An Brief Introduction to Spatial Database Management Systems

Hashim Mohamed         Sihan Wei
moham992@umn.edu      wei00114@umn.edu

April 2, 2020

Abstract

Spatial database management systems form the underlying database technology for economic and societal significant applications in fields such as agriculture, navigation, and public health and safety. This article explores issues in storing spatial data in conventional database management systems and how SDBMS build upon conventional DBMSs to allow for simple and more efficient handling of spatial data. Popular data models for spatial information used in spatial database management systems are introduced and compared. Also covered is the topic of spatial indexing and how it differs from the conventional one dimensional indexing of traditional DBMS systems. Techniques such as space filling curves, and R-trees are introduced and compared to their conventional one dimensional counterparts. Spatial query processing is explored namely in regards to spatial joins. Example of these types of queries include point queries: Find all polygons containing a given point, range queries: finding all objects within a query circle and nearest neighbor queries: finding the spatial object closest to a query point. Finally, the article briefly introduces adjacent research areas such as query processing optimization, spatial-temporal data storage and spatial big data.

Keywords: Spatial Database Management Systems (SDBMS), Spatial Data Modeling, Spatial Query Languages, Spatial Query Processing, Spatial Storage and Indexing.

1 Introductory Concepts and Definitions

- **Spatial Data**: Any form of data that includes geospatial information. Examples may include a point with longitude and latitude numbers, a street address, or a road network represented as a graph.
• **Geographic Information System (GIS):** Specialized tools that allow visualization, modification and analysis of geospatial data. A popular example is ArcGIS.

• **Database Management System (DBMS):** A software designed to store and manipulate databases.

• **Database Indexing:** Organized structures designed to allow for quickly looking up values.

• **Spatial Database Management System (SDBMS):** A type of DBMS designed to support effective storage, indexing and querying of spatial data. Often built on top of an underlying database management system.

## 2 Introduction

Spatial database management systems form the underlying database technology for economic and societal significant applications across a diverse number of fields such as agriculture, navigation, and public health and safety. Whether it is a simple address book to a complicated neurological 3-D map of a brain, tools that facilitate the easy storage, access, analysis, and modification of spatially annotated data have scene high societal and economic demand. With the massive increase in GPS connected devices such as cell phones and remote sensors, the use and development of these systems is ever increasing [1].

### 2.1 Relation to GIS systems

Modern spatial database management systems partially trace their roots to early Graphic information systems (GIS) software such as such as ESRI’s ARC/INFO released in 1981 [2]. Designed aid in the visualization, modification and analysis of maps, these early systems were limited in the type and resolution of the data stored as well as the type of querying and processing available. These systems were some of the earliest explorations of the computerized storage and processing of spatial data. Competing systems lacked interoperability with each other or other software applications that may require access to spatial data. Basic spatial operations that are simple in modern SDBMS such as a query to see if two polygons intersected often had to be implemented in application code.
2.2 Building upon conventional DBMS

Modern spatial database management systems are also rooted upon base (not necessarily spatial) database management systems [3]. Development in database management systems have developed schemes that carry many advantages for improving data sharing and data security. DBMS makes data accessible and persistent across failures, and support thousands real time transactions while maintaining robustness. Another powerful and useful feature of DBMS is data integrity, which means that data is accurate and consistent. Such feature is necessary and important because there may be multiple databases in the DBMS. Data integrity ensures the correctness and consistency of data for all users. Another unneglectable advantage of DBMS is its high efficiency in sorting, indexing and search trees for 1-D data. Conventional utilize Structured Query Language (SQL) like top level is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management systems and non relational systems alike acting as defacto standard language for database management systems.

