Literature Survey - TDT 4150
GIS Databases: An Overview

Atle F. Sveen
atlefren@stud.ntnu.no

April 18, 2008

Abstract

This literature survey covers the area of GIS databases, an overview of work done in the field is presented as a summary. Definitions of GIS, Geographical Information System, is presented. How the geographical data used in such systems are stored and managed in spatial databases, and the more specific term GIS database, are inspected in more detail, along with the definition of what a GIS database is and how it differs from spatial databases.

A historical overview and motivation for the research is given. The requirements for such a database is presented, focusing on such areas as spatial data types and the combination of spatial and alphanumerical data. The problems of modelling geographical data, with focus on *tesseral* and *vector* models, as well as the concepts *point*, *line* and *region* and two methods of modelling and implementing these (*simplices* and *realm*). An algebra for spatial data types (the ROSE algebra) is presented as an example of how to deal with these concepts and 3D data types are presented in a general manner.

Then the problems of defining and using a Spatial Query Language are presented, with focus on a presented solution to this. As a case study the query language *Spatial SQL* is presented. The emphasis lies on the two parts, the *retrieval* and *presentation*, the language it consists of. The reason for this division is also presented.

The last part covers data structures for spatial data, with main focus on quad trees and related structures. Examples of structures for *regions*, *points* and *lines* are given. Then the survey is concluded by giving some pointers to implementations of GIS databases and some thoughts about the work with this survey.

Contents

1 Introduction

The first problem in giving an overview of GIS databases is defining the term. The database community primarily associates GIS with *spatial databases* (Medeiros and Pires, 1994). The definition of a spatial database is by no means standardized, Güting (1994) refers to a spatial database as a database capable to:

> "...deal with large collections of relatively simple geometric objects"

and makes the distinction between an *image database* and a spatial database, stating that the latter:

> "...is associated with a view of a database as containing sets of objects in space rather than images or pictures of a space."

As stated by both Güting (1994) and Shekhar et al. (1999), a spatial database aim at the effective and efficient management of data related to such things as:

- a space such as the physical world (geographic space, geography)
- parts of living organisms (a model of the human brain)
- man made space or engineering designs (CAD models)

A definition of the term GIS (Geographic Information System) on the other hand are given by Medeiros and Pires (1994), which says that the more general definition of a GIS would be:

> "a digital information system whose records are somehow geographically referenced"

We observe the close relation between this definition and the first point mentioned in the list of what data a spatial database system aims to manage. On the basis of this it can be said that a GIS database is a special case of a spatial database. That is, a spatial database specialized to deal with data related to the physical world and to interact with a GIS system. This notion is supported by both Güting (1994) and Shekhar et al. (1999), as they state that the main area driving the research on and improving the functionality of spatial databases is the GIS technology.

Therefore it is not surprising that the term spatial database is used to describe a database for GIS purposes and that much of the literature on spatial databases consider GIS related questions in particular. It is however useful to remember that:

> "We consider spatial DBMSs to provide the underlying database technology for GISs and other applications."

and that:

"We do not claim that a spatial DBMS is directly usable as an application oriented GIS."

as Güting (1994) states.

With these words in mind we move on with this survey. Fist a historical overview is given (2), then a more thorough description of the requirements for a spatial DBMS for GIS (3). Modelling of spatial data (4), query languages (5) and data structures (6) for spatial databases are presented and discussed, before the survey is concluded with some pointers to existing databases for GIS (7).

2 Historical Overview

At least since the advent of the relational database systems (1970's) there have been attempts to manage spatial data in DBMSs (Güting, 1994). The INGRES team had been interested in GISs in the early days of the project and suggested mechanisms to support such systems in an article published in 1975 (Stonebraker and Hellerstein, 2005).

