A Notation for Spatiotemporal Queries

Vassilis J. Tsotras, Christian S. Jensen, and Richard T. Snodgrass

April 8, 1997

TR-10

A TIMECENTER Technical Report

Title	A Notation for Spatiotemporal Queries
Author(s)	Vassilis J. Tsotras, Christian S. Jensen, and Richard T. Snodgrass
Publication History	April 1997. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector)
Michael H. Böhlen
Renato Busatto
Heidi Gregersen
Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector)
Anindya Datta

Individual participants
Curtis E. Dyreson, James Cook University, Australia
Kwang W. Nam, Chungbuk National University, Korea
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, University of South Florida, USA
Andreas Steiner, ETH Zurich, Switzerland
Vassilis Tsotras, Polytechnic University, New York, USA
Jef Wijsen, Vrije Universiteit Brussel, Belgium

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors, The Rune alphabet (second phase) has 16 letters. They all have angular shapes and lack horizontal lines because the primary storage medium was wood. However, runes may also be found on jewelry, tools, and weapons. Runes were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract:

Temporal, spatial and spatiotemporal queries are inherently multidimensional, combining predicates on time dimension(s) with predicates on explicit attributes and/or several spatial dimensions. In the past there was no consistent way to refer to temporal or spatiotemporal queries, thus leading to considerable confusion. In an attempt to eliminate this problem, we propose a new notation for such queries. Our notation is simple and extensible and can be easily applied to multidimensional queries in general.

1. Introduction

In spatiotemporal data each individual piece of data has associated a region in a multi-dimensional space with space and time dimensions. We will assume that there are three space dimensions, namely an x-dimension, a y-dimension, and a z-dimension. We will also assume that there are two time dimensions, namely one valid-time and one transaction-time dimension (bitemporal data). We say that each of these dimensions is recorded by an *implicit* attribute. Next, each piece of data has a number of regular attributes, termed *explicit* attributes. Spatial and temporal data are simply special cases without the temporal and spatial dimensions, respectively.

Temporal and spatiotemporal queries are inherently multidimensional as they combine predicates on time dimension(s) with predicates on explicit attributes and/or several spatial dimensions. In its simplest form, a query on a bitemporal database may combine the employee salary attribute with the valid and transaction time dimensions: “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 a temporal query.

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. Here we propose a general notation for spatiotemporal queries. For simplicity we initially assume a single explicit attribute; however our notation allows multiple explicit attributes that are covered by a simple extension to it. The proposed terminology and notation is restricted to queries on a single data set, i.e., we do not address join queries over two or more data sets.

Section 2 discusses the many previous approaches to temporal query terminology while section 3 presents the structure of the proposed terminology. Section 4 provides examples of various queries to illustrate the notation.

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 addressed by some temporal index.

[AS88] compares the performance of various access methods using a benchmark of sixteen temporal queries on four databases (snapshot, valid, transaction 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 time-key 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 query that in addition provides a range predicate on the explicit key dimension. In [KS91] queries are defined as random rectangles in the time-key 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 key dimension and a time interval.

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

[MK90, LM91] query *versions*, by specifying the appropriate key range and a version number. Versions are closely related to transaction time. [BGO93] concentrates on two queries: the *exact match* query which asks for the object with a given key in a given version, and, the *range* query which asks for all objects with keys in a given key 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.

[VV95] discusses the *key-search*, *key-range* search ("find objects with keys 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 query specifies a past transaction time and finds all objects alive at that time (snapshot or timeslice query). [TK95] also looks at the query that finds all objects alive during a time interval.

[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 when 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 key range predicate), and the *pure-key* query that finds all the time intervals associated with a given key. A similar terminology was adopted in [R97], where in addition the term *pure-key-timeslice* was used for the query that specifies one key and one time point.

Valid-time database queries are similar to problems 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, 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 key 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 standard query terminology is thus needed.