Traditional DBMS could only deal with non-spatial queries, e.g., listing names of all database related books in the library. If we want to handle spatial queries such as listing names of all Chinese restaurants in Minneapolis, traditional DBMS may not be that helpful. Many of the features that are included with traditional database management systems such as standardization, security, and data integrity are directly applicable to spatial data while others such as data indexing and query processing optimizations do not lend well to spatial data. The general trend in developments in SDBMS involves building on top of
3 SDBMS Core Data Models

In its most generic meaning a data model is a simplified representation, or a high level description of a real world system. Geographic Information Science (GIS) system generally models spatial data with a set of layers. Conventional databases organize data into a collection of tables each containing fields of simple abstract data types such as numbers, strings, and dates. While it is possible to store spatial data directly in these tables, for example encoding a polygon as a string, there are many problems with this approach. There are a number of popular models for spatial information used in spatial database management systems.

3.1 Spatial Fields/Rasters

The field data model is based on the partitioning of space. For example a geographical area can be partitioned by a set grid with spatial data encoded to collections of these grid cells [4]. Each grid cell stores all the information of the spatial information intersecting it. While this approach make certain operations like look up given a certain coordinates rather efficient, problems can arise in storing discrete objects that must be split among grid cells or the need for large / sparse areas while maintaining high resolutions. The usage of this model is generally limited to early GIS systems and specific domain applications.

Figure 3.1: Spatial Fields/Rasters. (Sourced from wikipedia under creative common license)
3.2 Spatial Objects/Vectors

Perhaps the most popular model for storing spatial data is using discrete objects like points, lines, and polygons. SDBMS typically use OGIS simple feature types inherited from GIS for the standardization of the types of objects and relations between them.

Figure 3.2: Lunar craters. (Source: jpl.nasa.com)

3.3 Spatial Networks

Certain spatial data can also be modeled in a spatial network. A spatial network is a graph with vertices and edges that represents spatial information. Examples may include a train network where stations are vertices and connections are edges (or vise versa). When modeled, network based spatial data often piggyback off built in graph modeling tools in database management systems. Often spatial networks have associated spatial object data. Multi-model database management systems such as OrientDB and NodeJS support both spatial and graph data.

Graph models are widely used in many application domains. A graph $G = (V, E)$ is a mathematical model, where $V$ represents a finite set of vertices and $E$ is a set of edges modeling a binary relationship between vertices. For example consider using a graph to model roadmaps with certain concepts like roads, road-intersections, road-segments, turns and so on [5]. Figure 3.3 shows the graph model representation.
4 Spatial Storage and Indexing

Conventional database management systems utilize various methods to store and index data to allow for efficient data storage uses and faster data querying. DBMSs typically support creating indexes on groups of records in a database. The index is a file containing an ordered set of pointers to records in a database. The ordering is always maintained by the DBMS on insertions and deletions. By maintaining an ordering, queries on records referenced by the indexed. Ordered data can be arranged into B-Trees.

These approaches are built for sortable one-dimensional data and thus do not handle spatial data objects that are usually multidimensional. Take for example a set of points on a 2D plane each denoted by longitude and latitude values. The set of 2D points must be given an 1D ordering in order to take advantage of B-trees. One method is ordering by just longitude or just latitude individually. While this may produce slightly better results than using no index at all but is not optimal as close by points may end up out of order. Some early approaches to spatial indexing involved using space filling curves to develop orderings that attempted to maintain some level of correlation between close-by coordinates [3][6]. Common space filling curves for this purpose are Hilbert curves and Z-curves. The overall principle is the same, producing a fixed ordering of the points of a map's region turning a 2D problem into a 1D problem. This allows use of built in B-tree indexing and other traditional database management systems. While effective for point queries these approaches get less effective and more complex when implementing range queries and supporting non point data such as lines and polygons. The ability to use traditional indexing solutions made these approaches popular in early spatial databases, they have however fallen out of favor with dedicated spatial indexing approaches such as R-trees and Quad trees.
4.1 R-Trees

R-trees are another widely used indexing structure that like the previous methods have a plentitude of variations [3,7]. R-trees generalize single dimensional B-Trees to multidimensional spatial data sets. Like B-trees they attempt to store data in a balanced hierarchy and maintain balance on insertion and deletion of data. Unlike B-trees which divvy data based on subranges of a single dimensional ordering, R-trees divvy data based on spatial regions such as rectangles. Spatial data objects in an R-tree form it’s leaf nodes while the nodes above in the hierarchy represent spatial regions that subsume all their child nodes.