However, there where still problems related to the managing and querying of spatial data. According to Stonebraker and Hellerstein (2005), a GIS issue haunted the INGRES team and in turn the proposed solution to this problem motivated the Object Relational (OR) era of database systems. The problem was this:

To store the location of a collection of intersections in a relational DB one could use this schema:

```
Intersections (I-id, long, lat, other-data)
```

This method requires that geographic points are stored in the database. To find all intersections within a bounding rectangle with borders (X1, X2, Y1, Y2) the SQL query would be:

```
Select I-id
From Intersections
Where X0 < long < X1 and Y0 < lat < Y1
```

This is a two dimensional search, and the B-trees used in INGRES was a one dimensional access method. To do a two dimensional search with a one dimensional method are painfully slow, so there was no way this query could run fast in the INGRES system.

The solution was to let the user define their own *data types*, *operators*, *functions* and *access methods*. This meant that the problem could be solved by letting the user define the data types *point* and *box*, the operator *point in rectangle*, a function to support the operator (*Point-in-rect(point,box)*) and an access method such as a *R-tree* (see section 6). This is the OR methodology in practice.

3 Requirements

Güting (1994) presents his personal view on the definition of a spatial DBMS. Since there are no generally accepted definition of this subject his definition will be presented here, as it gives a good background for understanding how a spatial database differ from "normal" database systems:

1. A spatial database system is a database system.
2. It offers spatial data types (SDTs) in its data model and query language.
3. It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

This list covers most of the requirements for a spatial database system. The first may seem a bit trivial, but the point is that spatial data always will be connected to "non spatial" (alphanumeric) data, so the system have to support such data as well. The usual approach here is using a full fledged DMBS with the added capability of handling spatial data.

The second point is critical. Without support for data types such as *point*, *line* and *region* the modelling, storage and operation of geographically referenced objects are difficult. More details on SDTs are given in section 4.

The last requirement presented by Güting (1994) is that the SDTs should be supported in the implementation, and that methods to provide *spatial indexing* should be implemented.

4 Modelling of spatial data

4.1 The representation of the world

There are two fundamentally different ways of perceiving the world reflected in the modelling of geographical data, the *field* and the *object* model. These are mapped into two different structures: *tesseral* (raster) and *vector* (Medeiros and Pires, 1994). These views can be said to differ in the way that the field model is a description of space itself (e.g. every point in space), while the object model describes distinct entities arranged in space, each with its own geometric description (Güting, 1994).

To try and capture the concept of both these views in a spatial database Güting (1994) uses the concepts of *single objects* and *spatially related collections of objects*. When it comes to single objects there are in principle three types used, *point*, *line* and *region* (or *surface*) (as shown in figure 1). According to Viqueira and Lorentzos (2007) the definition of these types should match the common interpretation by humans. Güting (1994) gives the following definitions: A point is only given by location in space, not by extent, a line is understood as a curve in space (usually a polyline) and a region is an abstraction of something that have a 2D extent in space, it can have holes and consist of disjoint pieces.

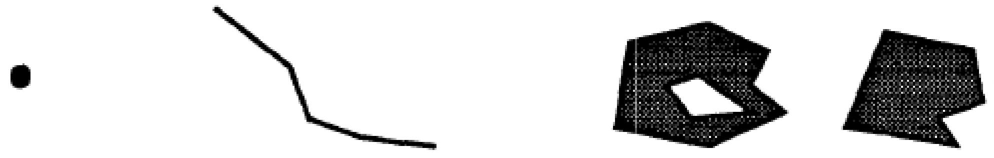

Figure 1: The three basic abstractions point, line and region

The most important models for *collections of objects* are *partitions* and *networks*. Partitions can be viewed as a set of disjoint region objects, and networks are commonly viewed as a graph embedded in the plane, with point objects forming the nodes and line objects forming the edges.

4.2 The basic concepts

From this we gather that the tree concepts point, line and region (or more generic: *polygon*) are fundamental in the modelling of spatial data. Schneider (1997) explains that from the users perspective the most popular and fundamental abstraction of spatial objects are these *structure oriented* data types. He also states that the idea of using these primitives for representing spatial data is an old one, and that there has been proposed many models that incorporate these concepts with alphanumerical data types.

