SPATIAL DATABASE

Ahmad Alhilal, Dimitris Tsaras
Content

• What is Spatial DB
• Modeling Spatial DB
• Spatial Queries
• The R-tree
• Range Query
• NN Query
• Aggregation Query
• RNN Query
• NN Queries with Validity Information
What is spatial database

Spatial Relationships

Spatial Data Types

Spatial Index

Road Network

Satellite Image
SDBMS Definition

A spatial database system:

- **Is a database system**
  - A DBMS with additional capabilities for handling spatial data

- Offers **spatial data types** (SDTs) in its data model and query language
  - Structure in space: e.g., POINT, LINE, REGION
  - Relationships among them: \((l \text{ intersects } r)\)

- Supports SDT in its implementation providing at least
  - **spatial indexing** (retrieving objects in particular area without scanning the whole space)
  - **efficient algorithms** for spatial joins (not simply filtering the cartesian product)
Modeling: spatial primitives for objects

• **Point**: object represented only by its location in space, e.g. center of a state

• **Line** (actually a curve or polyline): 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: Coverages

- **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 spatial type system (1)

\[ \text{EXT} = \{\text{lines, regions}\}, \ \text{GEO} = \{\text{points, lines, regions}\} \]

- **Spatial predicates for topological relationships:**
  - `inside`: \( \text{geo} \times \text{regions} \rightarrow \text{bool} \)
  - `intersect, meets`: \( \text{ext1} \times \text{ext2} \rightarrow \text{bool} \)
  - `adjacent, encloses`: \( \text{regions} \times \text{regions} \rightarrow \text{bool} \)

- **Operations returning atomic spatial data types:**
  - `intersection`: \( \text{lines} \times \text{lines} \rightarrow \text{points} \)
  - `intersection`: \( \text{regions} \times \text{regions} \rightarrow \text{regions} \)
  - `plus, minus`: \( \text{geo} \times \text{geo} \rightarrow \text{geo} \)
  - `contour`: \( \text{regions} \rightarrow \text{lines} \)
Modeling: a sample spatial type system (2)

- Spatial operators returning numbers
  - \textbf{dist}: \textit{geo1} \times \textit{geo2} \rightarrow \textit{real}
  - \textbf{perimeter}, \textbf{area}: \textit{regions} \rightarrow \textit{real}

- Spatial operations on set of objects
  - \textbf{sum}: \textit{set(obj)} \times \textit{objgeo} \rightarrow \textit{geo}
    - A spatial aggregate function, geometric union of all attribute values, e.g. \textit{union} of set of provinces determine the area of the country
  - \textbf{closest}: \textit{set(obj)} \times \textit{objgeo1} \times \textit{geo2} \rightarrow \textit{set(obj)}
    - Determines within a set of objects those whose spatial attribute value has minimal distance from geometric query object

- Other complex operations: \textit{overlay}, \textit{buffering}, …
Modeling: spatial relationships

- **Topological** relationships: e.g. adjacent, inside, disjoint.
- **Direction** relationships: e.g. above, below, or north_of, southwest_of, ...
- **Metric** relationships: e.g. distance

Valid topological relationships between two simple regions:

*disjoint, in, touch, equal, cover, overlap*
Modeling: SDBMS data model

• 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)
What is spatial database

Spatial Index

Spatial Relationships

Spatial Data Types

5. spatial Join
aggregate
NN
RNN

Road Network
Satellite Image
Spatial Queries

- **Range query** (spatial selection, window query)
  e.g. find all cities that intersect window W
  Answer set: \{c1, c2\}

- **Nearest neighbor** query
  e.g. find the city closest to the forest F
  Answer: c2

- **Aggregate query** (Count, Sum, AVG, MIN. . .)

