

# Spatiotemporal Data Modeling and Management: A Survey

Xiaoyu Wang, Xiaofang Zhou and Sanglu Lu<sup>1</sup>

Department of Computer Science & Electrical Engineering  
The University of Queensland, Brisbane 4072 Australia  
{xyw, zxf, sanglu}@csee.uq.edu.au

**Abstract.** Many data objects in the real world have attributes about location and time. Such spatiotemporal objects can be found in applications such Geographic Information Systems (GIS), environmental data management and multimedia databases. Traditional relational database technology is not suitable for managing spatiotemporal data, which are multi-dimensional with complex structures and behaviors. Spatiotemporal data can only be managed by new generation of data management technologies such as object-oriented and object-relational databases. In this paper, we present a comprehensive survey covering aspects from fundamental user requirements for spatiotemporal applications, spatiotemporal object modeling, object storage structures and techniques for manipulation of spatiotemporal objects such as multidimensional indexing, data structures, query evaluation strategies and architectures for Spatiotemporal Database Management Systems (TDBMS).

## 1. Introduction

Many data objects in the real world have attributes about location and time. For example, a database about sea turtles records the location and time data for turtles that carry radio transmitters. Other examples include tracking vehicles with global positioning systems, mobile phone users within mobile phone networks, and environmental changes over time. Traditional relational database technology is not suitable for managing spatiotemporal data, which are multi-dimensional with complex structures and behaviors. Queries such as finding number and species of turtles which are within certain areas during a time period cannot be efficiently answered by pure relational technologies. Such queries involving spatial and temporal relationships need to be supported by new types of database architecture, new object storage structures, multidimensional indexing mechanisms, and query evaluation strategies. Object-oriented and object-relational databases hold great promises to storage and manipulation of complex objects such as spatiotemporal data.

In the past, research in spatial and temporal data models and database systems has mostly been done independently. Spatial database research has focused on supporting modeling and querying of the geometry associated with objects in a database. Several spatial access methods (e.g. transformation, overlapping regions, clipping or multiple layers) have been proposed in the literature for storing multi-dimensional object (e.g. points, line segments, areas, volumes, and hyper-volumes) without considering the notion of time. A survey by Gaede and Guenther [35] on spatial access methods provides excellent information sources. On the other side, temporal database have focused on extending the knowledge kept in a database about the current state of the real world to include the past, in the two senses of transaction time and valid time. Transaction time is defined as the time when a fact is stored in the database. Valid time is defined as the time when a fact becomes effective (valid) in reality. Temporal access methods have been proposed to index data varying over time without considering space at all.

A temporal DBMS (database management system) would support at least one of these types of time. Depending on the time dimension(s) supported, we distinguish among three temporal

---

<sup>1</sup> Currently a visiting research fellow from Nanjing University, China.

DBMSs: valid-time (in the past called historical), transaction-time (in the past called rollback), and bi-temporal databases. A transaction-time database records the history of a database activity rather than real-world history. As such, it can “rollback” to one of its previous states. Since previous transaction times cannot be changed (every change is stamped with a new transaction time), there is no way to change the past. A valid-time database maintains the entire temporal behavior of an enterprise as best known now. It stores our current knowledge about the enterprise’s past, current, or even future behavior. If errors are discovered in this temporal behavior, they are corrected by modifying the database. When a correction is applied, previous values are not retained. It is thus not possible to view the database as it was before the correction. A bi-temporal database combines the features of the other two types. It more accurately represents reality and allows for retroactive as well as post-active changes. Indexing methods are also classified according to the above taxonomy. A wide range of access methods has been proposed to support multi-version/temporal data by keeping track of data evolution over time. For excellent recent surveys by Betty Salzberg and Vassilis J. Tsotras on temporal access methods see [3].

These databases both are important areas of database research. So many researchers in both areas have felt that there are important connections in the problems addressed by each area and the techniques and tools utilized for their solution. Since both deal with “dimensions” or “spaces” of some kind, and that an integration field of “spatiotemporal databases” should be studied and would have important applications. We can find many papers in temporal databases conclude with phrases such as “the ideas in this paper may be extended to spatial databases.” Similarly, many papers in spatial databases suggest that techniques developed for spatial databases apply to temporal databases, by restricting attention to one dimension only. But so far relatively little systematic interaction and synergy among these two areas have occurred.

There are many important applications of the spatiotemporal database management systems (STDBMSs). Some of them can be seen in Geographic Information Systems (GIS), environmental information system and multimedia. There are also many applications that are created the data that managed by spatiotemporal database system and changes over time. For example global change (as in climate or land cover changes), transportation (traffic surveillance data, intelligent transportation systems), social (demographic, health, etc.), telecommunications (mobile phones), multimedia (animated movies) applications and so on. For example, in the University of Hong Kong they have developed a dynamic video database management system called MOOVIS (Manager of an Object-Oriented Video Information System) that recognizes the composite structure of multimedia data objects. At the heart of the system is a sophisticated object-oriented data modeling system described as a Conceptual Clustering Mechanism (CCM), which is a compromise between a strongly typed data model that allows inheritance and a weakly typed data model that provides flexible facilities for updating objects.