One of the main problems in representing spatial data is that the geometric concepts traditionally are implemented using coordinates and Euclidean geometry. This representation of Euclidean geometry in a finite computer system can lead to problems, such as: (Egenhofer et al., 1990)

- Scaling of coordinates changes topology.
- The intersection of two lines does not necessarily lie on both lines.

To solve this problem the introduction of a discrete geometric basis for modelling and implementation has been suggested. Egenhofer et al. (1990), among others, have come up with an approach based on combinatorial topology. The main concept is that of a *simplex*. The simplex is the minimal object for each spatial dimension, examples are the 0-simplex (a node), the 1-simplex (an edge) and the 2-simplex (a triangle). Any n-simplex is composed of n+1 simplices of dimension n-1. This means that the 2-simplex triangle consists of 3 1-simplices (edges).

The benefit of the combinatorial topology is the fact that only a small set of geometric operations are necessary to manipulate the data collection and that the problems of implementing the Euclidean geometry are overcome (Egenhofer et al., 1990).

Another solution to overcome the problems posed by the Euclidean geometry is the *realm* concept. A realm is essentially a finite set of points and non intersecting line segments (see figure 2). Every point, line or region one would ever want to create can be described combining the points and line segments present in the realm. An important point is that the objects of the realm are defined by finite coordinates, and not in the abstract Euclidean space. Therefore, all the geometric primitives and the realm updates "are defined in error-free integer arithmetic" (Güting and Schneider, 1995).

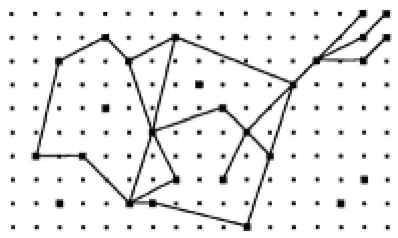

Figure 2: An example of a realm

4.3 Spatial Data Types

Güting (1994) presents the ROSE algebra (Güting and Schneider, 1995) as an example of a *spatial algebra*, or a *system of spatial data types*. A spatial algebra is a collection of spatial data types (e.g. the abstractions of point, line and region), their relations and the operations that can be performed on them.

The ROSE algebra is described in detail by Güting and Schneider (1995) and will not be repeated here. The main point is that a spatial algebra defines data types and describes the kinds of operations that can be done on them. It is also an important point that it is possible to implement the algebra into a DBMS data model and query language.

4.4 3D Spatial Data Types

Up until recently much of the work on spatial data types for GIS purposes have focused on the two dimensional case. 3D data types have been used in areas such as CAD and robotics, but the field of GIS have been limited to 2.5D (i.e. a 2D representation along with a z-coordinate) (Schneider and Weinrich, 2004).

An attempt to formally specify a comprehensive and general set of 3D spatial data types is made by Schneider and Weinrich (2004), which starts by defining an algebra for 3D data (*SPAL3D*). The 3D data types *point3D*, *line3D*, *surface*, *relief* and *volume* and the operators *union*, *difference* and *intersection* are mathematically defined and explained. Although the specification of SPAL3D is incomplete, it is a first step and serves as a case proving that the field of spatial 3D data types are researched.

5 Query Languages

One of the things that separates a GIS (or spatial) database from an "ordinary" database is the fact that it operates on entities that may have both spatial as well as alphanumerical attributes. Hence, queries to the database involves identification of these entities based on both the spatial and alphanumerical attributes (Ooi et al., 1989). In general there are two ways of achieving this, either to specifically design a new query language, or to extend an existing one with spatial operations.

