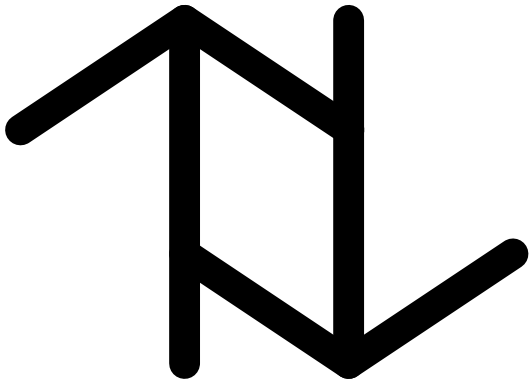

An Extensible Framework for Spatio-Temporal Database Applications

Glaucia Faria, Claudia Bauzer Medeiros and Mario A. Nascimento

April 23, 1998

TR- 27

A TIMECENTER Technical Report

Title	An Extensible Framework for Spatio-Temporal Database Applications
Author(s)	Glaucia Faria, Claudia Bauzer Medeiros and Mario A. Nascimento
Publication History	April 1998. A TIMECENTER Technical Report.

TIMECENTER Participants

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

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

Individual participants
Curtis E. Dyreson, James Cook University, Australia
Kwang W. Nam, Chungbuk National University, Korea
Mario A. Nascimento, State University of Campinas and EMBRAPA, Brazil
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, University of South Florida, USA
Andreas Steiner, ETH Zurich, Switzerland
Vassilis Tsotras, University of California, Riverside, USA
Jef Wijsen, Vrije Universiteit Brussel, Belgium

For additional information, see The TIMECENTER Homepage:
URL: <http://www.cs.auc.dk/research/DBS/tdb/TimeCenter/>

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, all of which have angular shapes and lack horizontal lines because the primary storage medium was wood. Runes may also be found on jewelry, tools, and weapons and were perceived by many as having magic, hidden powers.

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

Abstract

There is a wide range of scientific application domains requiring sophisticated management of spatio-temporal data. However, existing database management systems offer very limited (if any at all) support for managing such data. Thus, it is left to the researchers themselves to repeatedly code this management into each application. Besides being a time consuming task, this process is bound to introduce errors and increase the complexity of application management and data evolution. This paper addresses this very point. We present an extensible framework, based on extending an object-oriented database system, with kernel spatio-temporal classes, data structures and functions, to provide support for the development of spatio-temporal applications. Even though the paper's arguments are centered on geographic applications, the proposed framework can be used in other application domains where spatial and temporal data evolution must be considered (e.g., Biology).

1 Introduction

Data used in scientific applications are almost always time-dependent. Not only does the reliability of the applications depend on the validity of the data, but also the data analysis themselves can only be performed on temporally compatible values. The need for temporal data management, prompted the creation of a novel research field – that of temporal database management systems. The requirements of scientific applications for temporal data management have been taken into consideration by researchers in the field, to include, for instance, handling of temporal series and management of data versions (e.g., results of experiments). In addition, the proliferation of results in that area has led to recent efforts in standardization (e.g., [JCE+94, Sno95]).

Another important issue for a large set of scientific applications is that of handling spatial data. At first, this was only reflected in the need for displaying these data, spurring research in interfaces (e.g., [KV92]). Nowadays, spatial data handling has become a whole new issue in database systems research and development. Examples of such applications are found in Medicine, Biology or in Earth sciences (e.g., [BZ91, AKS+93, A+94b, BX94]). Again, as in the case of temporal data management, prototypes and systems have been built, usually geared towards geometric data processing, where the spatial description of the world is represented in terms of point, lines and polygons (the so-called *vectorial data representation*. Most of the recent prototypes have been built using as a basis either object-oriented or object-relational database systems, which provide at the same time the functionality of a DBMS and the flexibility and extensibility of the object paradigm. Nevertheless, almost all such systems reported in the literature are time-less, i.e., while they offer several facilities to handle spatial data, very few, if any, any aimed towards temporal data.

The need for integrated handling of time and space dimensions for scientific data has long been recognized (e.g., [FJP90]). An important issue in this sense is the proposal of adequate data models, which have been usually geared towards Geosciences (e.g., [Peu93, Lan93b, Gou93, NTE92, A+94a]), but a more general purpose implementation is yet to be reported, as shows a recent bibliography of spatio-temporal systems [SS94]. Reported prototypes are usually restricted to only one dimension – that is, dealing with temporal database management or with spatial database management, but not both (e.g., [GR93, SV92, AS91, SK96]). At most, the handling of time in a spatial context is limited to version management (e.g., [MJ94, AW96]), or restricted to a limited set of operations [SW95].

The difficulties in implementing a spatio-temporal system are due to the new problems that are encountered when combining the problems found for handling each dimension. First, there is the issue of defining a minimal set of operators which allow navigation through space and time. Here, questions such as compatibility and orthogonality of the operator set are arised. Next, there is the problem of an adequate data model which will support the operator set. Another issue is that of implementation, i.e., what data structures and algorithms to implement in order to support operations and model, while at the same time ensuring adequate performance. Data loading and integration is itself another complicating factor since scientific data are often unstructured, hence its insertion into a fixed schema database may cause problems for query processing [Abi97]. Finally, there are issues related to usability, e.g., how flexible and extensible is the system, and interface design.

For these reasons, in most cases, the combined handling of space and time is left to the scientists, who have to embed special purpose code for managing spatio-temporal data into their applications. On a spatial system, timestamp management is left to the application; conversely, users of temporal databases must themselves code appropriate spatial data modules. Besides being a time consuming task, this process is error-prone and bound to increase the complexity of application management and data evolution.

This paper provides a partial solution to some of these issues. We present an extensible database framework, based on enhancing the O2 [BDK92] object oriented DBMS with a minimal set of spatial, temporal and spatio-temporal primitives. These primitives can be used in a wide spectrum of scientific applications. The framework was constructed having two main issues in mind – the operator set, and a kernel set of application-independent spatio-temporal database classes. Different applications can be built using the framework by merely specializing and composing the kernel classes, and inserting the appropriate data into the database.

Furthermore, even though the application domain we considered was that of Geosciences, several other domains can also take advantage of the framework by implementing the appropriate application classes, and applying the set of primitive operators. The main contributions of this paper are thus the following:

- specification of a minimal set of spatio-temporal operators for managing and querying scientific data, in 2D, for vectorial data types;
- implementation of these operators into the framework in such a way that they can be applied orthogonally, thus leaving space for query optimization, and allowing users to extend the set;
- providing a kernel set of database classes which, combined with the operators (implemented either as methods or functions) allow development of scientific applications.