Put briefly, a spatiotemporal database is a database that embodies spatial, temporal, and spatiotemporal database concepts, and captures simultaneously spatial and temporal aspects of data. All the individual spatial and temporal concepts (e.g., rectangle or time interval) must be considered. However, attention focuses on the area of intersection, which is challenging, as it represents inherently spatiotemporal concepts (e.g., velocity and acceleration). In spatiotemporal data management, the simple aggregation of space and time is inadequate. Simply connecting a spatial data model to a temporal data model will result in a temporal data model that may capture spatial data or in a spatial data model that may capture time referenced sequences of spatial data. Rather, the temporal characteristics of spatial objects (i.e., how entities evolve in space) must be investigated in order to produce inherently spatiotemporal concepts such as unified spatiotemporal data structures, spatiotemporal operators (e.g., approach, shrink), and spatiotemporal user-interfaces.

Spatiotemporal databases deal with geometry changing over time. In general, geometry cannot only change in discrete steps, but continuously, for moving objects. If only the position in space of an object is relevant, then "moving point" is a basic abstraction; if also the extent is of interest, then the "moving region" abstraction captures moving as well as growing or shrinking

regions. We propose a new line of research where moving points and moving regions are viewed as three-dimensional (2D space + time) or higher-dimensional entities whose structure and behavior is captured by modeling them as abstract data types. Such types can be integrated as base (attribute) data types into relational, object-oriented, or other DBMS data models; they can be implemented as data blades, cartridges, etc. for extensible DBMSs.

In this paper, requirements for a spatiotemporal application environment are discussed in section 2. In section 3, the spatial, temporal and spatiotemporal concepts are presented and analyzed, and formal expressions are given. Section 4 introduces models and languages in spatiotemporal databases. Section 5 discusses storage structures, indexing techniques and query processing. Section 6 discusses architectures of STDBMS. We conclude this paper in Section 7.

## 2. Spatiotemporal Applications

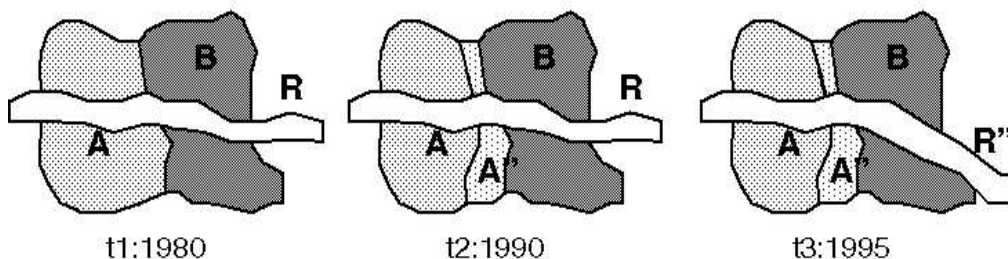
### 2.1 Three Types of Spatiotemporal Applications

Spatiotemporal applications can be categorized three different type [18] according to the type of data they manage:

- Applications may involve *objects with continuous motion*; for example, a navigational system manages moving objects. In this type of application, objects change position, but not shape. For example, a “car” moves in a road network, but its shape remains unchanged (That is normal case, we don’t consider car accidents.) [6].
- Applications dealing with *discrete changes of and among objects*. These applications involve objects located in space, whose characteristics, such as shape, as well as their position may *change* discretely in time. For example, “landparcels” or “rivers” change positions in cadastral information system, or neighboring “landparcels” change their common border, but these changes occur only discretely.
- Applications may manage objects integrating continuous *motion* as well as *changes of shape*. For example, in environmental application, a ‘storm’ is measured as a moving phenomenon, which changes properties (e.g., intensity) and shape over time.

In a cadastral system, Spatiotemporal objects may evolve as follows:

- During census year 1980, the system records that landparcel A has common borders with B and river R runs through both A and B. Landparcel A has soil type “clay,” while B has soil type “forest.”
- In 1990, landparcel A was split into A and A’. Landparcel B has soil type “high-density forest” and A’ has soil type “sparse forest.”
- In 1995, river changed its position and become R” [18].



Similarly, in a navigational system (e.g., fleet management), vehicles equipped with GPS devices transmit their positions to a computer using either radio communication links or cellular phones. At the central site, the data is processed and utilized. The queries are as follows:

- What is the exact position of taxi  $v$  in time  $t$ , given its previous position at time  $t'$ ?
- Which taxi is closest to customer A?
- What is optimal taxi distribution over the area (somewhat related to pick up per area)?
- Compute the optimal route for a ride, considering road characteristics such as the actual and theoretical speed limits, congestion, accidents, etc.

From these examples it is clear that spatiotemporal information management poses new modeling requirements. So in the next part we give a set of Spatiotemporal modeling requirements [7] and take some examples from the above three application categories.