5.1 Extension of existing query languages

Many attempts have been made using the latter approach, such as the SQL extension GEOQL (GEOgraphic Query Language) described by Ooi et al. (1989). Using GEOQL, the query "find all roads that intersect wit roads that are adjacent to Monash University or RMIT" would be:

```
SELECT   X.name
FROM     road X, road Y, region
WHERE    X intersects Y and
 Y is adjacent to Y and
 (region.name = 'Montash University' or
 region.name = 'RMIT')
```

As we see the SQL-form SELECT FROM WHERE is kept, and some geographical terms have been introduced (*intersects*, *is adjacent to*). Still, GEOQL is just one of many attempts to extend an existing (relational) query language. Egenhofer (1994) mentions GEO-QUEL (an extension of QUEL) and Query-By-Pictorial-Example (an extension of Query-By-Example). According to Egenhofer (1994), these extensions are not enough, as:

> ”. . . a spatial query language must be more than the following simple equation: *Relational Query Language + Spatial Relationships = Spatial Query Language*“

5.2 Including presentation and interaction in the query language

One way to extend this and add the ”more“ is to think of the query language as more than a tool for *retrieving* data, the task of *presenting* the data is also crucial, and should be part of a spatial query language. The standard SQL query language deals with alphanumeric input and presents the result as tables and/or alphanumeric variables. As stated earlier, one of the main characteristics of spatial data is that it is located in space, and the most obvious difference between a spatial and a conventional system is that spatial systems should have the ability to present the result on a map (e.g. graphically) (Egenhofer and Frank, 1988). It can be argued that several types of non-spatial data also could be presented graphically (such as in charts and graphs), but the main point is that spatial data is intended to be inspected graphically, a set of coordinates makes little sense to most people, whereas a map tells a great deal to many. In addition to presenting results on a map, the user should be able to pose queries to the database via a map. Egenhofer and Frank (1988) identifies two distinct types of referencing a geographical representation:

1. Selection of a sub-area (e.g. How many houses lies in this area?)
2. The identification of individual objects via the *pick* option (e.g. Who is the owner of this house?)

5.3 Spatial SQL

Egenhofer (1994) builds upon the foundations for a spatial query language presented by Egenhofer and Frank (1988) to create the spatial query language *Spatial SQL*. In his view several aspects must be considered when creating a spatial query language, such as:

- The language must provide syntactical means to deal with both purely alphanumerical data, a combination of spatial and alphanumerical data and purely spatial data. To do this, all the alphanumerical functionalities of SQL must be kept.
- To add more clauses than those already found in the SELECT-FROM-WHERE regime is undesirable (as it is complex enough as it is).
- Spatial data should be treated at a level independent of internal coding

- To present data graphically requires ”the display of context“, i.e. additional information not explicitly requested.
- Possibility to change the ”visual variables“ (colour, patterns, symbols etc.)
- A means to apply labels to objects.
- The possibility to restrict the area of interest as users often work on a subset of the data.
- Results are often obtained making dynamic changes, hence the ability to pose several queries in a row, the second based on the result of the first, are important.
- The presentation parameters are often the same, while the query changes.

To solve these problems, Egenhofer (1994) proposes to split the instructions into two separate languages, the *retrieval* and the *presentation* language. The retrieval language uses SQL as its host language, similar to TOSQL, an attempt to create a temporal query language. As the presentation language a language called GPL[1] (Graphical Presentation Language) is used. The use of this combination is described in figure 3: The display environment is set using GPL (a), then queries are posed using Spatial SQL and the results are presented using the environment set using GPL (b). Then either new queries can be posed using the same display environment or the query result can be kept, but the display environment can be changed using GPL again (c).

