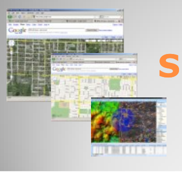


Introduction to Spatial Database Management System

Presenter
James Huynh

Discussion Lead
Srujan Kumar

Paper author: Ralf Hartmut Guting
Original slides by: Farnoush Banaei-Kashani

Outline

- Introduction & definition
- Modeling
- Querying
- Data structures and algorithms
- System architecture
- Conclusion and summary

Introduction

- Various fields/applications require management of geometric, geographic or spatial data:
  - A geographic space: surface of the earth
  - Man-made space: layout of VLSI design
  - Astronomy space: the universe
- Examples of non-spatial data:
  - Names, phone numbers, email addresses of people
- Examples of Spatial data:
  - NASA satellites imagery
  - Rivers, Farms

Introduction…

- Non-spatial query:
  - List the names of all bookstore <u>with more than ten thousand titles</u>
- Spatial query:
  - List the names of all bookstores **<u>with ten miles of Metrotown</u>**

Introduction…

- Common challenge: dealing with large collections of relatively simple geometric objects
- Different from image and pictorial database systems:
  - Containing sets of objects in space rather than images or pictures of a space

Definition

- A spatial database system:
  - Is a database system
  - Offers spatial data types (SDTs) in its data model and query language
  - Supports SDT in its implementation

Modeling

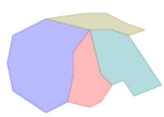
- Two basic things need to be represented:
  - Objects in space: cities, forests, or rivers
    - modeling single objects
  - Space: every point in space (e.g., partition of a country into districts)
    - modeling spatially related collections of objects

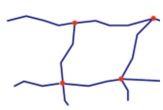
Modeling…

- Fundamental abstractions for modeling single objects:
  - Point: object represented only by its location in space, e.g., center of a state
  - Line (actually a curve or ployline): representation of moving through or connections in space, e.g., road, river
  - Region: representation of an extent in 2d-space, e.g., lake, city

Modeling…

- Instances of spatially related collections of objects:
  - Partition: set of region objects that are required to be disjoint (adjacency or region objects with common boundaries), e.g., thematic maps
  - Networks: embedded graph in plane consisting of set of points (vertices) and lines (edges) objects, e.g. highways, power supply lines, rivers





Modeling…

A sample (ROSE) spatial type system

EXT={lines, regions}, GEO={points, lines, regions}

- Spatial <u>predicates</u> for topological relationships:
  - inside: geo x regions → bool
  - intersect, meets: ext1 x ext2 → bool
  - adjacent, encloses: regions x regions → bool
- <u>Operations</u> returning atomic spatial data types:
  - intersection: lines x lines → points
  - intersection: regions x regions → regions
  - plus, minus: geo x geo → geo
  - contour: regions → lines

Modeling…

- Spatial <u>operators</u> returning numbers
  - dist: geo1 x geo2 → real
  - perimeter, area: regions → real
- Spatial <u>operations</u> on set of objects
  - sum: set(obj) x (obj→geo) → geo
- A spatial aggregate <u>function</u>, geometric union of all attribute values, e.g., union of set of provinces determine the area of the country
  - closest: set(obj) x (obj→geo1) x geo2 → set(obj)
  - Determines within a set of objects those whose spatial attribute value has minimal distance from geometric query object

Modeling…

- Spatial <u>relationships</u>:
  - Topological relationships: e.g., adjacent, inside, disjoint.
  - Direction relationships: e.g., above, below, or north_of, sothwest_of, …
  - Metric relationships: e.g., distance
- Enumeration of all possible topological relationships between two simple regions (no holes, connected):
  - Based on comparing two objects boundaries (δA) and interiors (Ao), filter to create <u>6 valid topological relationships</u>:
    - disjoint, in, touch, equal, cover, overlap

Modeling...

- DBMS data model must be extended by SDTs at the level of atomic data types (such as integer, string), or better be open for user-defined types (OR-DBMS approach):
  - relation states (sname: STRING; area: REGION; spop: INTEGER)
  - relation cities (cname: STRING; center: POINT; ext: REGION; cpop: INTEGER);
  - relation rivers (rname: STRING; route: LINE)

Discussion 1

- Most of the databases have some or the other aspect of location/spatiality.
- What examples can you think of where time is involved that you'd want the specialized support of spatial databases, and when would normal support be fine?

Querying

- Two main issues:
  - 1. Connecting the operations of a spatial algebra to the facilities of a DBMS query language.
  - 2. Providing graphical presentation of spatial data (i.e., results of queries), and graphical input of SDT values used in queries.

Querying...