## 2.2 Spatiotemporal Modeling Requirements

Based on theoretical research [32] as well as applied experience the following set of spatiotemporal requirements, at the modeling level, is drawn by D. Pfoser and N. Tryfona at reference [7]:

- Need for representation of objects with position in space and existence in time.
- Need to capture the *change of position* in space over time. Depending on the type of application we are interested, two different changes of position matter: (a) *continuous change*, which results into *motion*. For example, in a navigational system the continuous change of a “vehicle’s” position. (b) *discrete change*, such as the *change* of the shape of objects over time. An example taken from a cadastral system reflects this need: a “landparcel” has a position in space at some point in time. When it changes shape (e.g., a new part of land is attached to it) its position changes.
- Need for the definition of attributes of space (*spatial attributes*) and organization of them into *layers* or *fields*. Spatial attributes can be visualized as continuous (e.g., “temperature”) or as discrete (“soil type”).
- Need to capture the change of spatial attributes over time. Changes-like changes of positions, previously-can be discrete (e.g., changes on a map of “landparcels”) or continuous (change of “temperature”).
- Need to connect spatial attributes to objects. For example, a landparcel has “soil type” as an attribute. The “soil type” is an attribute of space and landparcels inherit part of it.
- Need for the representation of spatial relationships among objects in time.
- Need for the representation of relationships among spatial attributes in time. For example, the “soil type” is a result of the combination of the “acidity” and the “corrosively” of soil.
- Need to specify spatiotemporal integrity constraints, imposed either by the user, or by the designer for the integrity of the database.

## 3. Concepts of A Spatiotemporal Environment

Now we discuss concepts involved in a spatiotemporal database environment via the users’ requirements presented above. To start with, let’s review the definitions of a database:

A *database* is a collection of objects  $o_i$ , which represent part of the real world entities. We will assume that each object  $o_i$  belongs to an *object class*  $O_i$ . An object class  $O_i$  is characterized by a set of properties or *attributes*,  $A_j^i$ . Each attribute  $A_j^i$  has a *domain*  $D(A_j^i)$ . Domains are implemented by data types. So, each object in a database instance has a set of attribute values, each belonging to the domain of the corresponding attribute of the object class. A database is

called *spatial*, *temporal*, or *spatiotemporal* if it manages spatial, temporal, or spatiotemporal concepts, respectively.

### 3.1 Spatial Concepts

The Space can be seen as a set. The elements of space are called *points*, while finite sets of points (i.e., subsets of space, which can be point, lines or regions) are called *geometric figures*. Many different sets will do for space, but for practical reasons, space is modeled as a subset of  $R^2$  or  $R^3$  in current spatial applications.

The objects in real world are located in Space. In specific application environments, objects' position in space *matters* and these objects are called *spatial objects*. For example, a moving "car" in a navigational system has position, as well as a "landparcel" in a cadastral system.

Objects have attributes that characterize them. Spatial objects have, apart from descriptive attributes, also spatial attributes: for example, "vegetation" of a "landparcel". Values of spatial attributes depend on the referenced position and not on the object itself. If the spatial object "landparcel" changes position, then the value of "vegetation" will change. More specifically, *Spatial attributes are properties of space*, and spatial objects located in specific positions inherit parts of these attributes. However, not all spatial objects have spatial attributes; this depends in the application requirements. For example, no spatial attribute can be assigned to a moving "car", while many (e.g., "vegetation", "soil type") can be assigned to a "landparcel".

Spatial attributes refer to the whole space and can be represented as layer (or fields) representing one theme (i.e., thematic maps). A layer L is a representation of a spatial attribute or a layer is a set of geometric figures (which can be points, lines, regions or combinations of them) with associated values.

There are two basic types of layers: (a) those representing functions with continuous range; for example, "temperature", or "erosion", and (b) those representing functions with range of discrete values; for example, "vegetation" represented as set of regions.

### 3.2 Temporal Concepts

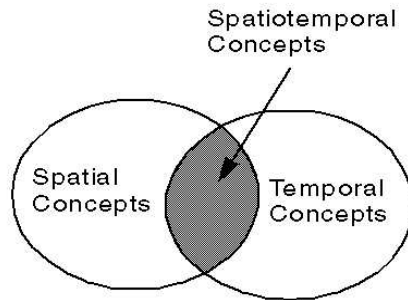
We assume a linear ordered time line, isomorphic to a finite subset of the natural numbers. The elements of this set are termed chronons.

When talking about time in databases, we refer to the temporal aspects of the real world that can be captured in the database. In literature the temporal aspects of *transaction time* and *valid time* are of foremost importance. We have been introduced them in section 1 (Introduction).

Two basic models of time are used to record facts and information of a database: *time points* and *time intervals*. A time point  $t_1$  is considered as one chronon, while a time interval  $[t_k, t_m]$ , with  $t_k, t_m$  time points and  $k \leq m$ , has duration and is defined as set of chronons. Time points and time intervals can record (or, represent) valid or transaction time. Finally, we define a time period as a set of time points, time intervals or any combination thereof. There are two basic facts for which we want to record time: *events* and *states*. An *event* occurs at an exact time point, i.e., an event has no duration. Example events are a "car crash", "sunrise" etc. A *state* is defined for each chronon in a time interval, hence it has a duration. For example, a "exam" takes place from 10am until 1pm.

### 3.3 Spatiotemporal Concepts

We combine the concepts of spatial and temporal by simply recording the spatial view (i.e., objects and layers) in time (time point and time interval); get a new result: spatiotemporal concepts.



