Introduction and Recent Advances in Spatial Databases

As Part of the Course

CENG-713 Advanced Topics in File Systems

by

Fatih Abut
Department of Computer Engineering
Cukurova University
⟨*fabut@student.cu.edu.tr*⟩

Advised by

Assoc. Prof. Dr. S. Ayşe Özel

December 21, 2014

Abstract

Location-based services grow at a remarkable pace and are getting increasingly popular due to the penetration of GPS in mobile phones. Such services provide the ability to find the geographical location of a mobile device and then provide services based on that location. However, at the heart of such location-based services lie spatial databases which enable the effective and efficient retrieval and management of geospatial data. In particular, spatial databases provide data types and operations specifically for working with spatial data and support index structures to efficiently store, manage and query data that represents objects defined in a geometric space. In this work, an introduction to the most important aspects of spatial database systems is given. Moreover, the R-Tree - a widespread spatial index structure - is presented which intends to expedite the execution of search queries on spatial data. Finally, recent advances in the research area of spatial databases are summarized.

Contents

1 Introduction

Spatial data represents the essential geometric characteristics of objects as well as their relationships to the space in which they exist. For example, a map is a two-dimensional object containing points, lines and polygons which in turn are used, for example, to represent state boundaries, cities and roads. Thus, a road map can be considered as a visualization of geographic information. The location of cities, roads, etc. in the world is projected on a two-dimensional area while maintaining their relative positions and distances. The data determining the length and width or height and depth are the spatial data.

Conceptually, spatial database systems are similar to relational databases in that they share the same set of components including a database management system (DBMS), data types and functions as well as indexing techniques. The main difference, however, is that spatial databases are not only able to manage one-dimensional traditional data but also spatial objects in multi-dimensional space. In this sense, spatial database systems can be seen as relational databases with a concept of spatial location and spatial extension. Spatial databases support spatial data types, spatial functions and corresponding query languages in their implementation, providing at least spatial indexing and efficient algorithms for spatial join.

Application areas of spatial databases include but are not limited to Geography (e.g. 2D maps), Engineering (e.g. 2D/3D architectural drawings), Geophysics (e.g. 3D geological strata) and Geographic Information Systems (GIS). Popular examples for existing spatial database systems are Oracle Spatial, IBM DB2 Spatial Extender, PostgreSQL along with the spatial extension PostGIS and Microsoft SQL Server.

The rest of the work is organized as follows. In section 2, a brief introduction to the basics of spatial databases is given. In section 3, the R-Tree - a widespread spatial index structure - is briefly described. Section 4 summarizes the recent advances in spatial databases and finally, section 5 concludes the work.

2 Spatial Databases: Data Types, Functions and Queries

A spatial database management system (SDBMS) is designed to facilitate the processing and handling of spatial data both for the user and applications such as Geographic Information Systems. It is a collection of functions and procedures that makes it possible to store, recover, update spatial data and execute queries on it. While typical databases are designed to manage various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently.

A SDBMS has a three-layer structure and works with a spatial application at the front end and a DBMS at the back end. The three-layer structure, illustrated in figure 1 consists of the following components:

- Interface to spatial application
- Core spatial functionality
- Interface to DBMS

The spatial attribute of an object, referred to as its geometry is an ordered sequence of vertices, which are joined together by straight lines or arcs. The semantics of the geometry is determined by the object type. Usually, the following 3 primitive geometric types are supported by each spatial database system:

- Point: Representation of the position of a single location. Points are used to represent objects when the exact details such as shape and size are not important.
- Linestring: One or more composite pairs of points that define a line segment.
- Polygon: Composition of connected line strings that form a closed ring and enclose the inner surface.