- **Spatial join**
  e.g. find all pairs of cities and rivers that intersect
  Answer set: \{(r1,c1), (r2,c2), (r2,c5)\}
Spatial Queries: spatial selection

- *Spatial selection*: returning those objects satisfying a spatial predicate with the query object
  - “All cities in Ontario”
    ```sql
    SELECT sname FROM cities c WHERE c.center inside Ontario.area
    ```
  - “All rivers intersecting a query window”
    ```sql
    SELECT * FROM rivers r WHERE r.route intersects Window
    ```
  - “All big cities no more than 100 Kms from Toronto”
    ```sql
    SELECT cname FROM cities c
    WHERE dist(c.center, Toronto.center) < 100 and c.pop > 500k
    ```
  (conjunction with other predicates and query optimization)
Spatial Queries: spatial join

• *Spatial join*: A join which compares any two joined objects based on a predicate on their spatial attribute values.

• “For each river pass through Ontario, 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 ontario.area and dist(r.route,c.area) < 50
```
Spatial Indexing: Operations

• Spatial data structures either store points or rectangles (for line or region values)

• Operations on those structures: insert, delete, member

• Query types for points:
  – Range query: all points within a query rectangle
  – Nearest neighbor: point closest to a query point
  – Distance scan: enumerate points in increasing distance from a query point.

• Query types for rectangles
  – Intersection query
  – Containment query
R-Tree Motivation

**Range query**: find the objects in a given range.
E.g. find all hotels in Boston.

No index: scan through all objects. NOT EFFICIENT!
R-Tree: Clustering by Proximity

Minimum Bounding Rectangle (MBR)
R-Tree
R-Tree
R-Tree: Operation

\[ m \leq \text{node Utilization} \leq M \]

- **Insertion**:
  - Overflow: split

- **Deletion**:
  - Underflow: re-insert

- **Bulk load**

  Sort objects (e.g., x-coordinates, Hilbert value)

  Insert sorted objects in nodes
Spatial Queries: spatial join

Index-based Join
The R-Tree Join (RJ)

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

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

The diagram illustrates a range query in a multi-dimensional space. The figure shows a grid with points labeled from a to m. The space is divided into regions labeled E1, E2, E3, E4, E5, E6, and E7. The query is represented by the shaded area, which indicates the points that fall within the specified range.

The x and y axes are labeled, and the grid helps to visualize the spatial distribution of the points and the regions they are associated with.
Range Query

- **E1**
- **E2**
- **E3**
- **E4**
- **E5**
- **E6**
- **E7**

- **Root**

- **x axis**
- **y axis**

- Points:
  - a
  - b
  - c
  - d
  - e
  - f
  - g
  - h
  - i
  - j
  - k
  - l
  - m
Content

• What is Spatial DB
• Modeling Spatial DB
• Spatial Queries
• The R-tree
• Range Query

• **NN Query**
• Aggregation Query
• RNN Query
• NN Queries with Validity Information
Nearest Neighbor (NN) Query

• Given a query location $q$, find the nearest object.

• E.g.: given a hotel, find its nearest bar.
Metrics: MINDIST, MINMAXDIST

- **MINDIST**: Minimum distance between \( q \) and an MBR.
  
  ![Diagram](image.png)

- It is an lower bound of \( d(o, q) \) for every object \( o \) in \( E1 \).

- **MINMAXDIST**:
  
  - compute max dist between \( q \) and each edge of \( E_1 \), then take min.
  