For a moving “vehicle” in a route network we can record its “new” position in time points  $t_1, t_2, \dots, t_n$ . In the same way, as a “landparcel” changes its shape (e.g., fgsplit, expanded etc), we capture snapshots of such spatial object in time.

Recording a spatial object in a time interval is translated into capturing its *evolution* over time. Depending on the type of change, different spatiotemporal concepts are involved: (a) if the change is *continuous*, the *motion* of objects is captured. (b) if the change is *discrete*, the changes of objects are captured, through this time interval. The discrete changes of an object during a time interval can be seen as snapshots of that object, taken in time points of this time interval.

If we record an object in time of the same time period, we capture *versions* of that object; in other words, versions of an object, are related (through its identity) snapshots of it. However, we may want to record specific appearances of an object in some time points of a time period, without relating them to previous or next ones. In this case, the result is snapshots of the object.

Recording a layer in a time point results into taking *snapshots* of it. For example, the recording of the “vegetation” (of an area of interest) in May 5, 1990 gives a specific vegetation map of that area.

Recording a layer in a time interval is translated into capturing its evolution over time. Depending on the type of change, we deal with different spatiotemporal concepts: (a) if the change is continuous, it is understood as phenomenon. So, we refer to the continuous change of a layer as phenomenon. (b) if the change is discrete, we capture changes of the layer, through the time interval. The discrete changes of a layer during a time interval can be seen as snapshots of it, taken in time points of this time interval.

If we record a layer in time points of the same time period, we capture versions of that layer. Again, a layer in a specific time point of the time period is a snapshot. Finally, observing a layer in time-intervals of a time period results into observing a phenomenon, or discrete changes of it.

The following table shows all the combinations of the spatial and temporal domain, resulting in the new, spatiotemporal, concepts of *snapshot*, *change*, *motion*, *version* and *phenomenon* [7].

	Spatial object	Layer
Time point	Snapshot	Snapshot (temporal layer)
Time interval	Motion Change	Phenomenon Change
Time period	Snapshot Version Motion Change	Snapshot Version Phenomenon Change

Table 1: combining spatial and temporal concepts.

## 4. Models and Languages in STDB

Research in the models and languages for spatiotemporal database systems may be divided into two categories: (a) research that focuses on tightly integrated spatiotemporal support, and (b) previously initiated efforts that have dealt mainly with temporal aspects, extended to also cover spatial aspects. An important effort focuses on identifying the main requirements for a spatiotemporal data model and a spatiotemporal DBMS. Based on a data type approach to data modeling, the following concept of abstract models for moving objects has been studied in reference [10].

### 4.1 Abstract Models

The Abstract models allow us to make definitions in terms of infinite sets, without worrying whether finite representations of these sets exist. This allows us to view a moving point as a continuous curve in the 3D space, as an arbitrary mapping from an infinite time domain into an also infinite space domain. All the types that we get by applying the type constructor  $\tau$  are functions over an infinite domain, hence each value is an infinite set.

This abstract view is the conceptual model that we are interested in. The curve described by a plane flying over space is continuous; for any point in time there exists a value, regardless of whether we are able to give a finite description for this mapping (or relation). In an abstract model, we have no problem in using types like “moving real”, *mreal* and operations like *mdistance*, since it is quite clear that at any time some distance between the moving points exists (when both are defined).

So abstract models are conceptually simple and their semantic can be defined relatively easily. Again, this simplicity is due to the fact we admit definitions in terms of infinite sets and functions without worrying whether finite representations exist.

The only trouble with abstract models is that we cannot store and manipulate them in computers. Only finite and in fact reasonably small sets can be stored; data structures and algorithms have to work with distance (finite) representations of the infinite point sets. From this point of view, abstract models are entirely unrealistic; only discrete models are useable.

This means we somehow need discrete models for moving points and moving regions as well as for all other involved types (*mreal*, *region*,...). We can view discrete models as approximations, finite descriptions of the infinite shapes we are interested in. In spatial databases there is the same problem of giving discrete representations for in principle continuous shapes; there almost always linear approximations have been used. Hence, a region is described in terms of polygons and a curve in space (e.g a river) by a polyline. Linear approximations are attractive because they are easy to handle mathematically; most algorithms on computational geometry work on linear shapes such as rectangles, polyhedra, etc. A linear approximation for a moving point is a polyline in 3D space; a linear approximation for a moving region is a set of polyhedra. Note that a moving point can be a partial function, hence it may disappear at times, the true for the moving region.

### 4.2 Conceptual Model: the ER Model

Spatial and temporal conceptual modeling extends previous work on temporal and spatial data modeling. Spatial modeling aspects, e.g., the representation of objects’ “position” in space, as well as temporal modeling aspects, e.g., the capture of the valid time of objects’ properties, have been studied, and resulting new modeling constructs have been applied to existing conceptual models such as the ER model.

Furthermore, the structure and behavior of so-called spatiotemporal phenomena (e.g., a “storm”) have been investigated, and a formal framework with a small set of new modeling constructs for capturing these during conceptual design, has been defined as following.