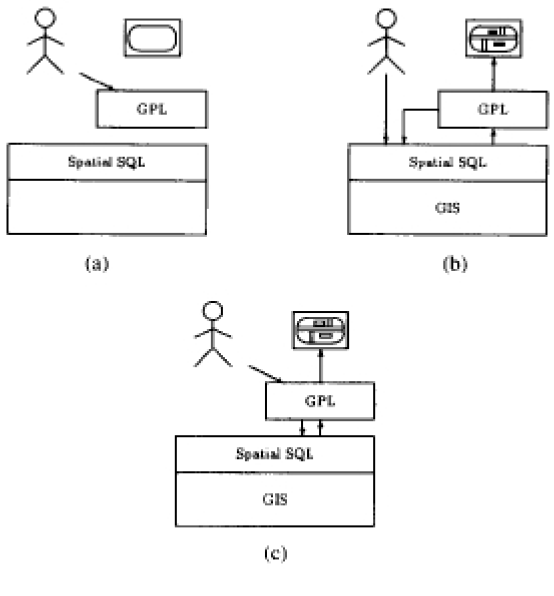

Figure 3: Interaction with Spatial SQL using GPL

[1]Not to be confused with the free software license GNU GPL.

To deal with the problems of spatial data (subsection 4.3), and to give users a high-level abstraction, the domains in the relational calculus are extended by four spatial domains (spatial_0, spatial_1, spatial_2 and spatial_3, generalized into the dimensionless domain *spatial*). Along with these domains comes operators which are divided into *unary* and *binary*, that is, operators that return spatial objects and alphanumeric values respectively.

The data definition in SQL is extended to handle spatial attributes, so to create the relation *city* with an attribute *name* and a spatial attribute *geometry*, the query would be:

```
 CREATE TABLE city
(name char(20)
 geometry spatial_2);
```

It is worth remembering that in general, a relation will have only one spatial attribute (such as the 2D attribute *(area)* in this case), but several can be used (if, for example the city is represented as a point at a larger scale).

6 Data Structures

One way to store geographical data in a database is by using a *hierarchical* data structure. This section covers some such structures used for geographical data. As shown in section 2 the structures used for 2D (and 3D) data differs from those used in storing 1D data such as the B-tree. This section is based on an article by Hanan Samet (Samet, 1984), although the article is over 20 years old the description of the structures still holds. There are quite a lot of data structures, here two variations of the *quad tree* for regions and points and the *strip tree* for curvilinear data will be presented to give an impression of how data structures for these types of data can be.

6.1 Quad trees

One class of hierarchical data structures for geographic data is *quad trees.* This class of structures is based on the principle of recursive decomposition of space. They can be used to store different types of data (point, regions, curves) and the method of decomposition may vary (based on *equal partitioning*, partitioning dependant of the input). Here two types of quad trees are presented, the *region* and *point* quad tree. They both share the same structure of decomposition, the root node represents the entire area, and has four children (hence the name *quad*tree), labelled NW, NE, SW and SE that represents a sub-quadrant (where NW is the North-Western quadrant etc.).

Each of the sub-nodes can then be divided in a similar manner. This approach makes it possible to have variable resolution, as seen in figure 4, where area B contains only homogeneous data and does not need to be decomposed further, while areas D and E are decomposed using two more levels. This approach is useful as large homogeneous areas needs little storage space, while we still get an appropriate level of detail where needed.

Region quad tree

The region quad tree is the most studied type of quad trees. The approach is to divide an image-array by successive subdivisions of the array into equal-sized quadrants. The principle is shown in figure 4, where an area (a) is decomposed as much as needed in order to classify the area. The corresponding tree structure is shown in (b). This approach to modelling an area corresponds to the *field view* presented in subsection 4.1.