  - It is an upper bound of \( d(o, q) \) for every object \( o \) in \( E1 \).
Refinement Phase (Pruning)

It prune the qualifying results based on three rules:

• Downward pruning:
  An MBR R is discarded if there exists another R’ such that
  \( \text{MINDIST}(q,R) > \text{MINMAXDIST}(q,R’) \)

• Downward pruning:
  An object O is discarded if there exists an R such that the
  \( \text{Actual-Dist}(q,O) > \text{MINMAXDIST}(q,R) \)

• Upward pruning:
  An MBR R is discarded if an object O is found such that the
  \( \text{MINDIST}(q,R) > \text{ActualDist}(q,O) \)
Pruning 1 in NN Query

- If we see an object $o$, prune every MBR whose $\text{MINDIST} > d(o, q)$.

- Side notice: at most one object in $H$!
Pruning 2 using MINMAXDIST

• Prune even before we see an object!
• Prune $E_1$ if exists $E_2$ s.t.
  \[ \text{MINDIST}(q, E_1) > \text{MINMAXDIST}(q, E_2). \]

• $\exists$ Object $o$ in sub-tree of $E_2$ s.t. $d(o, q) \leq \text{MINMAXDIST}(q, E_2)$

• MINMAXDIST: compute max dist between $q$ and each edge of $E_2$, then take min.
Depth-First (DF) NN Algorithm

Note: distances not actually stored inside nodes. Only for illustration.
Depth-First (DF) NN Algorithm
Depth-First (DF) NN Algorithm

First Candidate NN: a with distance $\sqrt{5}$
Depth-First (DF) NN Algorithm

Backtrack to $E_1$ and Root
Then visit $E_2$
Depth-First (DF) NN Algorithm

Root

\[
\begin{array}{cc}
E_1 & E_2 \\
\sqrt{1} & \sqrt{2} \\
\end{array}
\]

Actual NN: i with distance \( \sqrt{2} \)

\[
\begin{array}{cccccccc}
E_3 & E_4 & E_5 & E_6 & E_7 \\
\sqrt{5} & \sqrt{9} & \sqrt{5} & \sqrt{2} & \sqrt{13} \\
E_1 & E_2 & E_3 & E_4 & E_5 \\
\sqrt{1} & \sqrt{2} & \sqrt{9} & \sqrt{5} & \sqrt{5} \\
E_3 & E_4 & E_5 & E_6 & E_7 \\
\end{array}
\]
Best-First NN Algorithm

• Keep a heap $H$ of index entries and objects, ordered by MINDIST.
• Initially, $H$ contains the root.
• While $H \neq \emptyset$
  • Extract the element with minimum MINDIST
  • If it is an index entry, insert its children into $H$.
  • If it is an object, return it as NN.
• End while
Best-First NN Algorithm

Action
Heap

Visit Root

Root
### Best-First NN Algorithm

**Diagram:**
- **x axis:** 0, 2, 4, 6, 8, 10
- **y axis:** 0, 2, 4, 6, 8, 10

**Nodes:**
- **Root:** E₁
- **Child Nodes:**
  - E₂
  - E₃
  - E₄
  - E₅
  - E₆
  - E₇

**Queries:**
- a, b, c, d, e, f, g, h, i, j, k, l, m

**Heap:**
<table>
<thead>
<tr>
<th>Action</th>
<th>Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit Root</td>
<td>E₁/1, E₂/2</td>
</tr>
<tr>
<td>follow E₁</td>
<td>E₂/2, E₃/5, E₅/5, E₄/9</td>
</tr>
</tbody>
</table>

**Edges:**
- E₁ to E₂
- E₁ to E₃
- E₁ to E₄
- E₁ to E₅
- E₃ to E₄
- E₃ to E₅
- E₅ to E₆
- E₅ to E₇
- E₆ to E₇
Best-First NN Algorithm

**Diagram:**
- **x axis:** a, b, c, d, e, f, g, h, i, j, k, l, m
- **y axis:** query
- **Action:** Visit Root, follow $E_1$, follow $E_2$
- **Heap:**
  - Visit Root:
    - $E_1$ with cost 1
    - $E_2$ with cost 2
  - follow $E_1$:
    - $E_3$ with cost 5
    - $E_4$ with cost 9
    - $E_5$ with cost 5
  - follow $E_2$:
    - $E_6$ with cost 2
    - $E_7$ with cost 13
Best-First NN Algorithm

Visit Root
follow $E_1$
follow $E_2$
follow $E_6$

Action

Heap

$E_1/1$ $E_2/2$
$E_2/2$ $E_3/5$ $E_4/9$ $E_5/5$
$E_6/2$ $E_3/5$ $E_4/9$ $E_7/13$
$i/2$ $E_3/5$ $E_4/9$ $j/10$ $E_7/13$ $k/13$
Best-First NN Algorithm

Visit Root
follow $E_1$
follow $E_2$
follow $E_6$

Action

Heap

Visit Root $E_1/1$ $E_2/2$