The Spatio-Temporal ER (STER) [18] model includes constructs with built-in spatial, temporal, and spatiotemporal functionality. The construct that captures a temporal aspect is called *temporal*; if it has built-in support only for a spatial aspect, it is termed *spatial*; and if it has both, it is *spatiotemporal*.

While all basic constructs of the ER model can have spatial and/or temporal extent, not all types of time can be assigned to each construct. There is a semantic explanation for this: existence time indicates the existence of “something,” and this has only meaning for identifiable ontology or, in other worlds, entities (or objects). An entity set may be given attributes that describe the properties of the set’s entities. In the ontology, we stated that valid time is meaningful only for *facts*. When assigning valid time to an attribute of an entity set, we indicate that the valid times of the facts, that entities in the set are associated with specific values for this attribute, are to be captured in the database. The same applies to attributes of relationship sets in place of entity sets. Finally, valid time may be assigned to a relationship set, indicating that the time when each relation in the set is true in the miniworld is to be captured in the database. Transaction time applies to any “element” stored in the database, regardless of whether or not it may be assigned a truth value. So unlike valid time, transaction time applies to entity sets.

#### **4.2.1 STSQL Language**

Based on SQL language, the STSQL has been proposed in reference [4]. This language generalizes previous proposals by permitting relations to include multiple temporal as well as spatial attributes, and it generalizes temporal query language constructs, to apply to both the spatial and temporal attributes of relations. Because space and time are captured by separate attributes, STSQL is intended for applications that do not involve storing the movement of continuously moving objects.

Finally, modeling issues related to uncertain spatiotemporal data need to be examined. By adopting fuzzy set methodologies, a general spatial data model can be extended to incorporate the temporal dimension of geographic entities and their uncertainty. In addition, the basic data interpretation operations for handling the spatial dimension of geographic data have been extended to also support spatiotemporal reasoning and fuzzy reasoning. Some work has been initiated in this direction [6].

## **5. Storage Structures, Indexing Techniques and Query Processing**

We have given a brief overview of the data modeling efforts. This section concentrates on efforts to develop techniques for the efficient implementation of the proposed data models and languages.

### **5.1 Storage Structures and Indexing**

Substantial efforts have been devoted to the study of storage structures and indexing. In particular, (a) efficient extensions of spatial storage structures to support motion have been proposed. (b) benchmarking issues have been studied, and (c) research on purely spatial or temporal structures has been continued.

Modern DBMSs should be able to efficiently support the retrieval of data based on the spatiotemporal extents of the data. To achieve this, existing multidimensional access methods need to be extended. Work has been done in this area as well; specifically, approaches that extend R-trees and quadtrees are reported in [30] and [33], respectively, along with extensive experiments on a variety of synthetic data sets.

Work on benchmarking issues for spatiotemporal data was initiated in [31], which introduced basic specifications that a spatiotemporal index structure should satisfy, evaluated existing proposals with respect to the specifications, and illustrated issues of interest involving object

representation, query processing, and index maintenance. As a second step, a benchmarking environment that integrates access methods, data generation, query processing, and result analysis was proposed. A platform for evaluating spatiotemporal query processing strategies has been designed, implemented and used for evaluating spatial join strategies [1].

As regards spatial structures, efficient spatial access methods based on hierarchical regular decomposition of the space for images containing multiple non-overlapping or overlapping features were proposed. Bulk operations, in particular bulk loading but also bulk insertions and join operations, are inefficient if carried out by repeatedly calling operations for individual items; a generic bulk loading algorithm for spatial and also non-spatial indexes has been proposed in [14]. This method is asymptotically optimal, since it achieves the runtime of external sorting.

Concerning temporal structures, overlapping  $B^+$ -trees were used for transaction-time indexing in [34]. An asymptotically optimal B-tree supporting transaction time was proposed in [2]. Taking the R-tree as its outset, the GR-tree permits the data regions it indexes and the bounding regions inside the tree to expand with the passing of time and indexes in this way efficiently general bi-temporal data (i.e., data with both valid and transaction time support). An Informix DataBlade is available that implements the tree. Up to now, the four spatiotemporal indexing methods has appeared in the literature: 3D-trees, MR-trees and RT-trees, and HR-trees. These approaches have the following characteristics:

- 3D R-trees treat time as another dimension using a state-of-the-art spatial indexing method, namely the R-tree,
- MR-trees and HR-trees use overlapping in R-trees to represent successive states of the database, and
- RT-trees couple times intervals with spatial ranges in each node the tree structure by adopting ideas from R-trees and TSB-trees.

All these methods are extensions of the R-tree, which is based on the conservative approximation principle", i.e. spatial objects are indexed by considering their minimum bounding rectangles (MBRs). (MBRs are the most common approximations in spatial applications.) These methods are not suitable for representing regional data, in cases where a lot of empty ("dead") space is introduced in the MBRs, since this fact decreases the index ability to prune space and objects during a top-bottom traversal.

Overlapping is a technique used in access methods to combine consecutive instances into a single structure by not storing identical sub-structures. This way, space is saved without sacrificing time performance. There is an efficient indexing and retrieval method for regional data and describe the design and implementation of a new structure: Overlapping Linear Quadrees [33]. This structure is based on linear quadrees which are enhanced by using the overlapping technique in order to avoid storing identical sub-quadrants of successive instances of image data evolving over transaction time. Experimentation with synthetic region data shows that considerable storage is saved in comparison to independent linear quadrees, in the case of similar consecutive images.