(a) Lunar craters  
(b) R-Tree representation  
(c) Hierarchical structure

Figure 4.1: Lunar craters and an R-Tree representation (sourced from jpl.nasa.com)

Take for example the images above in Figure 4.1 representing a set of craters on a 2D map of a section of the moon and a representation of an accompanying R-tree structure. Suppose the task is to figure out if a given point represented by 2D coordinates intersects one of the lunar craters. Without an indexing structure one would need to check for intersection with every crater. With an R-tree, as represented in the figure above by simply first checking if the point is within the A or B rectangles you can eliminate the need to check half, or all of the craters.

4.2 Quad Trees

Like R-trees, quad trees divvy data in a tree-like structure where spatial objects in a region are represented by the leaves. Each node in the tree has exactly four children yet the tree is not necessarily balanced. An example is shown in the figure to the right. In addition to GIS and spatial databases, quadtrees are used in image compression systems. The Quad tree method loses
effectiveness for more complex shapes. Polygons may require being stored in many leaf nodes compromising performance. The best results occur with evenly distributed point data.

![Quad Tree Example](image)

Figure 4.2: Quad Tree Example

5 Spatial Query Processing

Query processing and optimization is the procedure of translating a SQL query to an execution plan, which is expressed in low-level language. Here the execution plan is a series of compilation steps a database system will go through when it receives a query of update or information retrieval. The goal of Query processing is to reduce run-time of an execution plan, so that the system answers a query in as little time as possible. However, there are some constraints, since the overheads are small and the optimization time is far less than the planned execution time.

5.1 Spatial Joins

In a conventional DBMS join operations relate entities based on the presence of a relation between fields. For example, conventional DBMS relations that are often used for joins are checking for equality or checking if a field in one record is greater than a field in another. Spatial joins on the other hand utilize spatial relations for example finding objects that are within a certain distance of each other or intersect each other. Both spatial and non-spatial joins depend on efficient indexing and storage methods for quick computation. Examples of queries that utilize spatial joins include:

- **Point query:** Find all rectangles containing a given point.
• **Range query**: Find all objects within a query rectangle.

• **Nearest neighbor**: Find the point closest to a query point.

• **Intersection query**: Find all the rectangles intersecting a query rectangle.

### 5.2 Spatial Network Queries

Graph databases typically include built in representations for nodes/vertices and edges and spatial network implementations often piggyback on built in graph features for queries. Exact details and supported graph functions differ by DBMS implementation and unlike OGIS objects and object relations no definite standard for spatial data. Database management systems that support graphs often include built-in functions for basic operations like shortest path between two nodes. These can be used as building blocks for more complex operations like calculating transitive closures of arbitrary length traversals. Some systems introduce spatial specific network algebra systems for modeling spatial networks in database systems [8].

### 5.3 Spatial Query Optimizations

Database management systems utilize complex query processing techniques to translate high level human readable queries to commands that can run efficiently on the underlying database system [9]. The complexities that come with spatial data often require new approaches to query optimization in a spatial context to bridge the gap between the high-level spatial queries and the simpler query language supported by the underlying spatial data-structure [10]. A naive direct translation of the spatial commands to basic table scans can get very computationally expensive as even basic spatial commands and joins can require break down to multiple redundant table scans.