The rest of this paper is structured as follows. Section 2 presents our space and time primitives. Section 3 describes our data model, and its implementation into the kernel classes. Section 4 gives an example of typical queries that have been implemented in the framework, for a geographic application (land use management). Section 5 presents the related work. Section 6 sums up the paper and presents opportunities for future research. Examples will be presented using the syntax of the O2 system, in which the framework was implemented.

2 Space and Time Primitives

This section defines the basic set of operators used to formulate typical spatio-temporal queries in databases. This set ban be seen as a "tool box" from which further spatial, temporal or spatio-temporal operators may be defined. All operators will be defined keeping in mind that data are managed according to the object oriented paradigm – i.e., data is stored in objects, organized in class hierarchies. Query formulation in scientific databases can use spatial, temporal, or spatio-temporal operators. The operands of these queries can be scalars, objects (temporal, spatial or spatio-temporal) or sets thereof.

In our framework, spatio-temporal data is organized into spatial-temporal objects. These objects encapsulate three basic components, which are complex objects themselves: conventional, spatial and temporal. The spatial component describes the spatial properties of an object, which may vary according to the representation used (e.g., in geographic data, geometric description varies according to the scale used). The temporal component describes the temporal properties of the object, and is here considered to be formed by time values. After [JCE+94], we consider that time values can be represented by a chronon, an interval or temporal element. A temporal element is a finite set of intervals. Throughout the paper we use the terms time values and timestamp as equivalent ones.

Assume A is a spatio-temporal object. Then, A = ⟨ CT, SpT, T ⟩, where CT is the timestamped conventional component of A; SpT is the timestamped spatial component of A; and T is the temporal component of A, representing its valid time. The valid time of an object is the time interval in which its semantic is true in the modeled reality [JCE+94]. An object's component (spatial or conventional) can assume many values along time. Thus, SpT is a set of tuples $\langle Sp_i, T_i \rangle$, where Sp_i is the value of the spatial component at T_i. Similarly, CT is a set of tuples $\langle C_i, T_i \rangle$, where C_i is the value of the conventional component at T_i. Note that all T_i do not intersect and are contained in T.

For instance, consider a Road whose single conventional component is its name. Assume furthermore that the road's name has, at different times, assumed values 'Rio-SP' and 'Dutra', and the road's geometry, represented by object GL has also evolved temporally. Then, a simplified specification of the road could be:

⟨ { (Rio-SP, [t1-t5]), (Dutra, [t6-NOW]) }, { (GL1, [t1-t2]), (GL2, t3), (GL3,[t4-NOW]) }, [t1-NOW) ⟩

this denotes, for instance, that at time t3 the road was called 'Rio-SP', and was described by geometry GL2. GL itself is a complex object; for instance, the road may be represented by a polyline pl1 in a given scale s1, and by a set of polygons pol1 in another scale s2. In this simple case, GL1= { (s1, pl1), (s2,pol1) } – note that each polyline and set of polygons are complex objects as well. Computing the state of the road requires performing *temporal*

intersections, i.e., computing intersection of the temporal values. When several spatio-temporal phenomena are involved, spatial operations only make sense when computed for attributes that co-exist in a certain time period.

Naturally, spatial and temporal objects are special cases of spatio-temporal objects without temporal and spatial components, respectively. Then, if A is a spatial object, A = ⟨[C], Sp⟩, where C is the conventional component of A and Sp is the spatial component of A. If A is a temporal object, then A = ⟨[CT], T⟩, where CT is the timestamped conventional component of A and T is its valid time.

In what follows we denote an operator by OPER (A,B,C,...), where OPER is the operator's name and A, B, C, ... are its operands.

2.1 Spatial Operators

Following [PS94], we classify spatial operators into orientation, metric and topological operators. We only consider vectorial data (represented by points, lines and polygons, or sets thereof). All operators are applied to the spatial component of database objects. This spatial component is retrieved by a special operator, named SP, defined next.

Operator to retrieve spatial component – The SP operator returns the spatial component of its parameter A (a database object). This component is a complex object that contains the spatial representation of A, and is composed of non-spatial attributes (to hold specific information about the representation, such as scale or measurement system) and a spatial attribute that enumerates the geometric objects that describe A's geometry.

Orientation Operators – The orientation operators verify whether there exists a specific orientation relationship between two sets of geometric objects [PS94], or deal with relative order in space [PTS94]. For simplicity we used the following operators (based on definitions in [TP95] – note that several other operators can be built upon these):

- NORTH (A,B): returns true if all elements of A have a North relationship with all elements of B, and
- EAST (A,B): defined similarly as NORTH but with respect to East direction.

Metric Operators – The metric operators return a scalar that represents an intrinsic property of the analyzed objects [Cil96]. We consider the following operators:

- AREA (A): computes the area occupied by object A (set of polygons)
- LENGTH (A): returns the length of A (polyline)
- PERIMETER (A): returns the perimeter of a set of polygons.
- DISTANCE (A,B): calculates the distance between two sets of geometric objects (points, lines, or polygons).

Topological Operators – The topological operators return true if there is a specific topological relationship between two set of geometric objects. Topological relationships are those which do not change with transformations of scale, translation and rotation. [CdFvO93] showed that all binary topological relashionships can be expressed by five operators (*disjoint, touch, cross, overlap, in*) and two functions (*from, to*). We base our primitive operators on their work, extending them to sets of objects (instead of pairs of objects), as follows:

- DISJOINT (A,B): returns true if all elements of A have a disjoint relationship with all elements of B; otherwise it returns false.
- TOUCH (A,B), OVERLAP (A,B), CROSS (A,B): return true if there is a pair *a, b* from A and B having, respectively, a touch, overlap, cross relationship; otherwise it returns false.
- IN (A,B): returns true if for all $a \in$ A there exists a $b \in$ B such that *a in b*, otherwise it returns false.

2.2 Temporal Operators

Temporal operators can be unary or binary, returning time values, boolean values or objects. Operands are time values, or objects (either temporal or spatio-temporal).

Temporal unary operators return a time value:

- TV (A): returns A's temporal component.
- BEGIN (A), END (A): returns respectively A's start time and end time.
- DAY(A), MONTH(A) and YEAR(A): A is a chronon and as such can be casted in the format *day/month/year*. Those operators return respectively *day*,*month* and *year*.
- TWHEN (A): returns A's valid time.