follow $E_1$ $E_2/2$ $E_3/3$ $E_4/5$ $E_5/5$ $E_7/9$
follow $E_2$ $E_2/2$ $E_3/3$ $E_4/5$ $E_5/5$ $E_7/9$
follow $E_6$ $i/2$ $E_3/3$ $E_4/5$ $E_5/5$ $E_7/9$ $j/10$ $7/13$ $k/13$

Report i and terminate

Root

$E_1/1$ $E_2/2$

$E_3/5$ $E_4/9$ $E_5/5$

$E_6/2$ $E_7/13$

$a$ $b$ $c$ $d$ $e$ $f$ $g$ $h$ $i$ $j$ $k$ $l$ $m$
Discussion NN Queries

- Both DF and BF can be easily adapted to
  - I. extended (instead of point) objects
  - II. retrieval of k (>1) NN.

- BF is incremental; i.e., it can report the NN in increasing order of distance without a given value of k.
Aggregation Query

- Given a range, find some aggregate value of objects in this range.
  - **Distributive**: COUNT, SUM, MIN, MAX
    - E.g. find the total number of hotels in HK.
  - **Algebraic**: AVG
  - **Holistic**: Median

- Straightforward approach: reduce to a range query.

- Better approach: along with each index entry, store aggregate of the sub-tree
Aggregation Query: COUNT

The Aggregate R-Tree
aR-Tree
Aggregation Query: COUNT

Subtree pruned!
Aggregation Query: COUNT

Partially Intersects q Window
Minimum Aggregate Distance

- Given a set $P$ of data points and a set $Q$ of query points, an aggregate NN query returns the data point $p$ with the minimum aggregate distance.

- The aggregate distance between a data point $p$ and $Q=\{q_1, \ldots, q_n\}$ is defined as $adist(p,Q) = f(|pq_1|, \ldots, |pq_n|)$, where $|pq_i|$ is the Euclidean distance of $p$ and $q_i$. We use $f=\text{sum}$ in the examples.
Multiple Query Method (MQM)

- Idea based on the *threshold* algorithm for top-\(k\) queries: Perform **incremental** NN queries for each point in \(Q\) and combine their results.

- \(<p_{10}, 7>, <p_{11}, 6>\), Threshold \(T=5\) (2+3)

- Every non-yet visited point must have distance \(\geq T=5\)

- Since the best candidate found \(p_{11}\) has distance \(> T\) we must continue search
**Multiple Query Method (MQM)**

- Idea based on the threshold algorithm for top-$k$ queries: Perform **incremental** NN queries for each point in $Q$ and combine their results.

- *We find* $p_{11}$ again, this time as the second NN of $q_1$
- The value of the threshold is updated to $T=6$
- We already have a candidate with distance equal to $T=6$
- Search terminates
Minimum Bounding Method (MBM)

- MQM applies multiple NN queries and may visit the same node and data point multiple times
- MBM Applies the MBR of \( Q \) to prune the search space with a single (DF or BF) traversal, using heuristics to prune the search space

- **Heuristic 1:** Let \( M \) be the MBR of \( Q \), and \( \text{best\_dist} \) be the distance of the best points found so far. A node \( N \) cannot contain qualifying points, if:

\[
\text{mindist}(N, M) \geq \frac{\text{best\_dist}}{n}
\]

- **Heuristic 2:** A node \( N \) cannot contain qualifying points, if:

\[
\sum_{q_i \in Q} \text{mindist}(N, q_i) \geq \text{best\_dist}
\]
Content

• What is Spatial DB
• Modeling Spatial DB
• Spatial Queries
• The R-tree
• Range Query
• NN Queries
• Aggregation Query
• RNN Query
• NN Queries with Validity Information
Reverse NN Queries

- **Monochromatic**: given a multi-dimensional dataset $P$ and a point $q$, find all the points $p \in P$ that have $q$ as their nearest neighbor.
- **Bichromatic**: given a set $Q$ of queries and a query point $q$, find the objects $p \in P$ that are closer to $q$ than any other point of $Q$.