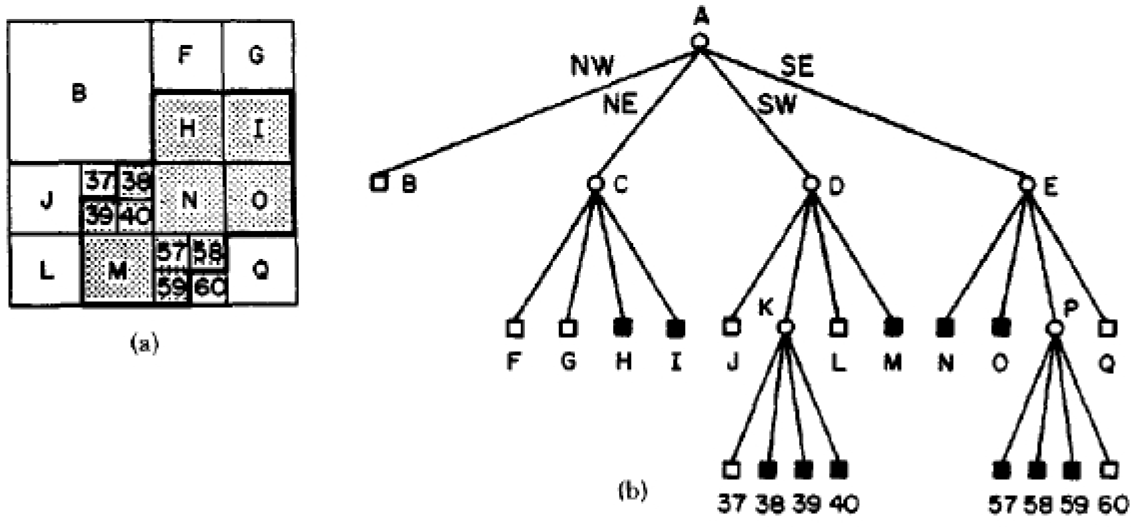

Figure 4: A region quad tree

Point quad tree

Another approach based on the quad tree is the point quad tree. This structure stores the location of points, and is a multidimensional generalization of a binary search tree. In the two dimensional case, each point to be represented is a node in the tree with four children, in a NW, NE, SW and SE order. As seen in figure 5 the area is split on the location of the first point, as the next point is inserted in the region it belongs. Then that region is split further (as with point B in the figure). Thus, the structure of the tree depends on the order the points are inserted. Because of the way points are handled only one point can occupy a specific location.

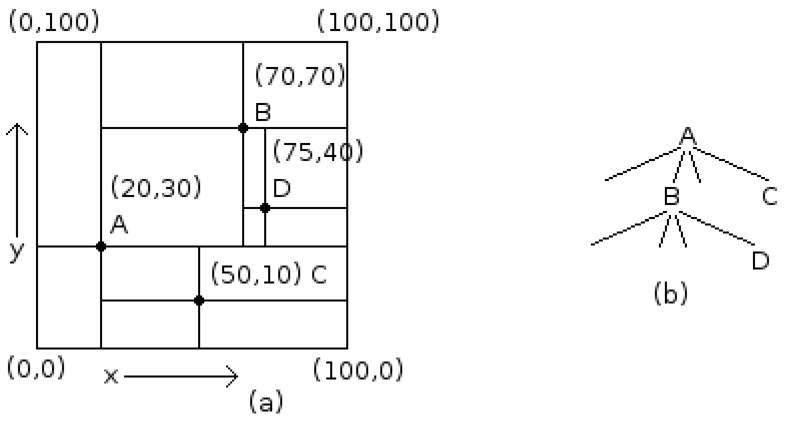

Figure 5: A point quad tree

6.2 Curvilinear data

The region quad tree describes an area by describing the interior of it. Another way to describe an area is to describe its boundaries, using a data structure for curvilinear data. The simplest form is a polygon consisting of vectors stored as a list of x and y coordinates. Several other methods to describe a curve exists, here the structure *strip tree* is presented.

By recursively approximating segments of a single curve by enclosing rectangles a strip tree representation of the curve is obtained. The decomposition is determined by selecting a splitting point so that the width of the new rectangles are as small as possible. The decomposition of the curve continues until the width is equal to or smaller than a pre-defined value. An example of this process and the resulting strip tree is shown in figure 6.