Temporal binary operators can return a boolean or time value, an object or a set of objects satisfying the temporal predicate. Our operators are based on [Sno95, S$^+$94, VBH96, BVH96], They search for temporal relationships between two time values (or timestamps). They can also be used to express other temporal relationships, e.g., those defined in [All83, RP92, GS89]. In the following definitions $T_S(.)$ and $T_E(.)$ denote, respectively, the start time and end time of the operand (which is a time value); and t is a chronon.

- T_BEFORE (A,B): returns true if T_E (A) $< T_S$ (B); false otherwise.
- T_OVERLAPS (A,B): returns true if $\exists$t (t $\in A \wedge t \in B$); false otherwise.
- T_EQUAL (A,B) returns true if A = B; false otherwise.
- T_CONTAINS (A,B) returns true if $\forall t \in B$, t $\in A$; false otherwise.
- T_MEETS (A,B) returns true if T_S (B) $- T_E$ (A) = 1, where 1 represents a chronon; false otherwise.

Other binary operators are the following:

- INTERVAL (s, e): returns an interval having start time **s** and end time **e**.
- T_INTER (A,B): returns a time value corresponding to the temporal intersection between A and B.
- VSLICE (A, T): returns A's states valid at time T.

2.3 Spatio-temporal Operators

These operators extend the spatial operators, defined in Section 2.1 by adding a temporal dimension. Their operands are one or two spatio-temporal objects; or a spatio-temporal object and a spatial object.

Location-temporal Operator – This operator, denoted ST_SP (A,T), returns the spatial representations of object A valid at time T. That is, it returns a list of tuples $\langle Sp_i, T_i \rangle$, where all T_i are disjoint and cointained within T.

Orientation-temporal Operators – These operators return a boolean value indicating whether there is or not a specific orientation relationship between two objects (A and B), for all intervals of temporal intersection (in the time domain T) between the spatial components of A and B. We consider the operators ST_NORTH and ST_EAST.

Metric-temporal Operators – These operators may involve one or two objects. In the first case the operator has two parameters – an object A and a time value T. In particular, we have defined: ST_AREA (A,T), ST_LENGTH (A,T) and ST_PERIMETER (A,T). They return, respectively, pairs ⟨area, time⟩, ⟨length, time⟩, ⟨perimeter, time⟩, for each state of the spatial component of A valid at T. The second case is represented by the ST_DISTANCE (A,B,T) operator returns the distance between the spatial components of A and B (valid at time T) for all intervals of temporal intersection between these components.

Topologic-temporal Operators – These operators return a list of boolean values associated to the respective intervals in which the location of object A and the location of object B are relatively constant during the time T. That is, these boolean values indicate whether there is a specific topological relationship between the locations in the considered time: ST_DISJOINT, ST_TOUCH, ST_OVERLAP, ST_INSIDE and ST_CROSS.

2.4 Other Operators

Besides the spatial, temporal and spatio-temporal operators above, we must define a few additional operators which are necessary to allow the formulation of spatio-temporal queries. These are:

- IS_S_TRUE (A): returns true if there is at least a true value in the list of boolean values A; false otherwise.
- IS_A_TRUE (A): returns true if all values in the list of boolean values A are true; false otherwise.
- GT (A,v), GE(A,v), LT(A,v), LE(A,v), EQ(A,v), NE(A,v): receive as a parameter a list A of numeric values returning true if all elements of the list are, respectively, greater than, greater than or equal, lesser than, lesser than or equal, equal, or different from the value v; false otherwise.

2.5 Queries

The operators form a basic set from which queries can be computed. We classify these queries as spatial, temporal or spatio-temporal queries. Spatial queries deal with spatial relationships between objects. They can return spatial attributes or attributes calculated from them, and are formulated in terms of spatial operators. Temporal queries are those containing at least one temporal operator and no spatial operator. They deal with temporal relationships between objects, besides dealing with the valid time of an object. Spatio-temporal queries deal with spatial relationships along time. They may return spatial components or metric values valid at a specific time, or valid time of spatio-temporal relationships. These queries involve spatial and temporal predicates, using spatio-temporal operators.

Next we show examples of representative queries, using a simple example which will be seen again in Section 4. Consider a geographic database containing classes with the following entities: farms, rivers, roads and telephone poles. We assume that farms may contain rivers and poles; and can be intersected by rivers or roads. For simplicity we assume that all needed classes have been properly defined.

Recall that the implementation of spatial operators depends on the stored spatial representation (just as the implementation of temporal operators depends on the time granularity considered). We assume that rivers and roads are represented by polylines, farms by polygons and poles by points.

In what follows examples of spatial (SQ), temporal (TQ) and spatio-temporal queries (STQ1, STQ2 and STQ3), are presented in an informal manner followed by a format similar to relational calculus using some of the operators described earlier.

SQ: "Which are the rivers inside farm X"

r | r ∈ River ∧ INSIDE (SP(r), SP(X))

TQ: "Which are the farms that existed in 01/03/1990."

f | f ∈ Farm ∧ T_OVERLAPS (TV (f), 01/03/1990)

STQ1: "What is the foreseen area for expansion of farm A from 01/01/98 to 01/01/99?"

ST_AREA (A, INTERVAL(01/01/98, 01/01/99)

STQ2: "In which time intervals, from 01/01/80 to 31/12/95, was road B adjacent to farm A?"

TWHEN (ST_TOUCH (A, B, INTERVAL (01/01/80, 31/12/95)))

STQ3: "Which were the farms crossed by some road in 01/12/96."

f | f ∈ Farm ∧ ∃r (r ∈ Road ∧ IS_A_TRUE (ST_CROSS (f,r,01/02/96)))

Notice that these examples show that one can orthogonally apply temporal and spatial operators. Furthermore, the time involved may be of several types.

3 The Kernel Classes

Our framework is based on providing users with a kernel of spatial and temporal class hierarchies, from which scientific applications can be built using the object oriented concepts of inheritance, composition and polymorphism (for method redefinition). This section presents this kernel (classes and methods), as well as a basic set of functions which allow implementing the operators. The kernel is based on a modification of the model proposed in [Bot95], where spatial properties are geared towards geographic applications.

3.1 Classes and Methods

The kernel class hierarchies are rooted at three main classes (see Figure 1, which uses the OMT notation [R+91]): Time, which allows associating different temporal properties to objects; Location, which allows associating different spatial properties to objects; and Conventional (not shown in the Figure), which represents all classes of objects that have neither spatial nor temporal properties. In this paper, we are concerned with the spatial and temporal components of objects, and their interactions, and thus will not discuss aspects related to the implementation of time for conventional components, since our primary focus is on the implementation of spatio-temporal facilities.