The structure of Overlapping Linear Quadrees presents small differences from the structure of overlapping  $B^+$  trees. The former structure has an auxiliary substructure, a binary table that keeps the set of quadrees of the last image version. Besides, a disk page may host a number of consecutive  $B^+$  tree leaves. On the other hand in the latter structure there is a direct correspondence between pages and leaves, assuming that each leaf entry is accompanied with extra data related to it. Nevertheless, Overlapping Linear Quadrees are used in a completely different context than overlapping  $B^+$  trees: they do not store ordinary numeric data, but they store quadcodes that represent image parts and they are used for answering temporal window queries. Therefore, this structure can be used in spatiotemporal databases to support query processing of evolving images.

A very natural way to operate on a GIS is by direct interaction via a walk-through user interface. We have proposed and implemented the virtual reality geographic information system ViRGIS [25] that holds the geographic data on a central database server and interactivity loads and selectively displays the graphical information over the Internet. In this context, speed limitations, as given by the network, required the use of multi-resolution techniques and new image compression methods [20]. The system can be used for many practical applications, e.g., visualizing a flight path of an airplane or for placing antennae on a terrain [16].

The support of multi-user access in a distributed system creates severe efficiency problems already with classical data structures. An approach to solve these problems by interleaving the operations was presented by [15,9]. In the context of spatial data structures, particular theoretical problems related to space-filling curves, binary space partitions, and spatial join scheduling have been solved satisfactorily.

Two lines of research, the application-oriented discipline of geographic information systems and the technical discipline of geometric computation, in particular geometric algorithms and spatial data structures, will have significant practical impact in the near future. A state-of-the art compendium on algorithmic foundations of geographic information systems was presented by [17], and a recent survey of spatial data structures can be found in [19]. As a first result, a new basic data structure for GIS that is based on Voronoi diagrams for moving points [11] has been proposed in [5].

## 5.2 Spatiotemporal Query Processing

Work on query processing and optimization has focused on (a) the development of efficient strategies for processing spatial, temporal, and inherently spatiotemporal operations, (b) the development of efficient cost models for query optimization purposes, and (c) the study of temporal and spatial constraint databases.

In [36] it was argued that expressing spatial operations, required by different application domains, is possible through a set of window searches, so that their execution could be supported by the available spatial indexing techniques. When the availability of index structures is not guaranteed, incremental algorithms have been proposed to support join operations for time-oriented data. Regarding the execution of inherently spatiotemporal operations, the basic classes of spatiotemporal operations required by different application domains involving the representation and reasoning on a dynamic world were defined.

For query optimization purposes, analytical models that estimate the cost (in terms of node or actual page accesses) of join [29], nearest neighbor [24], and similarly [23] queries involving R-tree indexed spatial data were introduced. Also, earlier analytical models for selection queries were used as a platform to support direction (e.g., north, north-east) between two-dimensional objects [36], and the problem of *parallel and distributed similarity search* was studied in a shared-nothing multi-computer architecture, where an R-tree is declustered among the sites of the network [21,22].

In [21] it was proposed a technique which is based on a careful investigation of the currently available data in order to exploit parallelism up to a point, retaining low response times during query processing. The underlying access method is a variation of the  $R^*$ -tree, which is distributed among the components of a disk array, whereas the system is simulated using event-driven simulation. The performance results conducted demonstrate that the proposed approach outperforms by factors a previous branch-and-bound algorithm and a greedy algorithm, which maximizes parallelism as much as possible.

In [22] it was developed an efficient query processing strategy for parallel nearest neighbor finding, assuming a shared nothing multi-processor architecture, where the processors communicate via a network. In this method, the relevant sites are activated simultaneously. In order to achieve this goal, statistical information is used. The efficiency measure is the response

time of a given query. Experimental results, based on real-life and synthetic datasets, show that the proposed method outperforms the branch-and bound method by factors.

## **6. STDBMS Architectures**

In database technology, many different system architectures [28] have been proposed and are used in systems dealing with spatial and temporal data. Each has its particular strength and weaknesses. The most obvious system architectures are the ones proposed as being especially useful in spatial only and temporal only systems. Three possible architectures are presented next.

### **6.1 Standard Relational with Additional Layer**

A first possibility is to implement a spatiotemporal layer on top of a standard relational, object oriented or object relational DBMS. One can distinguish between two different approaches: The ‘thin layer’ approach focuses on exploiting, as much as possible, the DBMS data model in order to handle spatial and temporal data. Concepts such as abstract data types are used whenever possible. The ‘thick layer’ approach emphasizes the middle-ware aspects. The DBMS is primarily used as a persistent object store.

While query processing in the thin layer approach is done in the generic component using generic concepts, in the thick layer approach, tailored application specific query processing is done in the additional layer. This is similar to the support of application object services known from middle-ware layers. Both approaches make it possible to offer data models and query languages that are different from those supported by the underlying generic component.