- E.g. I want to open a hotel such that it is the NN for the most tourist sights.
**KM Algorithm (for static datasets)**

- Find the NN of every data point $p$ - let the *vicinity circle* $(p, dist(p, NN(p)))$ centered at $p$ with radius equal to the Euclidean distance between $p$ and its NN.
- Index the MBRs of all circles with an R-tree, called the RNN-tree.
- The reverse nearest neighbors of $q$ are retrieved by a point location query on the RNN-tree, which returns all circles that contain $q$. 

![Diagram showing KM Algorithm](attachment:image.png)
SAA Algorithm (supports updates)

- Divide the space around the query $q$ into six equal regions $S_1$ to $S_6$.
- Find the NN $p_i$ of $q$ in each region $S_i$.
- Find the NN of each $p_i$
  - if $\text{distance}(p_i, \text{NN}(p_i)) < \text{distance}(p_i, q)$ there is no RNN of $q$ in $S_i$
  - otherwise, the only RNN of $q$ in $S_i$ is $p_i$. 
TLPL Algorithm (supports updates, >2 dimensions, $k$-RNN)

- Filter-refinement approach
  - Find the set $S_{cnd}$ of candidate points
    - Find neighbors of the query point $q$ incrementally
    - Every new neighbor prunes the search space
    - Continue until the entire space is pruned
    - Keep all the pruned points and nodes in a set $S_{rfn}$
- Refinement step: eliminate false positives from $S_{cnd}$
Example of TPL

data R-tree

contents omitted

contents omitted

....
Filter Step – Visit Root
Filter Step – Visit $N_{10}$
Filter Step – Visit \( N_3 \)
Filter Step – Visit $N_{11}$

Action: visit $N_{11}$  
Heap: $\{N_5, N_2, N_1, N_{12}\}$  
$S_{\text{end}}$: $\{p_1\}$  
$S_{\text{rfn}}$: $\{p_3, N_4, N_6\}$

data R-tree
Filter Step – Visit $N_5$

Action: visit $N_5$

Heap: $\{N_2, N_1, N_{12}\}$

$S_{cnd}$: $\{p_1, p_2\}$

$S_{ifn}$: $\{p_3, N_4, N_6, p_6\}$
Filter Step – Visit \( N_1 \)

**Action**

Visit \( N_1 \)

**Heap**

\( \{ N_{12} \} \)

**\( S_{\text{cmd}} \)**

\( \{ p_1, p_2, p_5 \} \)

**\( S_{\text{rfn}} \)**

\( \{ p_3, N_4, N_6, p_6, N_2, p_7 \} \)

---

Data R-tree

**Contents omitted**
Example (end of filter step)

Action

Heap

$S_{cnd}$

$S_{rfn}$

$\emptyset$

$\{ p_1, p_2, p_5 \}$

$\{ p_3, N_4, N_6, p_6, N_2, p_7, N_{12} \}$
Refinement Step

- Nodes outside the circles centered at the candidates can be discarded
- $p_3$ invalidates candidate $p_1$
- In order to verify candidate $p_2$ we need to visit $N_4$
- $p_5$ is an actual RNN

Other Work:
- Bichromatic RNN, Stanoi et al., VLDB 2001
- Road Networks, Yiu et al., TKDE 2006
- Metric Spaces, Tao et al., TKDE 2006
- Continuous RNN, Benetis et al., VLDBJ 2006
Chapter 5. NN Queries with Validity Information

• The result of conventional NN queries may be invalidated as soon as the query or some data object(s) move
• Goal: To return some additional information that determines the validity of the current result
• The validity region is such that the NN set remains the same as long as q remains in the region.
• The validity region of a NN query q is the Voronoi Cell of the NN o.
Validity Time

- Settings: Static Objects, Moving Query. Client/server architecture
- A central server indexes the Voronoi diagram of the objects
- Given a query, the server returns the current NN and the validity time of the result. The current NN is the object whose Voronoi cell covers the query point
- Restrictions: (i) assumes a maximum speed (ii) applicable only to single NN (iii) and static objects.