Since we are interested in Geosciences applications, spatial (and spatio-temporal objects) are generalized by class GeoObject, and spatial properties are described in a class called Location (which basically generalizes geometries). Notice that for other application domains the classes Location and GeoObject would receive other names, but the functionality would remain basically the same.

The SP operator, defined previously, is specified as a method (*sp*) of GeoObject, being redefined in its subclasses.

Subclasses of Time (Event, Interval and TempElement) are used to represent the different kinds of timestamps users may want to associate to temporal objects. The objects of class TempElement are composed of a list of objects of class Interval. An Interval object is composed of a tuple of Event objects which, in turn, allow representing time instants (chronons). This hierarchy has methods *year, month, day, begin, end, t_before, t_equal, t_overlaps, t_contains, t_meets* and *t_inter*, which implement the temporal operators of the same name defined in Section 2.2.

Temporal objects (instances of TempObject) are those with a temporal component (i.e., object of the class Time). At any time, users can transform a non-temporal object (Conventional or SpatialObject) into a temporal object by applying composition operators with objects from the Time hierarchy. The operators VSLICE and TV, defined on section 2.2 are specified as methods *vslice* and *tv* of the class TempObject.

The class SpatialObject represents the non-temporal spatial objects, that is, objects that have a spatial component (from class SP) and no Time component, whereas SpatioTempObject instances have both a temporal and a spatial (SPT) component. This independence of a spatial object from its spatial description (generalized by class Location) allows objects to share spatial properties. For instance, a phone pole and a transformer installed on the pole may be represented by a point having the same (x,y) coordinates. This helps maintenance of spatial integrity constraints.

The spatial component, represented by a SpatialObject so, is in fact a list of objects $geom_1, geom_2$, etc from class Geom. Each Geom object $geom_i$, in turn, corresponds to one spatial representation of so, which, as we have seen in the previous section, is composed of non-spatial attributes to keep representation-specific information and a spatial attribute to define the representation geometry. Consider again the Road instance, but assume there is no temporal variation, e.g.,

⟨ { (Rio-SP) }, { (GL1) }⟩, where GL1= { (s1, pl1), (s2,pol1) }

in this case, 'Rio-SP' is an instance of Conventional, GL1 is an instance of SpatialObject, and the tuple (s1,pl1) is an instance of Geom.

The geometric description itself is described by the class hierarchy rooted at class Obj_Geom – in the example, pl1 and pol1 are instances respectively of classes Line and Polygon. A special method – *select_geometry* – is used to select, among the spatial representations of an object, the adequate Geom object appropriate to the users' needs.

An instance of Polygon is defined as a list of geographic coordinates, which describe the polygon boundary; an instance of Line is defined by a tuple of coordinates, that represent the line extremes; an instance of class Point is a point in the bidimensional space $I\!R^2$. The operators LENGTH, AREA, DISTANCE, DISJOINT, TOUCH, INSIDE, OVERLAP, CROSS, NORTH and EAST, are implemented by methods of the same name of the Obj_Geom hierarchy.

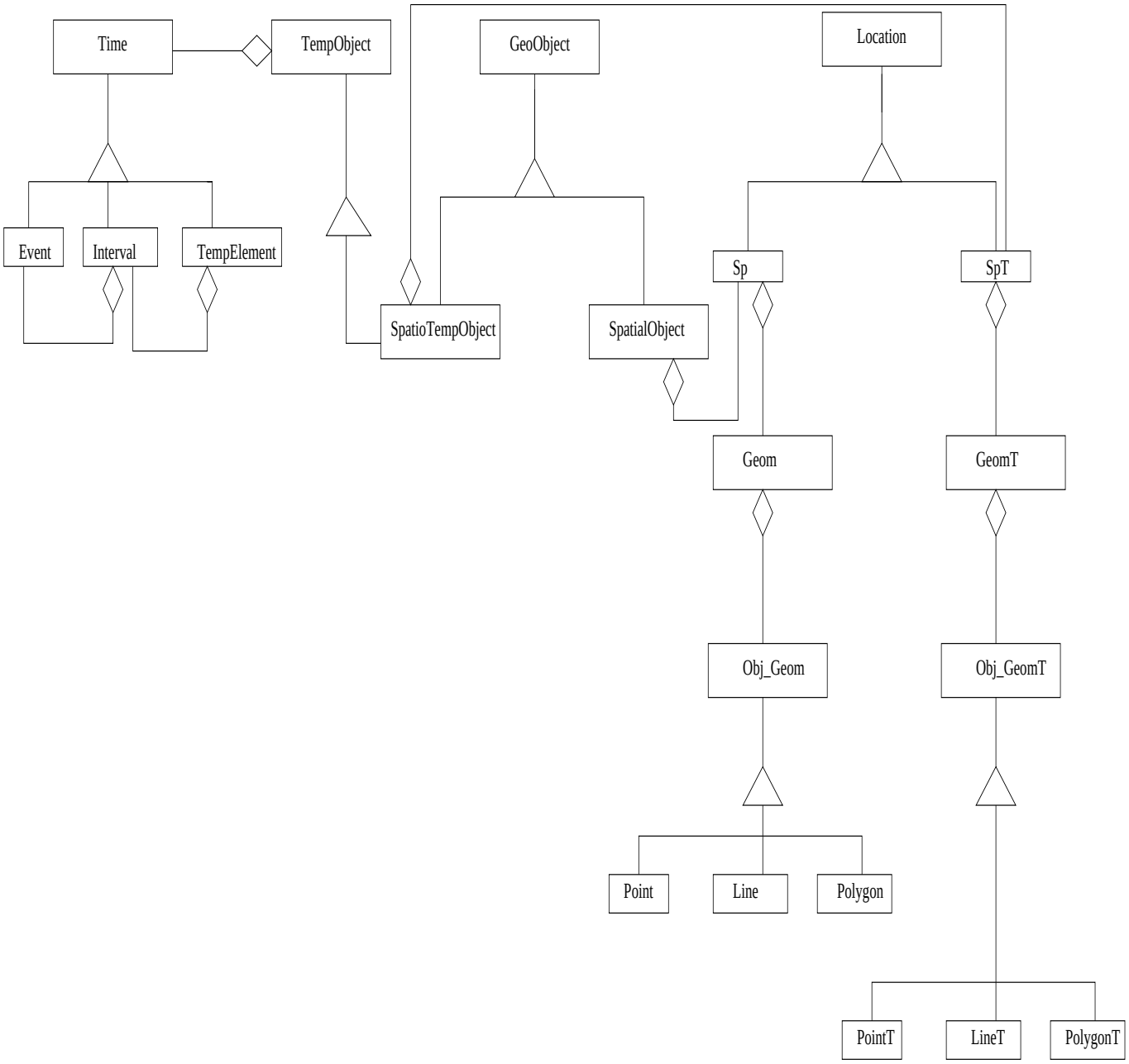

Figure 1: Basic data model

Special attention was given to the implementation of metric and topologic methods (*length, area, distance, disjoint, touch, inside, overlap*), which were implemented using algorithms by O'Rourke [O'R94], and Preparata and Shamos [PS85]. We point out that the use of computational geometry techniques to implement spatial operators is not new in the literature (e.g., [KBS91]). The difference, however, is that here they can be combined with the temporal properties, which increases implementation complexity.