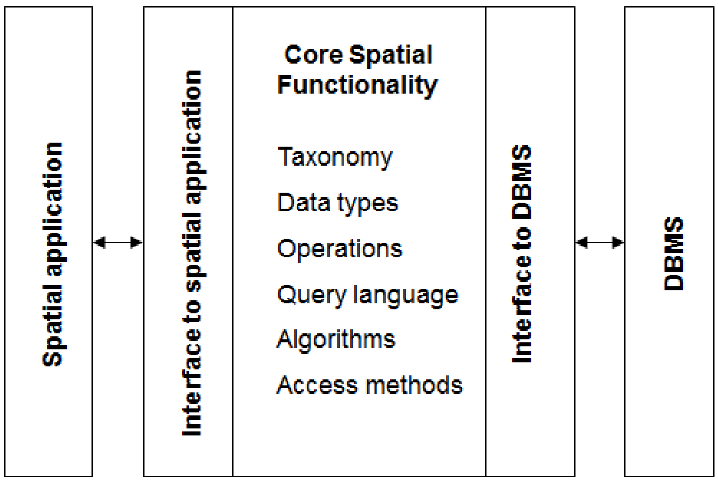

Figure 1: SDBMS Three-layer structure

To provide some sample data for illustration, the following PostGIS-based example will create a table (map) and populate it with three different geometry object types with their coordinates on a two-dimensional space - a point, a line, a polygon [1]. In this example, the object types point, line and polygon are used to represent a tree, a street and a building, respectively.

```
CREATE TABLE map (name varchar, geom geometry);

INSERT INTO map VALUES
  ('tree', 'POINT(5 12)'),
  ('street1', 'LINESTRING(8 0, 8 15)'),
  ('street2', 'LINESTRING(0 20, 20 20)'),
  ('building1', 'POLYGON(1 1, 4 1, 4 6, 1 6, 1 1)'),
  ('building2', 'POLYGON(1 10, 4 10, 4 15, 1 15, 1 10)');
```

The following spatial functions to describe the relationships between geometrical objects are also usually provided by spatial databases:

- Equals: Tests the spatial equality of two geometries.
- Intersects: Test whether the interiors of the geometries intersect.
- Disjoint: If two geometries are disjoint, they do not intersect, and vice-versa.
- Touch: Only boundaries of the objects interact with each other, but do not intersect in their interiors.
- Contains: Tests whether one geometry is fully within the other.
- Distance: Calculates the (shortest) distance between two geometries.

Once the modeling of spatial objects and their relationships is completed, spatial queries can be carried out on the model. Spatial query languages provide tools and structures specifically for working with spatial data. They enable to identify and analyze spatial relationships and achieve a localization of geometric objects. Usually, there exist three types of queries:

- Basic operations that can be applied to all data types (e.g. IsEmpty, Envelope, Boundary)
- Topological/set operators (e.g. Disjoint, Touch, Contains)

- Spatial analysis (e.g. Distance, Areas, Intersection)

Using a spatial query a number of criteria is formulated which are to be fulfilled by the spatial object(s) looked for. The following query, for example, returns the area of the building ’building1’ from the table ’map’.

```
SELECT name, ST_Area(geom)
FROM map
WHERE name = ’building1’;
```

Similarly, an another query shows how the distance between the two buildings ’building1’ and ’building2’ is calculated.

```
SELECT ST_Distance(m1.geom, m2.geom)
FROM map m1, map m2
WHERE m1.name=’building1’ and m2.name=’building2’;
```

This query returns the length of the linestring object ’street1’.

```
SELECT ST_Length(geom)
FROM map
WHERE name = ’street1’;
```

Finally, the following query checks whether the two streets ’street1’ and ’street2’ do intersect.

```
SELECT ST_Intersects(m1.geom, m2.geom)
FROM map m1, map m2
WHERE m1.name = ’street1’ and m2.name=’street2’;
```

Many other examples about spatial queries and functions can be found in [1].

3 Spatial Indices and R-Tree

Spatial databases use spatial indices to expedite the execution of search queries on spatial data. A spatial index uses the types of the spatial relationship to manage data entries, whereas each key value is considered as a point (or region in case of range data) in a k-dimensional space. k is the number of fields in the search key of the index.

There are many types of structures used for spatial indexing of both point and range data. An example of an index structure for point data is the grid file whereas structures such as R-Trees and Region Quad Trees are suitable for range data. In general, there is no index structure that always performs best. However, compared to the rest of index structures, the R-tree is most widely used and therefore in the following it is described in more detail.

The R-tree proposed by Antonin Guttman in 1984 [2] is a multi-dimensional spatial index structure which has found a significant use in both theoretical and applied contexts. The ’R’ in R-tree is for rectangle. Similar to a B-tree it is a balanced index structure. An R-tree can efficiently answer range queries (i.e. interval requests). Similar to a B+ tree, leaf nodes of an R-tree contain the spatial data to be indexed. But in contrast to this, it has no separating keys but stores in the index node rectangular data regions, which include all data regions lying under the subtree. Data points, rectangular regions or polygons are saved in the leaf nodes. Polygons, however, are often approximated by a so-called minimum bounding rectangle.

