CMPUT 391
Database Management Systems

Spatial Data Management
Spatial Data Management

- Shortcomings of Relational Databases and ORDBMS
- Modeling Spatial Data
- Spatial Queries
- Space-Filling Curves + B-Trees
- R-trees
The Need for a DBMS

• On one hand we have a tremendous increase in the amount of data applications have to handle, on the other hand we want a reduced application development time.
  – Object-Oriented programming
  – DBMS features: query capability with optimization, concurrency control, recovery, indexing, etc.

• Can we merge these two to get an object database management system since data is getting more complex?
Manipulating New Kinds of Data

• A television channel needs to store video sequences, radio interviews, multimedia documents, geographical information, etc., and retrieve them efficiently.

• A movie producing company needs to store movies, frame sequences, data about actors and theaters, etc.

• A biological lab needs to store complex data about molecules, chromosomes, etc, and retrieve parts of data as well as complete data.
What are the Needs?

• Images
• Video
• Multimedia in general
• Spatial data (GIS)
• Biological data
• CAD data
• Virtual Worlds
• Games
• List of lists
• User defined data types
Shortcomings with RDBMS

• Supports only a small fixed collection of relatively simple data types (integers, floating point numbers, date, strings)
• No set-valued attributes (sets, lists,…)
• No inheritance in the Is-a relationship
• No complex objects, apart from BLOB (binary large object) and CLOB (character large object)
• Impedance mismatch between data access language (declarative SQL) and host language (procedural C or Java): programmer must explicitly tell how things to be done.

⇒ Is there a different solution?
Existing Object Databases

• Object database is a persistent storage manager for objects:
  – Persistent storage for object-oriented programming languages (C++, SmallTalk, etc.)
  – Object-Database Systems:
    • Object-Oriented Database Systems: alternative to relational systems
    • Object-Relational Database Systems: Extension to relational systems
  • Market: RDBMS ($8 billion), OODMS ($30 million) world-wide
  • OODB Commercial Products: ObjectStore, GemStone, Orion, etc.
DBMS Classification Matrix

<table>
<thead>
<tr>
<th>Query</th>
<th>Simple Data</th>
<th>Complex Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Query</td>
<td>File System</td>
<td>Object-Oriented DBMS</td>
</tr>
<tr>
<td></td>
<td>Relational DBMS</td>
<td>Object-Relational DBMS</td>
</tr>
</tbody>
</table>
Object-Relational Features of Oracle

Methods

CREATE TYPE Rectangle_typ AS OBJECT (  
    len NUMBER,  
    wid NUMBER,  
    MEMBER FUNCTION area RETURN NUMBER,  
);  

CREATE TYPE BODY Rectangle_typ AS  
    MEMBER FUNCTION area RETURN NUMBER IS  
        BEGIN  
            RETURN len * wid;  
        END area;  
    END;
Object-Relational Features of Oracle

Collection types / nested tables

CREATE TYPE PointType AS OBJECT (  
    x NUMBER,  
    y NUMBER);

CREATE TYPE PolygonType AS TABLE OF PointType;

CREATE TABLE Polygons (  
    name VARCHAR2(20),  
    points PolygonType)  
  NESTED TABLE points STORE AS PointsTable;

The relations representing individual polygons are not stored directly as values of the points attribute; they are stored in a single table, PointsTable.
Spatial Data Management

- Shortcomings of Relational Databases
- Modeling Spatial Data
- Spatial Queries
- Space-Filling Curves + B-Trees
- R-trees
Relational Representation of Spatial Data

- *Example*: Representation of geometric objects (here: parcels/fields of land) in normalized relations

Redundancy free representation requires distribution of the information over 3 tables: *Parcels, Borders, Points*
Relational Representation of Spatial Data

• For (spatial) queries involving parcels it is necessary to reconstruct the spatial information from the different tables
  – E.g.: if we want to determine if a given point P is inside parcel F₂, we have to find all corner-points of parcel F₂ first

    ```sql
    SELECT Points.PNr, X-Coord, Y-Coord
    FROM Parcels, Border, Points
    WHERE FNr = 'F₂' AND Parcel.BNr = Borders.BNr AND
      (Borders.PNr₁ = Points.PNr OR Borders.PNr₂ = Points.PNr)
    ```

• Even this simple query requires expensive joins of three tables
• Querying the geometry (e.g., P in F₂?) is not directly supported.
Extension of the Relational Model to Support Spatial Data

• Integration of spatial data types and operations into the core of a DBMS (object-oriented and object-relational databases)
  – Data types such as Point, Line, Polygon
  – Operations such as ObjectIntersect, RangeQuery, etc.