While the class SpatialObject describes non-temporal spatial objects, the class SpatioTempObject represents spatio-temporal objects, i. e., whose components change along time. The characteristics of SpatioTempObjects are similar to those of SpatialObjects as regards spatiality. However, they differ from the former because they also have a temporal component. This is subsumed by classes SPT and GeomT. Like class Geom, GeomT corresponds to a spatial representation, with the difference that its attributes are "temporalized", i.e., a GeomT object is built from composition of Geom objects and objects of the Time hierarchy. This allows keeping the evolution of spatial representations through time. Obj_GeomT and SPT classes follow the same reasoning, and for this reason will not be described here.

We remark that we separate the spatial properties into Geom and Obj_Geom classes to allow differentiating a spatial property Geom from its representation Obj_Geom. The same applies to GeomT and Obj_GeomT.

3.2 Implementation of the Spatio-temporal Operators

We have already remarked on an example of implementation of a spatial operator (method *touch*). Here, we comment on the implementation of the spatio-temporal operators. The operator ST_SP, defined on section 2, is mapped to methods *st_sp* and *loc* of SpatioTempObject. Method *loc* returns the location of an instance of SpatioTempObject at the time instant *t* (an Event object) whereas method *st_sp* returns the *temporal evolution* of the location of an instance of SpatioTempObject at a time *t* (Event, Interval, TempElement). In other words, *loc* returns one spatial description, whereas *st_sp* returns a list of tuples $\langle$ Sp, Time $\rangle$.

The spatio-temporal operators – ST_LENGTH, ST_DISJOINT, etc., defined on section 2.3, are implemented as methods of SpatioTempObject. The unary spatio-temporal operators ST_LENGTH, ST_PERIMETER and ST_AREA are methods of SpatioTempObject invoked with a time parameter (t), being implemented in three main steps:

1. The method *st_sp* creates a list of tuples $\langle G, T \rangle$, where G is a set of Geom objects that compose the object's geometry, and T is the associated valid time, which must be contained in the specified time parameter (t).
2. A (length/ perimeter/ area) value is computed for the geometry G component of each tuple.
3. The final result is a list of tuples $\langle$value,$T \rangle$, where for each value is associated the corresponding valid time T.

The implementation of binary spatio-temporal operators – ST_DISJOINT, ST_DISTANCE, etc. – comprehends basically the following steps, when applied to sets of SpatioTempObject (or SpatialObject) X and Y:

1. For each object in X and in Y is created a list of tuples $\langle G, T \rangle$, in the same way as for the unary operators. We suppose that when parameters objects of type SpatialObject then they are considered to have undetermined valid time.
2. The valid times T of each element of these lists are processed for obtaining *intervals* of temporal intersection, using method *t_inter* of the Time hierarchy.
3. For each interval of temporal intersection, the G objects valid at this interval are processed for the corresponding spatial function (*disjoint, distance, etc.*, being calculated a result.
4. A list with the final results is returned.

Take in consideration again the previous example of farms A and B, and now consider the temporal dimension, and let us examine the *disjoint* operator. It is now used repeated times for the pairs of spatial descriptions that coexist in each interval of temporal intersection of the two objects. Consider, for example, the spatio-temporal evolution presented in Figure 2, *st_disjoint (B,[t1,t3])* will return the values $\langle$false, t1$\rangle$, $\langle$true, t2$\rangle$, $\langle$false, t3$\rangle$.

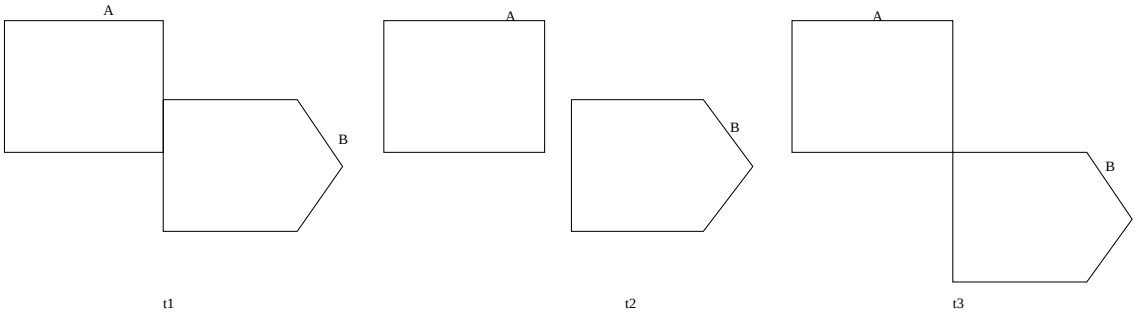

Figure 2: Farms A and B geometries between times t1 and t3

3.3 Functions

Some of the operators presented in section 2 were implemented as functions: orientation NORTH and SOUTH; metric AREA, LENGTH and DISTANCE; topological DISJOINT, INSIDE, TOUCH, CROSS and OVERLAP; temporal TWHEN and INTERVAL; boolean IS_A_TRUE and IS_S_TRUE; and comparators GT, GE, EQ, NE, LT and LE.

The functions implementing the spatial operators invoke methods of the Geom and GeomT hierarchies having the same name. Consider, for example, the function *disjoint*:

disjoint (A: list (Obj_Geom), B: list (Obj_Geom))

this function verifies if for all element *objA* of A and all element *objB* of B,

$objA \rightarrow disjoint\ (objB)$

4 Using the Implemented Database

The kernel described in the previous section forms the basis for developing applications that require integrated data analysis in the spatial and temporal dimensions. This section describes an example of use of the implemented database for a geographic application. The problems and query specification were elaborated based on examples given by researchers from the Faculty of Agricultural Engineering at the State University of Campinas.