Spatial query optimization systems use complicated mathematics that can order and modify commands in attempt reduce redundant database scans and increase query performance. Spatial queries can also be made more efficient with better utilization of hardware resources mainly through techniques to parallelize queries. One optimization method that applies to spatial queries is logical transformation. Such transformation is both accurate and efficient without changing the answer of a query. Generally speaking, push down **SELECT** operation below **JOIN**, reduce size of table for **JOIN** operation and push project down would be helpful. Figure (5.1) is an example of logical transformation by pushing down **JOIN** operation.
6 SDBMS Development Areas

The general trend in developments for spatial database management systems is the porting of one conventional database management development to a spatial context. Two primary areas are spatio-temporal database management systems and spatial Big Data systems.

Spatiotemporal database management systems extend basic SDBMS functionality with temporal data. As spatial data often reflects data in the real world, temporal information is often an essential dimension. There have been developments in temporal databases for non spatial data and much of the work has been toward developing spatial analogies [4]. Adding the temporal dimension to data often involves increases to storage space and querying time, two areas that are often already stressed by spatial data. There are a number of proposed modeling methods in the area but none completely mature.

Another area of concern is that of database management systems for Spatial Big Data applications. In its most general definition, Big Data is often defined as the data that could not be perceived, acquired, managed, and processed by traditional IT and software/hardware tools within a tolerable time [11]. Spatial Big Data (SBD) could be all types of data objects or elements that have geographical information present [12]. SBD examples include trajectories of cellphones and GPS devices, vehicle engine measurements from thousands of sensors across millions of vehicles, large temporally detailed road maps, etc. The massive increases of spatial data from internet connected geospatially aware devices such as cell phones and remote sensors make spatial data an important aspect of the big data problem. The general direction of database management systems for big data involves the use of parallelism and distributed storage and
processing with platforms like Hadoop. The novel developments in these areas
often leave behind traditional spatial database features as they are often not as
trivially parallelized. Trends include developing spatial database solutions to
Big Data platforms like Hadoop [13].

7 Learning Objectives

1. What differentiates a spatial database management system from a con-
   ventional DBMS

2. Identify issues with trying to encode and query spatial data such as a
   polygon as a String in a conventional, non-spatial, DBMS.

3. Describe problems with trying to use a traditional 1D index to index a set
   of 2D points.

4. List three spatial object types a spatial database may need to store.

8 Instructional Assessment Questions

1. Which of the following are examples of spatial data?
   
   (a) A list of street addresses
   (b) An anatomical model of a human body including labels for certain
       organs
   (c) A collection of short poems about the beauty of nature

2. Which of the following queries require a spatial join given a table with a
   list of cities and their geographical locations in longitude, latitude pairs.
   
   (a) What is the highest latitude city?
   (b) Where do cities tend to be clustered?
   (c) Which pair of cities are the closest to being exactly on the other side
       of the world from each other?

3. When modeling a road network as graph in a DBMS which of the following
   is true
   
   (a) Roads are edges and intersections are vertices.
   (b) Roads are vertices and intersections are edges.
   (c) Either can be true depending on the model.

4. What is the complexity of searching for a record in a sorted index
5. What is the complexity of searching for a record in a unsorted table
   (a) O(1)
   (b) O(log(n))
   (c) O(n)
   (d) O(nlog(n))

9 Additional Resources

The primary source for this tutorial was the book Spatial Databases: A Tour by S. Shekhar and S. Chawla[7]. Although somewhat dated, it covers primary aspects covered in this tutorial. An overview and slides are available at: http://www.spatial.cs.umn.edu/Book/

For implementation examples, tutorials, and documentation the PostGIS documentation webpage available at https://postgis.net/documentation/ is an excellent resource and entirely open source. A more general overview of OpenGIS simple features types and their specifications can be found here https://www.ogc.org/standards/sfo. Simple Features are an ISO standard and are implemented by a large number of spatial database management systems.

In regards to the SDBMS Development areas section, [4] and [11] proved relatively recent surveys in developments in spatial-temporal and big data database management systems respectively.

References


