Topic 1: Spatial Data Management

- **Spatial Database Systems** manage large collections of multidimensional objects (typically 2D/3D)
- A *spatial object* contains (at least) one spatial attribute that describes its geometry and location
- A *spatial relation* is an organized collection of spatial objects of the same entity (e.g., rivers, cities, road segments)

<table>
<thead>
<tr>
<th>Road Segment Name</th>
<th>ID</th>
<th>Start Location</th>
<th>End Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boulevard avenue</td>
<td>1</td>
<td>(10023,1094)</td>
<td>(9034,1567)</td>
</tr>
<tr>
<td>Leeds highway</td>
<td>2</td>
<td>(4240,5910)</td>
<td>(4129,6012)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3813,6129)</td>
<td>(3602,6129)</td>
</tr>
<tr>
<td>……</td>
<td>……</td>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

**Spatial Queries**

- **Range query (spatial selection, window query)**
  - e.g., find all cities that intersect window W
  - Answer set: \((c_1, c_2)\)
- **Nearest neighbor query**
  - e.g., find the city closest to the forest F
  - Answer: \(c_2\)
- **Spatial join**
  - e.g., find all pairs of cities and rivers that intersect
  - Answer set: \((r_1,c_1), (r_2,c_2), (r_2,c_5)\)

**Two-step Spatial Query Processing**

- Evaluating queries on geometric data is slow; a *spatial object* is approximated by its *minimum bounding rectangle* (MBR)
- The spatial query is then processed in two steps:
  1. **Filter step**: The MBR is tested against the query predicate
  2. **Refinement step**: The exact geometry of objects that pass the filter step is tested for qualification

**What Is Special About Spatial**

- There is no total ordering of objects in the multidimensional space that preserves spatial proximity
- Relational indexes (e.g., B+-trees) and query processing methods (e.g., sort-merge join, hash-join) are not readily applicable to spatial data
- Multidimensional access methods index spatial data and facilitate efficient processing of simple spatial query types (i.e., range queries)

**R-trees**

- A height balanced tree similar to B+-tree that indexes spatial objects
- Each node corresponds to a disk page and is at least 40% full
- Each node entry is a pair (MBR, ptr), that contains:
  - a pointer ptr to an indexed object or a lower level node
  - the MBR of the pointed object or node

- R-trees can be used to efficiently process the filter step of most spatial query types ... more in presentation 1

**Processing of Spatial Joins**

- The spatial join is an expensive predicate: \(O(n \cdot m)\) w.c. complexity
- Most spatial predicates on actual objects reduce to intersection of MBRs in the filter step. Thus spatial join algorithms consider mainly the filter step, using the intersect predicate
- Three categories of spatial join algorithms:
Index-based Spatial Join Methods

- The R-tree join (RJ):

  ![Diagram of R-tree join]

  Level 0 qualifying pairs: \{(a_1, b_1), (a_2, b_2)\}

  Level 1 qualifying pairs: \{(A_1, B_1), (A_2, B_2)\}

  ... more in presentation 2

Single-index Spatial Join Methods

- Indexed Nested Loops applies a window query for every object of the non-indexed dataset

- Seeded tree join and Bulk-load and Match build an on-the-fly R-tree and match it with the existing tree

- Sort and Match sorts the non-indexed dataset and packs leaf nodes without building the R-tree. Each node is matched with the existing tree

Spatial Join Algorithms on Row Data

- Spatial hash join

  ![Diagram of spatial hash join]

- Partition based spatial merge join

  ![Diagram of partition based spatial merge join]

Multi-way Spatial Joins

Given:

a) \(n\) datasets \(D_1, D_2, \ldots, D_n\)

b) an query graph \(Q\) where \(Q_{ij}\) is a spatial predicate that should hold between \(D_i\) and \(D_j\)

Find: all \(n\)-tuples \((r_1, r_2, \ldots, r_n)\), \(r_i \in D_i\), such that \(\forall i, j r_i Q_{ij} r_j\)

Examples:

- find all cities adjacent to forests which are intersected by a river
- find all VLSI sub-circuits that formulate a specific topological configuration

Combining 2-way Methods for Multi-way Joins

- Pairwise spatial join algorithms are implemented as operators that exchange data in a multiway join execution engine

- Large differences between alternative plans (orders of magnitude). Query optimization requires the following:
  a) Accurate cost formulae for pairwise join algorithms
  b) Accurate estimation of the output size of a sub-query

Alternative: Synchronous R-tree Traversal (ST)

- Extension of R-tree join for multiple inputs

- Combination of ST with 2-way algorithms is the best choice in many cases:
Nearest Neighbor Search

- Indexing can accelerate processing of NN queries
- In presentation 3 you will see branch and bound algorithms which perform nearest neighbor search on data indexed by R-trees
- There are more complex query types related to nearest neighbor search and spatial joins:
  - distance join query:
    Find pairs of hotels and restaurants within 500m from each other
  - closest pairs query:
    Find the 100 closest <hotels,restaurants> pairs
  - incremental NN and CP queries:
    Output all <hotels,restaurants> in increasing distance

Other Topics in Spatial Databases

- Selectivity estimation for query optimization
  Example:
  - relation: Cities(id, name, population, geometry)
  - B-tree on population, R-tree on geometry
  - query: Find cities with population>5,000 contained in W
  - Evaluation plan 1: Use B-tree to find large cities, then see if in W
  - Evaluation plan 2: Use R-tree to cities in W, then see if large
  - In order to choose the best plan we need accurate models that estimate the selectivity of spatial queries (selections, joins, etc.)
- Mining spatial data
  Example:
  - 80% of schools that are close to sports centers are also close to parks

Summary

- Managing large collections of spatial data is a large research topic with still many open research problems
- Apart from the intuitive applications (like GIS data management), many spatial data management problems are transformations of other problems to geometric ones (e.g., clustering)
- In the presentations that follow we will see in more detail some fundamental problems in spatial data management:
  1. Indexing
     - The R*-tree (an improved R-tree)
  2. Query processing
     - Processing spatial joins using R-trees
     - Processing NN-queries using R-trees