- Fundamental spatial algebra operations:
  - Spatial selection: returning those objects satisfying a spatial predicate with the query object
    - **"All cities in Bavaria":**
      - SELECT sname FROM cities c WHERE c.center inside Bavaria.area
    - **"All rivers intersecting a query window"**
      - SELECT * FROM rivers r WHERE r.route intersects Window
    - **"All big cities no more than 100 Kms from Hagen"**
      - SELECT cname FROM cities c WHERE dist(c.center, Hagen.center) <100 and c.pop > 500k
  - Spatial join: A join which compares any two joined objects based on a predicate on their spatial attribute values.

Querying...

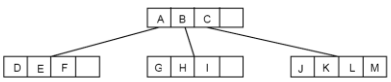
- "For each river pass through Bavaria, find all cities within less than 50 Kms."
  - **SELECT** r.rname, c.cname, length(intersection(r.route, c.area))
    FROM rivers r, cities c
    WHERE r.route intersects Bavaria.area and dist(r.route,c.area) < 50 Km
  - Graphical I/O issue: how to determine "Bavaria" (input); or how to show "intersection(route, Bavaria.area)" or "r.route" (output) (results are usually a combination of several queries).
- Requirements for spatial querying:
  - Spatial data types
  - Graphical display of query results
  - Graphical combination (overlay) of several query results (start a new picture, add/remove layers, change order of layers)
  - Display of context (e.g., show background such as a raster image (satellite image) or boundary of states)
  - Facility to check the content of a display (which query contributed to the content)

DBMS extensions required

- Representations for spatial algebra data types
- Procedures for the atomic operations (e.g., overlap)
- Spatial index structures
- Access operations for spatial indices (e.g., insert)
- Filter and refine techniques
- Spatial join algorithms
- Cost functions for all operations (for query optimizer)
- Statistics for estimating selectivity of spatial selection and join
- Extensions of optimizer to map queries into the specialized query processing method
- Spatial data types & operations within data definition and query language
- User interface extensions to handle graphical representation and input of SDT values

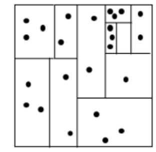
Spatial Indexing…

- To deal with spatial selection quickly and efficiently (as well as other operations such as spatial joins, …)
- It organizes space and the objects in it in some way so that only parts of the space and a subset of the objects need to be considered to answer query
- Two main approaches:
  - 1. Dedicated spatial data structures (e.g., R-tree)
  - 2. Spatial objects mapped to a 1-D space to utilize standard indexing techniques (e.g., B-tree)

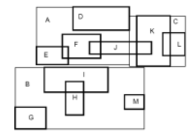


Spatial Indexing…

- A spatial index structure organizes points into buckets.
- Each bucket has an associated bucket region

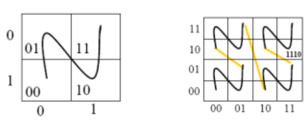


- For point data structures, the regions are disjoint & partition space so that each point belongs into precisely one bucket.
- For rectangle data structures, bucket regions may overlap.



Spatial Indexing…

- One dimensional embedding: z-order or bit-interleaving
  - Find a linear order for the cells of the grid while maintaining "locality" (i.e., cells close to each other in space are also close to each other in the linear order)
  - Define this order recursively for a grid that is obtained by hierarchical subdivision of space



Spatial join

- Traditional join methods such as hash join or sort/merge join are not applicable.
- Filtering cartesian product is expensive.
  - "For each river pass through Bavaria, find all cities within less than 50 Kms."
    - SELECT r.rname, c.cname, length(intersection(r.route, c.area))
      FROM rivers r, cities c
      WHERE r.route intersects Bavaria.area and dist(r.route,c.area) < 50 Km
- Central ideas:
  - filter + refine
  - use of spatial index structures

Discussion 2

- "Time and space are modes by which we think and not conditions in which we live" -- Albert Einstein
- Within Einstein's view time and space are equivalent quantities, and time is only a fourth dimension where existence resides.
- We can possibly think about merging the concepts of temporal databases and spatial databases and have everything defined within the framework of spatial databases.

➢ **Yes**, we can and **we should**.

➢ **No**, even if we technically can, **we should not**.

System Architecture Revisited

- The only clean way to accommodate these extensions is an integrated architecture based on the use of an extensible DBMS.
- Hence, current commercial solutions are OR-DBMSs:
  - IBM DB2 (spatial extenders)
  - Informix Universal Server (spatial datablade)
  - Oracle 10g (spatial cartridges)

Conclusion

- SDBMS is valuable to many important applications
- A spatial database system:
  - Is a database system which offers SDTs in its data model and query language and supports SDTs in its implementation, especially spatial indexing and spatial join
- Objects in space and space are two basic entities need to be modeled/represented
- Fundamental spatial algebra operations includes spatial selection, spatial join