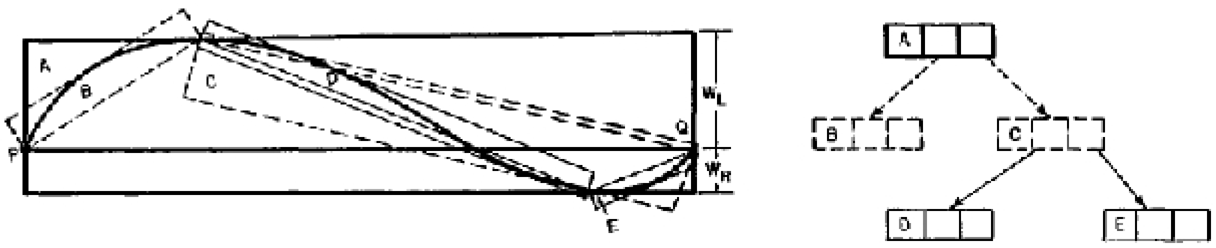

Figure 6: The decomposition of a curve and the resulting strip tree

7 Conclusions

As we have seen a GIS database differs from an "ordinary" database dealing with alphanumerical data in many ways. This survey has been an attempt to summarize the main challenges of GIS databases, and to point out some specific techniques and issues. It is in no way a complete account of the field of databases used for GIS, and some areas such as spatio-temporal databases (e.g. databases combining spatial and temporal data) have been left out. This is an area that has gotten more attention, as GIS applications starts to find the temporal dimension interesting (Posing queries such as: *Who was the owner of this house ten years ago?*).

Neither have this article focused on complete, running GIS databases, although some examples of query languages such as Spatial SQL has been presented. This is in part due to the fact that no such commercially available systems exists, although many more or less capable solutions based on for example Postgres have been developed (Shekhar et al., 1999). Special GIS databases such as the Paradise Project (The Paradise Team, 1995) have been researched and early versions have been delivered.

The goal of this article was to present an overview of the usage of databases for GIS, with emphasis on the techniques and problems posed by such a task. The intention was to present the material from the view of the database community, although I have strong bonds to the GIS community.

References

Egenhofer, M. J. (1994). Spatial sql: a query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95.

Egenhofer, M. J. and Frank, A. U. (1988). Towards a spatial query language: User interface considerations. In *VLDB '88: Proceedings of the 14th International Conference on Very Large Data Bases*, pages 124–133, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Egenhofer, M. J., Frank, A. U., and Jackson, J. P. (1990). A topological data model for spatial databases. In *SSD '90: Proceedings of the first symposium on Design and implementation of large spatial databases*, pages 271–286, New York, NY, USA. Springer-Verlag New York, Inc.

Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399.

Güting, R. H. and Schneider, M. (1995). Realm-based spatial data types: the rose algebra. *The VLDB Journal*, 4(2):243–286.

Medeiros, C. B. and Pires, F. (1994). Databases for gis. *SIGMOD Rec.*, 23(1):107–115.

Ooi, B. C., Sacks-Davis, R., and McDonell, K. J. (1989). Extending a dbms for geographic applications. *Data Engineering, 1989. Proceedings. Fifth International Conference on*, pages 590–597.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260.

Schneider, M. (1997). *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems.* Springer.

Schneider, M. and Weinrich, B. E. (2004). An abstract model of three-dimensional spatial data types. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 67–72, New York, NY, USA. ACM.

Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., and tien Lu, C. (1999). Spatial databases-accomplishments and research needs. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):45–55.

Stonebraker, M. and Hellerstein, J. M. (2005). What goes around comes around. *Readings in database systems.*

The Paradise Team (1995). Paradise: a database system for gis applications. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, page 485, New York, NY, USA. ACM.

Viqueira, J. R. R. and Lorentzos, N. A. (2007). Sql extension for spatio-temporal data. *The VLDB Journal*, 16(2):179–200.

A Appendix - "In-depth" articles

According to the specification of the task given in the course TDT 4150 the author should have curricular knowledge of at least three of the articles the survey is based on. For this literature survey these articles are:

- Ralf Hartmut Güting. An introduction to spatial database systems. (Güting, 1994)
- Max J. Egenhofer. Spatial sql: a query and presentation language. (Egenhofer, 1994)
- Hanan Samet. The quadtree and related hierarchical data structures. (Samet, 1984)