3. The Proposed Query Notation

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

<spatiotemporal query> ::= <key> "//" <space qualifier> "//" <time qualifier>

<spatial query> ::= <key> "//" <space qualifier>

<temporal query> ::= <key> "//" <time qualifier>

<key> ::= <qualifier> | <qualifier> "/" <key>

<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> ::= *S* (slice: single value)
| *R* (range: continuous time interval)
| *E* (element: 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:

S In a key entry, *S* represents a single attribute value provided by the user. In the valid and

transaction entries it corresponds to a single time instant. In the spatial entries it corresponds to the values of a point's spatial coordinates. *S* is short for "single value."

R In the key 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."

E In a key entry *E* corresponds to a set of ranges of attribute values. In a temporal entry it represents a set of time intervals (or equivalently, a set of specific time instants). The letter "E" comes from the term "temporal element", which is a set of time 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.

\- The occurrence of - as a qualifier value indicates that the data does not include this qualifier, making the entry inapplicable.

The notation is extensible, in that it is easy to extend it to cover additional aspects of temporal, spatial or spatiotemporal queries in a manner that yields an extended notation which preserves the current notation as a special case. Next, we consider only conjunctive queries, i.e., the dash-separated terms are ANDed together. It is possible to introduce also explicit disjunctions and negations. We have chosen to include only notation for the types of queries that are considered frequently. Notation for additional types of queries may subsequently be introduced in the specialized settings where this is necessary.

Query predicate qualification on the key attribute uses exact value match or value containment. Thus, if a point is specified on the key entry, the returned data should have their key attribute matching the given value. If a key range is instead specified, the returned data should have key values inside the given range. Qualification for a temporal or spatial dimension can be more general since in that dimension a piece of data is associated with an interval and not a single value. For simplicity we assume 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. It is possible to also introduce any other of the variety of existing predicates involving periods, points, or both, e.g., CONTAINS, CONTAINED_IN, PRECEDES, and IS_SIMILAR_TO (for example, "find all objects with intervals before a query

time point/interval" or "all objects with intervals contained in a given interval" [BO95, NDK96]).

If multiple explicit attributes are used, they should be separated by a '/' (single slash) and precede the time or spatial dimensions. For example, if queries are based on two explicit attributes, name and salary, a temporal *S*/*R*//*S*/* query indicates a specific name, a range of salaries, a specific valid time and any transaction time.

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

These queries are over the relation Employee(Name, Salary) with a key of Name. Queries are based on a single explicit attribute (the key attribute Name) and the temporal dimensions, thus three entries are needed, *Key//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: *S//S/S*

The key 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 find all objects with the given name, whose valid-time interval contains the given valid time (now) as of the given transaction-time instant (now). On a conventional relation we would have: *S//-/-*, because the relation does not include either valid nor transaction time. On a valid-time relation: *S//S/-* while on a transaction-time relation: *S//-/S.*

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

Notation: *R//S/S*

This query uses a range qualifier on the key.

Query 3: List the current salary for all employees.

Notation: **//S/S*

All values are allowed for the key. 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 **//-/S* query is an example of a *pure-timeslice* query [ST94, KT95] or the *snapshot* of [VV95].

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

Notation: *R//*/S*

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//-/S* which is on a transaction-time database (no valid-time exists). The later is an example of the *range-timeslice* query [ST94].

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

Notation: *S//R/S*

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: *S//E/S*

Information over a set of intervals, or a temporal element, 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: *S//S/**

This is a single-value query over the key (Bob) and valid time (now); all transaction times are requested. Note that this is a different query than *S//S*/- as the former is on a bitemporal database (and any transaction-time qualifies) while the later is for a valid-time database (where no transaction-time exists and hence there are not past database states that can be queried).

Query 8: What changes were made to Bob's current salary during 1996?

Notation: *S//S/R*

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

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

Notation: *S//*/R*

"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 key of Kind. Hence each object can be queried on its key, temporal and spatial dimensions. All entries are used: *Key//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: *S//S/S/-//S/S*

Here "rainfall" is a single value qualifier on the key (Kind). "Bob's house" is a single value 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 qualifiers are restricted to current in valid time and as best known in transaction time.