4.1 Problem Overview

The application's objective is to analyze land use in some areas in Brazil for agricultural purposes. This can be used to, for example, detect unproductive farms, or monitor deforestation. Also, one can classify farms according to existing infrastructure (next to or crossed by roads, served by electricity, etc.), places where to build roads, and so on.

4.2 Class Definitions

Our running example in this section describes part of the database involving spatio-temporal classes Farms, Roads and Poles. Farms are composed of Agricultural divisions (spatial units for crop rotation, soil preservation etc). Each AgDivision may have had several crops over time, and distinct production data. The spatial representation may vary with time, and so may the conventional attributes. Using our framework, classes Farm, Road and Pole are derived from SpatioTempObject using inheritance. The corresponding class definitions are shown next, using O2's syntax [BDK92].

```
class Farm inherit SpatioTempObject type
  tuple (name: string,
         hist_division: list (tuple (lst_divisions: list(AgDivision),
                                     t: time)))
end;
```

```
class AgDivision inherit SpatioTempObject type
  tuple (name: string,
         hist_crop: list (tuple (crop: string, t: time)),
         hist_production: list (tuple (production: real, t: time)))
end;

class Road inherit SpatioTempObject type
    tuple (name: string,
           hist_road_type: list (tuple (road_type: string, t: time)))
end;

class Pole inherit SpatioTempObject type
    tuple (number: integer,
           type: string)
end;

name Farms: set (Farm);
name AgDivisions: set (AgDivision);
name Roads: set (Road);
name Poles: set (Pole);
```

We recall that multiple representations are allowed. The adequate representation is retrieved by the method *select_geometry* (or, in the spatio-temporal domain, *t_select_geometry*) when a query is processed. To simplify the example, we assume that the representation used for a Farm is a list of polygons; for an Agricultural division, a polygon; for a Road, a polyline represented by a list of lines; and for a Pole, a point.

4.3 Query Examples

This section uses OQL (O2's query language) to show how some typical queries from the application considered can be written by using the operators implemented. It it important to note however, that our implemented system does not automatically generate OQL code given a user (natural language) query – i.e., we do not provide a user-friendly interface. Automatic generation of OQL code given a spatio-temporal query language is nevertheless an interesting open area, which requires much further research.

Each query was selected to show a specific characteristic of the framework. Queries involving spatial variables were defined to show examples of typical spatial queries; queries involving temporal data exemplify typical temporal data management. Query 1 is a spatial query, query 2 is strictly temporal, and the remaining queries are spatio-temporal, presented in increasing order of complexity.

Query 3 is an example of a spatial aggregate query that computes a spatial value (area), where the aggregate is computed over time. Query 4 is a typical example of a spatial window query (find all objects within a given window). However, temporal predicates are applied at the same time. Remark furthermore that query 4 shows the use of two previously posed queries (q1, conventional, and q2, spatial) as parameters of a temporal operator (*twhen*), which, in turn, is applied to a spatial – *st_inside* – and a temporal – *interval* – operator. This shows that the operators can be applied progressively in an orthogonal way, thereby allowing the progressive construction of more complex operators and queries. The orthogonality of operators is further illustrated in query 5. In this last example, the query combines spatial and conventional predicates to a temporal interval.

1. "Select the farms that contain electricity poles."

   $f \mid f \in \text{Farm} \wedge p \in \text{Pole} \wedge p.\text{type} = \text{“electricity”} \wedge \text{INSIDE}(SP(p), SP(f))$

   ```
   select f
   from f in Farms, p in Poles
   where p.kind = "electricity" and inside(p->sp, f->sp)
   ```

2. “Select the agricultural divisions that used to cultivate sugar cane before 01/07/1997.”

   d | d ∈ AgDivision ∧ e ∈ d.hist_crop ∧ d.crop = “sugar cane” ∧ T_BEFORE (BEGIN (TV(e), 01/07/1997)

```
select distinct d
from d in Divisions, e in d.hist_crop
where e.crop = "sugar cane" and
      e->t->begin->t_before(event("1/7/1997"))
```

3. “What was the area occupied by farms from 01/01/97 to 01/01/98?”

   ST_AREA (f, INTERVAL (01/01/97,01/01/98)) | f ∈ Farm

```
select tuple (farm: f,
              area:f->st_area(interval("01/01/97", "01/01/98")))
from f in Farms
```

4. “When did farm A lie in the rectangle delimited by the coordinates (1,1),(15,40)?”

   TWHEN (ST_INSIDE (X, TO_OBJ ((1,1), (15,1), (15,40), (1,40))), INTERVAL (Beginning, Now))

```
define q1 as element (select f
                      from f in Farms
                      where f.name = "A")
define q2 as to_obj (list (tuple(x:1.0,y:1.0),
       tuple(x:15.0,y:1.0),tuple(x:15.0,y:40.0),
       tuple(x:1.0,y:40.0)))
twhen (q1->st_inside(q2, interval(Beginning,Now())))
```

5. “Select agricultural divisions that cultivated sugar cane and were adjacents on 01/02/97.”

   q1 = d | d ∈ Divisions ∧ T_OVERLAPS (TV(d), 01/02/1997) ∧ ∃c (c ∈ d.hist_crop ∧ T_OVERLAPS (TV(c),01/02/1997) ∧ c.crop = “sugar cane”)

   q2 = d1, d2 | d1 ∈ q1 ∧ d2 ∈ q1 ∧ IS_A_TRUE (ST_TOUCH (d1,d2,01/02/1997))

```
define q1 as select d
             from d in Divisions
             where d.t->t_overlaps(event("01/02/1997"))
define q2 as select distinct d
       from d in q1,
            c in d.hist_crop
       where d.hist_crop != list() and
             c.t->t_overlaps (event("01/02/1997"))
             and c.crop= "sugar cane"
select tuple (Division1: d1, Division2: d2)
from d1 in q2,
     d2 in q2
where d1 != d2 and
      is_a_true(d1->st_touch(d2,event("01/02/1997")))
```

5 Related Work

As argued before, recent research have devoted more attention to incorporating time into spatial databases (e.g., [ATSS93, BVH96, Skj96, CT95, Lan93a, PW94, SB97, BJS97, Cho], but there is relatively few published work on actual implementation of spatio-temporal models. We are aware of only two such works [PW94, SB97]. The

majority of the proposed approaches are focused towards modeling instead of implementation. Careful analysis of some of those proposals show them to unfeasible to implement, requiring further modification to the proposed models. Indeed, this was the case with the model upon which we base our implementation [Bot95].