### **6.2 Combination Architecture**

Very similar to the layered architecture is the combined architecture that also uses a standard DBMS as its basis. In addition, other storage components (such as a file system) are used to store the spatial and temporal data and indexes. While this approach allows exploiting different pre-existing components possibly specialized for spatial or temporal data handling, it also introduces the problem of coordinating different independent sub-systems. In particular, the query optimization task across different sub-systems is non-trivial. As in the layered architecture, the combination architecture can distinguish between a thick-layer and a thin-layer approach.

### **6.3 Extensible DBMS**

Instead of only allowing implementing on top of the DBMS, extensible systems permit the integration of new system components such as data types, access methods, storage structures, and low-level query processing into the DBMS kernel. Extensible DBMS have been developed and implemented in research prototypes [13,27,26], and they have become commercially available.

One of the possible spatiotemporal system architecture is to use one of these systems and extend it with spatiotemporal components. The project participants at Hagen and ETH have own prototype implementations of extensible DBMS with special properties that make them particularly interesting in a spatiotemporal context.

Secondo [12,8] represents an approach to build extensible database systems based on formalism for describing such systems called second-order signature (SOS). In SOS, one can define a “logical” DBMS data model and a query plan algebra (“physical algebra”), and optimization rules to translate from the former to the latter. As a result, it is possible to construct a generic DBMS system frame that can be filled with implementations of a wide range of data models

CONCERT focuses on physical database design aspects in the context of extensible systems. One of the main difficulties in implementing a new integration with the built-in query processing engine. Instead of allowing arbitrary extensions, CONCENT provides a set of basic physical design possibilities with their corresponding query processing strategies. Extensions are built by defining relationships to the built-in physical design concepts.

A useful aspect of CONCERT in the context of spatiotemporal data is its ability to integrate non-database repositories such as legacy application systems or WWW servers into its query processing architecture. On the other hand, the effort to be made coordinating the sub-systems complicates the system design.

## 7. Conclusions

We have presented the phase of requirement analysis for spatiotemporal applications and addressed some key spatial, temporal and spatiotemporal concepts. We also have presented survey on research results on the data models, indexing, structures, query evaluation, and architectures for spatiotemporal data management.

Modern database systems such object-oriented and object-relational databases should be able to efficiently support type of data. Towards this goal, appropriate extensions of some existing multi-dimensional access methods can be exploited in order to index and retrieve spatiotemporal objects, satisfying user's demands. So efficient storage and retrieval techniques for non-traditional data, such as moving objects in a multi-dimensional space, are necessary in modern database systems.

An STDBMS should offer appropriate data types and query language to support (static or moving) spatial data, provide efficient indexing and retrieval methods, and exploit cost models for specialized spatiotemporal operations for query processing and optimization purposes. In this paper we have conducted a comprehensive review of studies in these areas. Such a review provides a solid foundation towards solving spatiotemporal data management problems using new generation of object-oriented database technologies.

## References

- [1] A. Papadopoulos, P. Widmayer, and M. Scholl: A Performance Evaluation of Spatial Join Processing Strategies, Proc. 6<sup>th</sup> Intern. Symp. On Spatial Databases (SSD 99), Hong Kong, 1999.
- [2] B. Becker, S. Gschwind, T. Ohler, B. Seeger, P. Widnayer: An Asymptotically Optimal Multiversion B-tree, The VLDB Journal 5(4), 1996.
- [3] B. Salzberg and V. Tsotras: Comparison of Access Methods for Time- Evolving Data. ACM Computing Surveys, Vol. 31, No. 2, June 1999.
- [4] Bohlen, M.H., Jensen, C.S., Skjellaug, B.: Spatio-Temporal Database Support for Legacy, 1998 ACM Symposium on Applied Computing, Georgia, 1998.
- [5] C. Gold, P. Remmele, and T. Roos: Fully Dynamic and Kinematic Voronoi Diagrams in GIS. Algorithmica, 1998. (special issue on Cartography and GIS).
- [6] Dieter Pfoser and Christian S. Jensen: Capturing the Uncertainty of Moving-Object Representations. SSD'99 LNCS 1651, pp. 111-131, 1999 (CH-99-2).
- [7] Dieter Pfoser and Necaria Tryfona: Requirements, Definitions and Notations for Spatiotemporal Application Environments. ACM GIS'98 (CH-98-9).
- [8] Dieker, S. and R.H. Gutting: Plug and Play with Query Algebras: SECONDo. A Generic DBMS Development Environment. Fernuniversitat Hagen, Informatik-Report 249, 1999.
- [9] E. Soisalon-Soininen, and P. Widmayer: Relaxed balancing in Search Trees, Invited contribution to the Festschrift for the 60<sup>th</sup> birthday of Ron Book, Advances in algorithms, languages, and Complexity, 1997.
- [10] Erwig, M., Gutting, R.H., Schneider, M., vazirgiannis, M.: Abstract and Discrete Modeling of Spatio-Temporal Data Types. ACM GIS'98
- [11] G. Albers, L. J. Guibas, J. S. B. Mitchell, and T. Roos: Voronoi Diagrams of Moving Points. Int. Journal of Computational Geometry and Applications, 8(3), 1998.