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

Notation: **//S/S/-//S/S*

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

Query 12: What is Tucson's rainfall today?

Notation: *S//R/R/-//S/S*

"Tucson" indicates a range in both the x and y axes. "rainfall" indicates a single value for the attribute key. "today" implies single value qualifiers in both valid and transaction time.

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

Notation: *S//R/R/-//R/S*

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: *S//E/E/-//S/S*

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

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

Notation: **//R/R/-//*/**

Here "changes" imply all transaction times; "history" implies all valid times; "precipitation" implies all key values.

4.3 Comparison with Previous Terminology

The following table presents the correspondence between some of the previous temporal query terms and the new notation:

Previous Term	*Citation*	*Proposed*
exact match, attribute-search, key-search	[BGO+93, VV94, VV95]	*S//-/S*
range, key-range, range-timeslice	[BGO+93, VV95, LS89, ST94]	*R//-/S*
snapshot, timeslice, pure-timeslice	[VV94, VV95, ST94, TK95, R97]	**//-/S*
attribute-history, key-history, pure-key	[VV94, VV95, ST94]	*R//-/R*
time-range, time-range view	[VV94, VV95, TK95]	**//-/R*
dynamic interval management, interval intersection	[KRVV93, NDK96]	**//S/-*
bitemporal pure-timeslice	[KTF95]	**//S/S*
bitemporal range-timeslice	[KTF95]	*R//S/S*
general bitemporal query	[KTF97, NDE96]	*R//R/R*

5. Conclusions

The absence of a common terminology for temporal, spatial or spatiotemporal queries has led in much confusion in the past. In an effort to eliminate this problem we introduced a query notation that associates with each portion of the selection criteria (be it one or more explicit attribute values, spatial dimensions, or temporal dimensions) a descriptive entry. Query qualifications 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.

R e f e r e n c e s :

[AS88] I. Ahn and R. T. Snodgrass, “Partitioned Storage for Temporal Databases”, *Information Systems*, Pergamon Press, 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+93] B. Becker, S. Gschwind, T. Ohler, B. Seeger and P. Widmayer, “On Optimal Multiversion Access Structures”, *Proc. Symp. on Large Spatial Databases*, in Lecture Notes in Computer Science, Vol. 692, pp 123--141, Singapore 1993.

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

[DENS95] M. H. Durham, 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 Conference on Very Large Databases*, 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, and R.T. Snodgrass (eds.), *Temporal Databases: Theory, Design, and Implementation*, Benjamin/Cummings, pp 433--456, 1993.

[GS93] H. Gunadhi and A. Segev, "Efficient Indexing Methods for Temporal Relations", *IEEE Transactions on Knowledge and Data Engineering*, 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 Engineering*, Vol. 3, No. 4, pp 461--473, 1991.

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

[KS89] C. Kolovson and M. Stonebraker, "Indexing Techniques for Historical Databases", *Proc.* 5*th IEEE Intern. Conf. on Data Engineering*, pp 127--137, 1989.

[KS91] C. Kolovson and M. Stonebraker, "Segment Indexes: Dynamic Indexing Techniques for Multi-dimensional Interval Data", *Proc. ACM SIGMOD Conf. on the Management of Data*, 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 in *IEEE Trans. on Knowledge and Data Engineering.*

[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. on the Management of Data*, 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* 5*th JCIT Conf.*, Jerusalem, Israel, Oct.22-25, pp 491--498, 1990.

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

[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 Conf.*, 1997.

[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.

[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.

[VV95] P. J. Varman and R. M. Verma, “An Efficient Multiversion Access Structure”, Tech. Rep. TR-9518, Dept. of Electr. and Comp. Engineering, Rice Univ. To appear in *IEEE Trans. on Knowledge and Data Engineering.*