Most spatio-temporal models have an associated query language (e.g., [BJS97, BVH96]. Alternatively, our prototype manages spatio-temporal data through the definition and implementation of a basic set of classes and operators on a OODB. Steiner and Norrie's [SN97] research also uses such an approach, though dealing only with temporal data.

Peuquet and Wentz's work [PW94] uses a time based representation for spatio-temporal data and implements temporal operators. However, they do not represent vector and MATRIX ?? data representation for geographical data, which is rather common on current GISs.

Becker et al [BVH96] describe a temporal geographical model (T/OOGDM) which is based on the geographical model OOGDM. However, the implemented prototype does not provide temporal support. OOGDM deals with raster and vector data, 2D and 3D data e define geometric and topological operators. T/OOGODM extends OOGDM by supporting valid and transaction time. Even though it defines temporal operators it does not define spatio-temporal operators. Furthermore the authors do not temporalize the geometric objects.

Bohlen et al [BJS97] define a spatio-temporal query language, named STSQL, based on the traditional relational model. STSQL's model supports multiple transaction and valid times, but does not treat space and time in a totally integrated manner. The actual implementation of STSQL, as originally proposed, is a rather complex, if feasible, task.

Skjellaug and Berre's model [SB97] extends spatial data with time, capturing the temporal nature of an object as a whole as well as of all its individual properties - very much like what we propose in the paper. The model defines the type T_Object and the qualifies (*temporal()* which temporalizes its operand. T_Object, like the type TempObject we introduce in this paper, introduces the concept of lifespan. However, this time span has a fixed representation (a list of temporal intervals). The authors consider three types of lifespan: transaction time lifespan, valid time lifespan and version lifespan. The model defines a temporal ODL with temporal restrictions *as_of* and *as_best_known*. It does not define boolean spatial, spatio-temporal and temporal operators. The authors report that the proposed model is being actually implemented.

6 Conclusions

This paper presented an extensible framework, based on extending an object-oriented database system with kernel spatio-temporal classes, data structures and functions, to provide support for the development of spatio-temporal applications. Different scientific applications can be implemented using this framework, by having the user specify application-dependent classes as subclasses of the classes provided in the kernel.

The class kernel implements a set of basic operators which can be used to perform all typical spatial and temporal queries described in the literature, where spatial data description is restricted to vector data in 2D. These operators can be further combined to implement complex spatio-temporal queries.

In order to arrive at the operator set, we used results from other researchers (in the spatial and temporal operator domains), referenced in the text. Furthermore, we extended their work by allowing distinct spatial representations for a given real world object, which can thus even have different spatial evolutions in time. A lake, for instance, can be represented by a polygon or by a point (polygon centroid) according to the scale adopted. The lake borders may suffer variation with time, without affecting the centroid – in this case, the lake is a spatio-temporal object with two different spatial representations, which present distinct temporal evolution characteristics. This, in turn, presents several interesting implementation problems which we were able to solve by implementing methods to choose the adequate representation for a given user context.

This framework is being incorporated into the UAPE environment [OPM97], which allows users to design databases for environmental applications. Another work under way concerns the development of integrated spatio-temporal index structures, to optimize queries. There are several open issues for future work. One of them, as remarked, is improving the interface. Another concerns extending spatio-temporal handling to raster data, maybe by extending the work of [WB97]. Finally, other directions concern the update of spatio-temporal data, and integration of heterogeneous databases.

Acknowledgments

Glaucia Faria is currently with Microsoft Corp. (glauciaf@exchange.microsoft.com). Claudia Bauzer Medeiros is with State University of Campinas (cmbm@dcc.unicamp.br). This work was partially supported by grants from FAPESP, CNPq, the European Community and by the SAI PRONEX program.

References

[A+94a] E. Apolloni et al. Requirements and Design Issues of Spatial Data Handling Systems. In *Advances in Database Systems: Implementations and Applications*, pages 49–68. Springer Verlag Courses and Lectures 347, 1994.

[A+94b] M. Arya et al. QBISM: Extending a DBMS to Support 3D Medical Images. In *Proc Data Engineering Conference*, pages 314–325. IEEE, 1994.

[Abi97] S. Abiteboul. Querying Semi-structured Data. In *Proc. ICDT Conference*, 1997.

[AKS+93] G. Anogianakis, A. Krotopoulou, P. Spirakis, D. Terpou, and A. Tsakalidis. Brain Data Base - BDB. In *Proc 4th International DEXA Conf.*, pages 361–364, 1993.

[All83] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 16(11):832–843, 1983.

[AS91] W. Aref and H. Samet. Extending a DBMS with Spatial Operations. In *Proc. 2nd Symposium Spatial Database Systems*, pages 299–317. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[ATSS93] K.K. Al-Taha, R. T. Snodgrass, and M. D. Soo. Bibliography on Spatiotemporal Databases. *ACM SIGMOD Record*, 22(1):59–67, March 1993.

[AW96] T. Andjelic and M. Worboys. Version Management for GIS in a Distributed Environment. In David Parker, editor, *Innovations in GIS 3*. Taylor and Francis, 1996.

[BDK92] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object-oriented Database System*. Data Management Systems. Morgan Kaufmann Publishers, 1992.

[BJS97] M. Bohlen, C. S. Jensen, and B. Skjellaug. Spatio-Temporal Database Support for Legacy Applications. Technical Report TR-20, TimeCenter, July 1997.

[Bot95] M. A. Botelho. Incorporating spatio-temporal facilities in object oriented databases. Master's thesis, Institute of Computing, UNICAMP, 1995. In Portuguese.

[BVH96] L. Becker, A. Voigtmann, and K. H. Hinrichs. Temporal Support for Geo-Data in Object-Oriented Databases. In *Proc. of DEXA'96*, volume 1134 of *LNCS*, pages 79–93. Springer, 1996.

[BX94] M. Batty and Y. Xie. Modelling inside GIS: Part I. Model Structures, Exploratory Spatial Data Analysis and Aggregation. *International Journal of Geographical Information Systems*, 8(3):291–308, 1994.

[BZ91] D. Benson and G. Zick. Symbolic and Spatial Database for Structural Biology. In *Proc. OOPSLA 91*, pages 329–339, 1991.

[CdFvO93] E. Clementini, P. di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings of the 3rd International Symposium on Large Spatial Databases*, number 692 in LNCS, pages 277–295, 1993.