There are three basic operations that are carried out on a R-tree: Insertion, splitting and deletion. To insert an object, the tree is traversed recursively from the root node. At each step, all rectangles in the current directory node are examined, and a candidate is chosen using a heuristic such as choosing the rectangle which requires least enlargement. The search is continued until reaching a leaf node. If the leaf node is full, it must be splitted before the

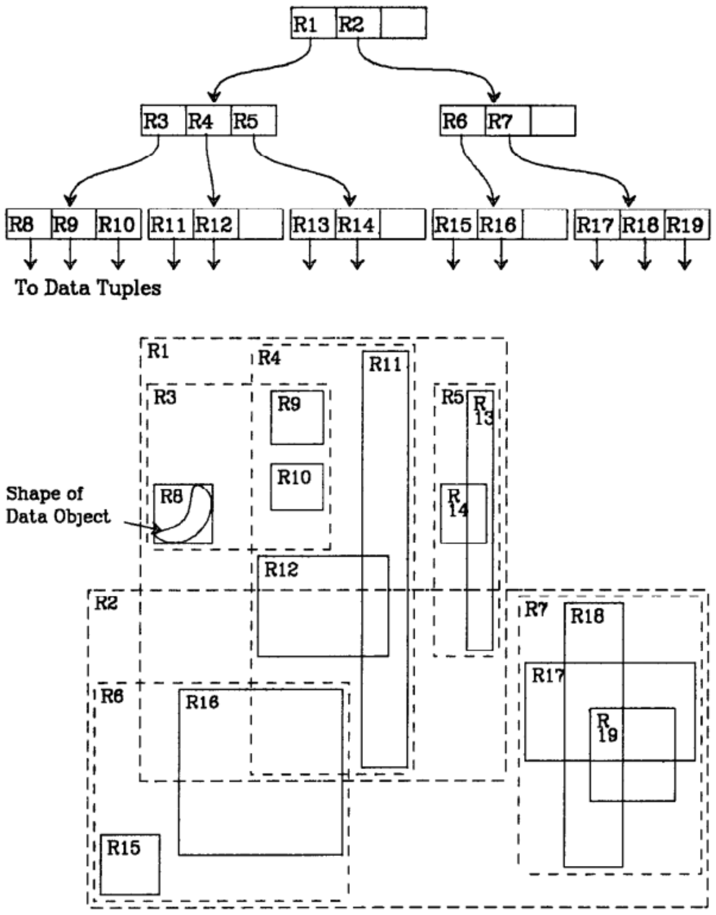

Figure 2: Simple example of an R-tree for 2D rectangles [2]

insertion is made. A heuristic needs to be employed to find the best split. In the classic R-tree, Guttman proposed two such heuristics, called QuadraticSplit and LinearSplit [2]. Deleting an entry from a page may require updating the bounding rectangles of parent pages. However, when a page is underfull, it will not be balanced with its neighbors. Instead, the page will be dissolved and all its children will be reinserted. Figure 2 shows an illustration of a an R-Tree [2].

Use of an R-tree can, for example, answer queries like

- "is point P in Figure A?",
- "is Figure B included in Figure C?" or
- "which Figures are included in Figure D?".

However, the approximation (minimum bounding rectangles) of the objects stored in the leaf nodes can lead to false positives that must be resolved by examining the request to concrete objects.

4 Recent Advances in Spatial Databases

This section gives a brief overview about the recent advances carried out in the research field of spatial database systems.

In [4], a database system, GeoSabrina, has been proposed that intends to integrate the management of both non-spatial and spatial data in a unique conceptual model. In more detail, the demand for the co-management of geographical and traditional data requires the development of dedicated database systems. The main difference between geographical data and traditional data is their complexity caused by the mix of alphanumeric and geometric information needed to describe geographical objects. To address such a challenge, an already existing relational database system has been extended by enriching it with additional spatial datatypes and operators. One point specially addressed in GeoSabrina is the extensibility of the system. GeoSabrina provides a variety of different built-in common spatial operators. Thus, it is also expected that a large subset of possible needs of the users are covered. However, just for the case that some special requirements are poorly, or not-at-all, satisfied, the system allows users to create their own specific domains and/or specific operators.

In [5], a new spatial query type is presented that concerns the minimization of the total travel cost for a group trip planning action. Current location-aware mobile devices allow users to access location-based social networks from anywhere and to connect to friends via such networks in an easy manner. For example, users may want to spontaneously meet their friends for a dinner at a restaurant nearby followed by a joint visit to a movie theater. This use case motivates the creation of a new query type - referred to as a group trip planning query. Hereby, the group is interested to minimize the total travel distance for all members. The distance is defined as the sum of each user's travel distance from each user's start location to destination via the restaurant and the movie theater.

The study in [6] presents a group based approach for shortest path queries in road networks. As map-based applications consistently become popular, the service provider often requires to process a large number of simultaneous or contemporary queries. Thus, the intensiveness of these queries requires the efficient processing of related queries on spatial networks (i.e. road networks). However, traditional systems considering one query at a time are not convenient for many applications because of the relatively higher computational and service cost overhead. To this end, a group based approach has been proposed to calculate the shortest path between a source and a destination of the user. More specifically, the challenge of finding the shortest paths for a large number of simultaneous path queries in road networks is addressed. The key rationale of the approach is to group queries that share a common travel path and finally compute the shortest path for the group.

