiley & Sons, 1975.

[KRVV93] P.C. Kanellakis, S. Ramaswamy, D.E. Vengroff, J.S. Vitter, “Indexing for Data Models with Constraints and Classes,” *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pp. 233-243, 1993.

[KS89] C. Kolovson and M. Stonebraker, “Indexing Techniques for Historical Databases,” *Proc.* 5*th IEEE ICDE Conf.*, pp 127-137, 1989.

[KS91] C. Kolovson and M. Stonebraker, “Segment Indexes: Dynamic Indexing Techniques for Multidimensional Interval Data," *Proc. ACM SIGMOD Conf.*, pp 138-147, 1991.

[KTF95] A. Kumar, V. J. Tsotras and C. Faloutsos, "Access Methods for Bitemporal Databases", *International Workshop on Temporal Databases.* In *Recent Advances in Temporal Databases,* J. Clifford, A. Tuzhilin (eds.), pp. 235-254, Springer-Verlag, 1995

[KTF97] A. Kumar, V. J. Tsotras and C. Faloutsos, "Designing Access Methods for Bitemporal Databases," Univ. of Maryland, UMIACS TR-97-24, 1997. To appear *IEEE Trans. on Knowledge and Data Eng*.

[L86] P. Ladkin, "A taxonomy of interval relations," *Proc. of National Conf. on Artificial Intelligence (AAAI*'86), 360-366, 1986.

[LM91] S. Lanka and E. Mays, "Fully Persistent B^+ Trees," *Proc. ACM SIGMOD Conf. on the Management of Data*, pp 426-435, 1991.

[LS89] D. Lomet and B. Salzberg, "Access Methods for Multiversion Data," *Proc. ACM SIGMOD Conf.*, pp 315-324, 1989.

[LS93] D. Lomet and B. Salzberg, "Transaction-Time Databases," in A.Tansel, J. Clifford, S. K. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass (eds.), *Temporal Databases: Theory, Design, and Implementation*, Benjamin/Cummings, pp 388-417, 1993.

[MK90] Y. Manolopoulos and G. Kapetanakis, "Overlapping B^+ Trees for Temporal Data," *Proc. of 5th JCIT Conf.*, Jerusalem, Israel, 1990.

[NDE96] M. Nascimento, M. H. Dunham and R. Elmasri, "M-IVTT: A Practical Index for Bitemporal Databases," *Proc. DEXA'*96, Zurich.

[NDK96] M. Nascimento, M. H. Dunham and V. Kouramajian, "A Multiple Tree Mapping-Based Approach for Range Indexing," *Journal of the Brazilian Computer Society*, Vol. 2, No. 3, April 1996.

[OS95] G. Özsoyoglu and R. T. Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 7, No. 4, pp 513-532, Aug. 1995.

[R97] S. Ramaswamy, "Efficient Indexing for Constraint and Temporal Databases," *Proc. ICDT*, 1997.

[ST94] B. Salzberg and V. J. Tsotras, "A Comparison of Access Methods for Temporal Data," to appear in *ACM Computing Surveys;* also available as TimeCenter Tech. Report TR-18, (`http://www.cs.auc.dk/general/DBS/tdb/TimeCenter/publications2.html`).

[TGH95] V. J. Tsotras, B. Gopinath and G. W. Hart, "Efficient Management of Time-Evolving Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 7, No. 4, pp 591-608, Aug.1995.

[TK95] V. J. Tsotras and N. Kangelaris, "The Snapshot Index, an I/O-Optimal Access Method for Timeslice Queries," *Information Systems, An International Journal*, Vol. 20, No. 3, pp 237-260, 1995.

[V82] M. Vilain, "A System for Reasoning About Time," *Proc. of National Conf. on Artificial Intelligence (AAAI*'82*)*, 197-201, 1982.

[VV94] R. M. Verma and P. J. Varman, "Efficient Archivable Time Index: A Dynamic Indexing Scheme for Temporal Data," *Intern. Conf. on Computer Systems and Education*, pp 59-72, 1994.

[VV97] P. J. Varman and R. M. Verma, "An Efficient Multiversion Access Structure," *IEEE Trans. on Knowledge and Data Eng.*,Vol. 9, No. 3, pp 391-409, 1997.