• Advantages
  – Natural extension of the relational model and query languages
  – Facilitates design and querying of spatial databases
  – Spatial data types and operations can be supported by spatial index structures and efficient algorithms, implemented in the core of a DBMS

• All major database vendors today implement support for spatial data and operations in their database systems via object-relational extensions
Extension of the Relational Model to Support Spatial Data – Example

Relation: **ForestZones** (Zone: *Polygon*, ForestOfficial: *String*, Area: *Cardinal*)

- The province decides that a reforestation is necessary in an area described by a polygon S. Find all forest officials affected by this decision.

```sql
SELECT ForestOfficial
FROM ForestZones
WHERE ObjectIntersects (S, Zone)
```
Data Types for Spatial Objects

- Spatial objects are described by
  - Spatial Extent
    - location and/or boundary with respect to a reference point in a coordinate system, which is at least 2-dimensional.
    - Basic object types: Point, Lines, Polygon
  - Other Non-Spatial Attributes
    - Thematic attributes such as height, area, name, land-use, etc.
Spatial Data Management

- Shortcomings of Relational Databases
- Modeling Spatial Data
- Spatial Queries
- Space-Filling Curves + B-Trees
- R-trees
Spatial Query Processing

- DBMS has to support two types of operations
  - Operations to retrieve certain subsets of spatial object from the database
    - “Spatial Queries/Selections”, e.g., window query, point query, etc.
  - Operations that perform basic geometric computations and tests
    - E.g., point in polygon test, intersection of two polygons etc.

- Spatial selections, e.g. in geographic information systems, are often supported by an interactive graphical user interface
Basic Spatial Queries

- **Containment Query**: Given a spatial object R, find all objects that completely contain R. If R is a Point: **Point Query**

- **Region Query**: Given a region R (polygon or circle), find all spatial objects that intersect with R. If R is a rectangle: **Window Query**

- **Enclosure Query**: Given a polygon region R, find all objects that are completely contained in R

- **K-Nearest Neighbor Query**: Given an object P, find the k objects that are closest to P (typically for points)
Basic Spatial Operation – Spatial Join

- Given two sets of spatial objects (typically minimum bounding rectangles)
  - \( S_1 = \{R_1, R_2, \ldots, R_m\} \) and \( S_2 = \{R'_1, R'_2, \ldots, R'_n\} \)

- Spatial Join: Compute all pairs of objects \((R, R')\) such that
  - \( R \in S_1, R' \in S_2, \)
  - and \( R \) intersects \( R' \) (\( R \cap R' \neq \emptyset \))
  - Spatial predicates other than intersection are also possible, e.g. all pairs of objects that are within a certain distance from each other

\[
\{A1, \ldots, A6\} \Join \{B1, \ldots, B3\}
\]

Answer Set

\[
(A5, B1) \\
(A4, B1) \\
(A1, B2) \\
(A6, B2) \\
(A2, B3)
\]

\( \text{Spatial-Join} \)
Index Support for Spatial Queries

• Conventional index structures such as B-trees are not designed to support spatial queries
  – Group objects only along one dimension
  – Do not preserve spatial proximity

  • E.g. nearest neighbor query:
    Nearest neighbor of Q is typically not the nearest neighbor in any single dimension

A and B closer in the X dimension; C and D closer in the Y dimension.
Index Support for Spatial Queries

• Spatial index structures try to preserve spatial proximity
  – Group objects that are close to each other on the same data page
  – Problem: the number of bytes to store extended spatial objects (lines, polygons) varies
  – Solution:
    • Store *Approximations* of spatial objects in the index structure, typically axis-parallel minimum bounding rectangles (MBR)
    • Exact object representation (ER) stored separately; pointers to ER in the index
Query Processing Using Approximations

Two-Step Procedure

1. Filter Step:
   - Use the index to find all approximations that satisfy the query
   - Some objects already satisfy the query based on the approximation, others have to be checked in the refinement step → Candidate Set

2. Refinement Step:
   - Load the exact object representations for candidates left after the filter step and test whether they satisfy the query

- a and b are certainly answers
- f, d, and g are certainly not answers
- c and e are candidates
- c is a false hit

Why?
Spatial Data Management

• Shortcomings of Relational Databases
• Modeling Spatial Data
• Spatial Queries
• Space-Filling Curves + B-Trees
• R-trees
Embedding of the 2-dimensional space into a 1 dimensional space

• Basic Idea:
  – The data space is partitioned into rectangular cells.
  – Use a space filling curve to assign cell numbers to the cells (define a linear order on the cells)
    • The curve should preserve spatial proximity as good as possible
    • Cell numbers should be easy to compute
  – Objects are approximated by cells.
  – Store the cell numbers for objects in a conventional index structure with respect to the linear order
Space Filling Curves