An another location-based service field gaining an increasing popularity is the development of friend-locator applications. They enable users to be notified if they are geographically close to any of the user's friends. Although this service looks very beneficial at first glance, it also entails a privacy issue as the users have to reveal their current location which in turn limits the attractiveness of the services. The challenge addressed in [7] is to develop a communication-efficient solution such that (i) it only detects proximity between a user and his friends, (ii) any other party is not allowed to infer the user's location, and (iii) users can flexibly configure the detection distance of their proximity.

Extensive research about the spatial index structures have been also conducted. For example, [8] has compared the performance of six R-tree variants including R-tree, R*-tree, Hilbert R-tree, CR-tree, CR*-tree and Hilbert CR-tree. Extensive experiments in the two categories of sequential accesses and concurrent accesses have been performed. It turned out that there is no R-tree variant that always performs best for all type of operations (i.e. for insert, update, delete and search operations). In more detail, the following observations could be gained from the experiments. In sequential accesses, CR*-trees are the best for search, Hilbert R-trees for update, and Hilbert CR-trees for a mixture of them. In concurrent accesses, Hilbert CR-trees for search if data is uniformly distributed, CR*-trees for search if data is skewed, Hilbert R-trees for update, and Hilbert CR-trees for a mixture of them.

Another work in [9] has intended to address the problem of bulk insertions into existing multidimensional index structures with a special focus on R-trees. As opposed to current and traditional techniques that insert data one by one, the proposed technique does it using bulk insertions. In particular, a large dataset is partitioned into sets of clusters. In the next step, a small R-tree from each cluster is constructed and suitable locations in the original large R-tree for insertion are identified and prepared. Finally, the insertions of the small tress are performed into the large R-tree in bulk. The experimental results achieved by using this technique revealed that it performs much better than the currently existing traditional techniques.

5 Conclusion

Spatial database systems provide structures for storage and analysis of spatial data that can be described as objects in multi-dimensional space. Spatial databases provide the capabilities of a traditional DBMS and also allow special storage and handling of spatial data. In more detail, they provide spatial data models and types as well as support spatial indices and querying languages specific to spatial data types for handling of spatial data and operations.

A brief overview about the existing spatial indices has been given which are mainly used to expedite the execution of search queries on spatial data. There exist various spatial indices including Grid files, Quadtrees and R-trees. Although there is no index structure that always performs best for any type of database operation (i.e. for insert, update, delete or search operation), the R-tree has gained a wide acceptance by the most spatial database systems and is used as the standard index structure. The key idea is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree.

A survey about the recent advances in spatial database research has been also conducted. It turned out that current research trends go into the direction of improving the existing location-based services or providing novel ones. For example, several location-based applications such as friend-locators, minimum travel cost and short-path calculators have been proposed in recent years. Also, it can easily be anticipated that this increasing trend of location-based and spatial data-based services will continue for the next years.

References

[1] Data storage and management in PostGIS, http://suite.opengeo.org/4.1/dataadmin/pgBasics/geometries.html, last visited Dec. 19, 2014.

[2] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching. In Proc. of ACM Special Interest Group on Management of Data, 1984, pp. 47–57.

[3] T. Larue, D. Pastre, Y. Viemont. Strong integration of spatial domains and operators in a relational database system. In Proc. of Intl. Symposium on Large Spatial Databases, Singapore, 1993, pp. 53-72.

[4] T. Hashem, T. Hashem, M. Eunus Ali and L. Kulik. Group Trip Planning Queries in Spatial Databases. In Proc of Intl. Symposium on Spatial and Temporal Databases, 2013, pp. 259-276.

[5] H. Mahmud, A. M. Amin, M. E. Ali, T. Hashem and S. Nutanong. A Group Based Approach for Path Queries in Road Networks. In Proc. of Intl. Symposium on Spatial and Temporal Databases, 2013, pp. 367-385.

[6] L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen. A location privacy aware friend locator. In Proc. of Intl. Symposium on Spatial and Temporal Databases, 2009, pp. 405-410.

[7] S. Hwang, K. Kwon, S.K. Cha, and B.S. Lee. Performance Evaluation of Main-Memory R-tree Variants. In Proc. of Advances in Spatial and Temporal Databases, Santorini Island (Greece), 2003, pp. 10-27.

[8] R. Choubey, L. Chen and E.A. Rundensteiner. GBI: A Generalized R-Tree Bulk-Insertion Strategy. In Proc. of Intl. Symposium on Spatial and Temporal Databases, 1999, pp. 91–108.