[Cho] Chorochronos Home Page. URL: http://www.dbnet.ece.ntua.gr/ choros/.

[Cil96] M. A. Cilia. Active Databases as a Support for Topological Restrictions in Geographical Information Systems. Master's thesis, Institute of Computing, UNICAMP, March 1996. In Portuguese.

[CT95] C. Claramunt and M. Thériault. Managing Time in GIS - An Event-Oriented Approach. In *Proc. of the International Workshop on Temporal Databases*, Zurich, Switzerland, September 1995. Springer.

[Far98] G. Faria. A Spatio-Temporal Database for Geographical Information Systems. Master's thesis, Institute of Computing, UNICAMP, 1998. In Portuguese.

[FJP90] J. French, A. Jones, and J. Pfalz. Summary of the Final Report of the NSF Workshop on Scientific Database Management. *ACM Sigmod Record*, 19(4):32–40, 1990.

[Gou93] G. Gould. Why Not? The Search for Spatiotemporal Structure. *Environment & Planning A*, pages 48–55, 1993. Anniversary issue.

[GR93] O. Gunther and W-F Riekert. The Design of GODOT: an Object-oriented Geographic Information System. *IEEE Data Engineering Bulletin*, pages 4–9, september 1993.

[GS89] H. Gunadhi and A. Segev. Query Optimization in Temporal Databases. In *Proc. of the Fifth International Conference on Statistical and Scientific Database Management Systems*, pages 131–147, 1989.

[JCE+94] C. S. Jensen, J. Clifford, R. Elmasri, S. Gadia, P. Hayes, and S. Jajodia. Consensus Glossary of Temporal Database Concepts. *SIGMOD Record*, 23(1):52–64, 1994.

[KBS91] H. Kriegel, T. Brinkhoff, and R. Schneider. The Combination of Spatial Access Methods and Computational Geometry in Geographic Database Systems. In *Proc 2nd Symposium Spatial Database Systems*, pages 5–22. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[KV92] M. Kraak and E. Verbree. Tetrahedrons and Animated Maps in 2D and 3D Space. In *Proc 5th International Symposium on Spatial Data Handling*, pages 63–71, 1992. Volume 1.

[Lan93a] G. Langran. *Time in Geographic Information Systems*. Taylor & Francis Ltda, 1993.

[Lan93b] G. Langran. *Time in Geographical Information Systems*. Taylor and Francis, 1993.

[MJ94] C. B. Medeiros and G. Jomier. Using Versions in GIS. In *Proc. International DEXA Conference*, pages 465–474, 1994. Springer Verlag Lecture Notes in Computer Science 856.

[NTE92] R. Newell, D. Theriault, and M. Easterfieldy. Temporal GIS - modeling the evolution of spatial data in time. *Computers and Geosciences: An international journal*, 18(4):427–434, 1992.

[OPM97] J. Oliveira, F. Pires, and C. B. Medeiros. An Environment for Modelling and Design of Geographic Applications. *GeoInformatica*, 1(1):29–58, 1997.

[O'R94] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

[Peu93] D. Peuquet. What, Where and When - a Conceptual Basis for Design of Spatiotemporal GIS Databases. In *Proc. ACM/ISCA Workshop on Advances in Geographic Information Systems*, pages 117–122, 1993.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction*. Springer-Verlag, 1985.

[PS94] D. Papadias and T. Sellis. Quality Representation of Spatial Knowledge in Two-Dimensional Space. *The VLDB Journal*, 3(4), 1994.

[PTS94] D. Papadias, Y. Theodoridis, and T. Sellis. The Retrieval of Direction Relations Using R-trees. In *Proc. of the 5th International Conference on Database and Expert Systems Applications*, number 856 in LNCS, 1994.

[PW94] D. Peuquet and E. A. Wentz. An Approach for Time-based Spatial Analysis of Spatio-Temporal Data . In *Advances in GIS Research*, pages 489–504, 1994.

[R+91] J. Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[RP92] J. F. Roddick and J. D. Patrick. Temporal Semantics in Information Systems - A Survey. *Information Systems*, 17(3):249–267, 1992.

[S+94] R. T. Snodgrass et al. An Evaluation of TSQL2. Commentary, University of Arizona, Department of Computer Science, October 1994. In: The TSQL2 Language Specification.

[SB97] B. Skjellaug and A-J Berre. Multi-dimensional Time Support for Spatial Data Models. Technical Report 253, Institutt for Informatikk, Universitetet i Oslo, May 1997.

[SK96] K. C. Sevcik and N. Koudas. Filter trees for managing spatial data over a range of size granularities. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the 22nd VLDB Conference*, pages 16–27, September 1996.

[Skj96] B. Skjellaug. Time and Temporal Data Management - Operationalization in a Tempora l GIS. Technical Report STF 40 A 96063, Department of Informatics, University of Oslo, November 1996.

[SN97] A. Steiner and M. C. Norrie. Implementing Temporal Databases in Object-Oriented Systems. In *Proceedings of 5th Intl. Conf. on Database Systems for Advanced Applications*, Melbourne, Australia, April 1997.

[Sno95] R. T. Snodgrass. Language specification. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, chapter 32, pages 599–603. Kluwer Academic Publishers, 1995.

[SS94] R. Snodgrass and M. Soo. Bibliography on Spatiotemporal Databases. *International Journal of Geographical Information Systems*, 8(1):95–103, 1994.

[SV92] M. Scholl and A. Voisard. Geographic Applications – an Experience with O2. In F. Bancilhon, C. Delobel, and P. Kanellakis, editors, *Building an Object-oriented System – the Story of O2*. Morgan Kaufmann, California, 1992.

[SW95] P. Story and M. Worboys. A Design Support Environment for Spatio-Temporal Database Applications. In *Proc COSIT*, Springer Verlag Lecture Notes in Computer Science 988, pages 413–430, 1995.

[TP95] Y. Theodoridis and D. Papadias. Range Queries Involving Spatial Relations: A Performance Analysis. In *Proc. of the 2nd European Conf. on Spatial Information Theory*, number 988 in LNCS, pages 537–551, 1995.

[VBH96] A. Voigtmann, L. Becker, and K. H. Hinrichs. Temporal extensions for an Object-Oriented Geo-Data-Model. Technical Report Bericht Nr. 6/96-I, Institut für Informatik, Münster, Germany, 1996.

[WB97] N. Widmann and P. Baumann. Towards Comprehensive Database Support for Geoscientific Raster Data. In *Proceedings of the V Intl. ACM GIS Workshop*, pages 54–57, 1997.