Lexicographic Order

- $Z$-Order preserves spatial proximity relatively good
- $Z$-Order is easy to compute

Hilbert-Curve
Z-Order – Z-Values

• Coding of Cells
  – Partition the data space recursively into two halves
  – Alternate X and Y dimension
  – Left/bottom → 0
  – Right/top → 1

  – Z-Value: \((c, l)\)
    \(c = \) decimal value of the bit string
    \(l = \) level (number of bits)

    if all cells are on the same level, then \(l\) can be omitted
Z-Order – Representation of Spatial Objects

• For Points
  – Use a fixed resolution of the space in both dimensions, i.e., each cell has the same size
  – Each point is then approximated by one cell

• For extended spatial object
  – Minimum enclosing cell
    • Problems with cells that intersect the first partitions already
  – Improvement: use several cells
    • Better approximation of the objects
    • Redundant storage
    • Redundant retrieval in spatial queries

Query returns the same answer several times
Z-Order – Mapping to a B⁺-Tree

• Linear Order for Z-values to store them in a B⁺-tree:
Let \((c_1, l_1)\) and \((c_2, l_2)\) be two Z-Values and let \(l = \min\{l_1, l_2\}\).

The order relation \(\leq_z\) (that defines a linear order on Z-values) is then defined by

\[(c_1, l_1) \leq_z (c_2, l_2) \text{ iff } (c_1 \text{ div } 2^{(l_1 - l)}) \leq (c_2 \text{ div } 2^{(l_2 - l)})\]

Examples:

\[(1,2) \leq_z (3,2),\]
\[(3,4) \leq_z (3,2),\]
\[(1,2) \leq_z (10,4)\]
Mapping to a B⁺-Tree - Example

Exact representations stored in a different location
Mapping to a B+-Tree – Window Query

- Window Query $\rightarrow$ Range Query in the B+-tree
  - find all entries (Z-Values) in the range $[l, u]$ where
    - $l$ = smallest Z-Value of the window (bottom left corner)
    - $u$ = largest Z-Value of the window (top right corner)
    - $l$ and $u$ are computed with respect to the maximum resolution/length of the Z-values in the tree (here: 6)

Result: (0,2)
Spatial Data Management

- Shortcomings of Relational Databases
- Modeling Spatial Data
- Spatial Queries
- Space-Filling Curves + B-Trees
- R-trees
The R-Tree – Properties

• Balanced Tree designed to organize rectangles [Gut 84].
• Each page contains between \( m \) and \( M \) entries.
• Data page entries are of the form \((\text{MBR}, \text{PointerToExactRepr})\).
  – MBR is a minimum bounding rectangle of a spatial object, which PointerToExactRepr is pointing to.
• Directory page entries are of the form \((\text{MBR}, \text{PointerToSubtree})\).
  – MBR is the minimum bounding rectangle of all entries in the subtree, which PointerToSubtree is pointing to.
• Rectangles can overlap.
• The height \( h \) of an R-Tree for \( N \) spatial objects:
  \[
h \leq \left\lceil \log_m N \right\rceil + 1
\]
The R-Tree – Queries

Point Query

Answer Set: [A2, A3]

Window Query

Answer Set: [A2, A3]
The R-Tree – Queries

PointQuery (Page, Point);
FOR ALL Entry ∈ Page DO
    IF Point IN Entry.MBR THEN
        IF Page = DataPage THEN
            PointInPolygonTest (load(Entry.ExactRepr), Point)
        ELSE
            PointQuery (Entry.Subtree, Point);
    WINDOW Query (Page, Window);
FOR ALL Entry ∈ Page DO
    IF Window INTERSECTS Entry.MBR THEN
        IF Page = DataPage THEN
            Intersection (load(Entry.ExactRepr), Window)
        ELSE
            WindowQuery (Entry.Subtree, Window);
    First call: Page = Root of the R-tree
R-Tree Construction – Optimization Goals

- Overlap between the MBRs
  - spatial queries have to follow several paths
  - try to minimize overlap

- Empty space in MBR
  - spatial queries may have to follow irrelevant paths
  - try to minimize area and empty space in MBRs

Start: empty data page (= root)

Insert: A5, A1, A3, A4

\[ M = 3, \ m = 2 \]

[Diagram showing the insertion process with overlapping and empty space issues]
R-Tree Construction – Important Issues

• Split Strategy

How to divide a set of rectangles into 2 sets?