- [12] Guting, R.H., C. Freundorfer, L. Becker, S. Dieker, and H. Schenk: Secondo/QP: Implementation of a Generic Query Processor. CHOROCHRONOS Technical Report CH-97-9
- [13] Haas, L.M., W. Chang, G. M. Lohman, J. McPherson, P. F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita: Starburst mid-flight: As the dust clears, IEEE TKDE, 2(1), 1990.
- [14] J. Van den Bercken, B. Seeger, P. Widmayer: Generic Approach to Bulk Loading Multidimensional Index Structures, Proc. 23th Int. Conf. On Very Large Databases, Athens, Greece, 1997.
- [15] Larsen K., E. Soisalon-Soininen, and P. Widmayer: Relaxed Balance Through Standard Rotations. Proc. 5<sup>th</sup> Workshop on Algorithms and Data Structures, LNCS 1272, 1997.
- [16] M. Beck, S. Eidenbenz, C. Stamm, P. Stucki, and P. Widmayer: A Prototype System for Light Propagation in Terrains. Invited Contribution to Computer Graphics Int'l. 1998.
- [17] M. van Kreveld, J. Nievergelt, T. Roos, and P. Widmayer: Algorithmic Foundations of Geographic Information. Proc. Algorithmic Foundations of GIS, Udine, LNCS 1340, 1997.
- [18] Nectaria Tryfona and Christian S. Jensen: Conceptual data modeling for Spatio-temporal Applications. CHOROCHRONOS Technical Report CH-98-8.
- [19] Nievergelt, J and P. Widmayer: Spatial data structures – concepts and design choices. Proc. Algorithmic Foundations of GIS, Udine, LNCS 1340, 1997.
- [20] Pajarola R., and P. Widmayer: Spatial Indexing into Compressed Raster Images. Proc. Int'l Workshop on Multimedia Database Management Systems, 1996.
- [21] Papadopoulos A. and Y. Manolopoulos: Similarity Query Processing using Disk Arrays, Proc. ACM SIGMOD Conf., ACM SIGMOD Record, 27(2), 1998.
- [22] Papadopoulos A. and Y. Manolopoulos: Parallel Processing of Nearest Neighbor Queries in Declustered Spatial Data, Proc. 4<sup>th</sup> ACM GIS, 1996.
- [23] Papadopoulos A. and Y. Manolopoulos: Performance of Nearest Neighbour Queries in R-trees. Proc. 6<sup>th</sup> Int. Conf. On Database Theory, 1997.
- [24] Proietti, G. and C. Faloutsos: I/O complexity for Range Queries on Region Data Stored Using an R-tree, Proc. 15<sup>th</sup> IEEE Conf. On Data Engineering(ICDE'99), 1999
- [25] R. Pajarola, T. Ohler, P. Stucki, K. Szabo, P. Widmayer: The Apls at your Finger Tips: Virtual Reality and Geographic Information Systems. Proc. 14<sup>th</sup> Int. Conf. On Data Engineering, 1998.
- [26] Schek, H.-J., H.-B. Paul, M.H. Scholl, and G. Weikum: The DASDBS project: Objectives, Experiences, and Future Prospects. IEEE TKDE, 2(1), 1990.
- [27] Stonebraker, M., L. Rowe, and M. Hirohama: The Implementation of POSTAGES, IEEE TKDE, 2(1), 1990.
- [28] The Chorochronos Participants: Chorochronos: A Research Network for Spatiotemporal Database Systems. CHOROCHRONOS Technical Report CH-99-11
- [29] Theodoridis, T., E. Stefanakis, and T. Sellis: Cost Models for Join Queries in Spatio Databases. Proc. 14<sup>th</sup> IEEE Conf. On Data Engineering, Orlando, Florida, 1998.
- [30] Theodoridis Y., M. Vazirgiannis, and T. Sellis: Spatio-Temporal Indexing for Large Multimedia Applications. Proc. 3th IEEE on Multimedia Computing and Systems, ICMCS'96, Hiroshima, Japan, 1996.
- [31] Theodoridis Y., T. Sellis, A. Papadopoulos, and Y. Manolopoulos: Specifications for Efficient Indexing in Spatiotemporal Databases. Proc. 10<sup>th</sup> Int. Conf. On Scientific and Statistical Database Management, Capri, Italy, 1998.
- [32] Tryfona, N., and Hadzilacos, Th.: 1995. Geographic Applications Development: Models and Tools at the Conceptual level. ACM GIS'95
- [33] Tzouramanis, T., M. Vassilakopoulos, and Y. Manolopoulos: Overlapping Linear Quadtrees: a Spatiotemporal Access Method. Proc. 6<sup>th</sup> ACM GIS 1998.
- [34] Tzouramanis, T., Y. Manolopoulos and N. Lorentzos: Overlapping  $B^+$  tree – an Implementation of a Transaction Time Access Method. Data and Knowledge Engineering.
- [35] V. Gaede and O. Guenther: Multidimensional Access Methods. ACM Computing Surveys, Vol. 30, No. 2, June 1998.
- [36] Y. Theodoridis, D. Papadias, E. Stefanakis and T. Sellis: Direction Relations and Two-Dimensional Range Queries: Optimisation Techniques. CHOROCHRONOS Technical Report CH-98-2.