• Insertion Strategy

Where to insert a new rectangle?
R-Tree Construction – Insertion Strategies

• Dynamic construction by insertion of rectangles $R$
  – Searching for the data page into which $R$ will be inserted, traverses the tree from the root to a data page.
  – When considering entries of a directory page $P$, 3 cases can occur:
    1. $R$ falls into exactly one $Entry.MBR$
       \[ \Rightarrow \text{follow } Entry.Subtree \]
    2. $R$ falls into the MBR of more than one entry $e_1, \ldots, e_n$
       \[ \Rightarrow \text{follow } E_i.Subtree \text{ for entry } e_i \text{ with the smallest area of } e_i.MBR. \]
    3. $R$ does not fall into an $Entry.MBR$ of the current page
       \[ \Rightarrow \text{check the increase in area of the } MBR \text{ for each entry when enlarging the } MBR \text{ to enclose } R. \text{ Choose } Entry \text{ with the minimum increase in area (if this entry is not unique, choose the one with the smallest area); enlarge } Entry.MBR \text{ and follow } Entry.Subtree \]

• Construction by “bulk-loading” the rectangles
  – Sort the rectangles, e.g., using Z-Order
  – Create the R-tree “bottom-up”
R-Tree Construction – Split

• Insertion will eventually lead to an overflow of a data page
  – The parent entry for that page is deleted.
  – The page is split into 2 new pages - according to a split strategy
  – 2 new entries pointing to the newly created pages are inserted into the parent page.
  – A now possible overflow in the parent page is handled recursively in a similar way; if the root has to be split, a new root is created to contain the entries pointing to the newly created pages.

\[ M = 3, \ m = 2 \]
R-Tree Construction – Splitting Strategies

• Overflow of node $K$ with $|K| = M+1$ entries $\rightarrow$ Distribution of the entries into two new nodes $K_1$ and $K_2$ such that $|K_1| \geq m$ and $|K_2| \geq m$

• **Exhaustive algorithm:**
  - Searching for the “best” split in the set of all possible splits is too expensive ($O(2^M)$ possibilities!)

• **Quadratic algorithm:**
  - Choose the pair of rectangles $R_1$ and $R_2$ that have the largest value $d(R_1, R_2)$ for empty space in an MBR, which covers both $R_1$ und $R_2$.
    
    $$d(R_1, R_2) := \text{Area}(\text{MBR}(R_1 \cup R_2)) - (\text{Area}(R_1) + \text{Area}(R_2))$$
  - Set $K_1 := \{R_1\}$ and $K_2 := \{R_2\}$
  - Repeat until STOP
    - if all $R_i$ are assigned: STOP
    - if all remaining $R_i$ are needed to fill the smaller node to guarantee minimal occupancy $m$: assign them to the smaller node and STOP
    - else: choose the next $R_i$ and assign it to the node that will have the smallest increase in area of the MBR by the assignment. If not unique: choose the $K_i$ that covers the smaller area (if still not unique: the one with less entries).
R-Tree Construction – Splitting Strategies

- **Linear algorithm:**
  - Same as the quadratic algorithm, except for the choice of the initial pair: Choose the pair with the largest normalized distance.
  - For each dimension determine the rectangle with the largest minimal value and the rectangle with the smallest maximal value (the difference is the **maximal distance/separation**).
  - Normalize the maximal distance of each dimension by dividing by the sum of the extensions of the rectangles in this dimension
  - Choose the pair of rectangles that has the greatest normalized distance.
  Set \( K_1 := \{ R_1 \} \) and \( K_2 := \{ R_2 \} \).
R-Trees – Variants

- Many variants of R-trees exist,
  - e.g., the R*-tree, X-tree for higher dimensional point data, …
  - For further information see http://www.cs.umd.edu/~hjs/rtrees/index.html (includes an interactive demo)

- R-trees are also efficient index structures for point data since points can be modeled as “degenerated” rectangles
  - Multi-dimensional points, where a distance function between the points is defined play an important role for similarity search in so-called “feature” or “multi-media” databases.
Examples of Feature Databases

- Measurements for celestial objects (e.g., intensity of emission in different wavelengths)

- Colour histograms of images

- Documents, shape descriptors, …

\[ (o_{11}, o_{12}, \ldots, o_{1d}) \]
\[ (o_{21}, o_{22}, \ldots, o_{2d}) \]
\[ \ldots \]
\[ (o_{n1}, o_{n2}, \ldots, o_{nd}) \]
Feature Databases and Similarity Queries

- **Objects + Metric Distance Function**
  - The distance function measures (dis)similarity between objects

- **Basic types of similarity queries**
  - **range queries with range** $\varepsilon$
    - Retrieves all objects which are similar to the query object up to a certain degree $\varepsilon$
  - **$k$-nearest neighbor queries**
    - Retrieves $k$ most similar objects to the query

![Diagram](https://via.placeholder.com/150)