15-823
Advanced Topics in Database Systems Performance

Spatial Application

Tiankai Tu
Computer Science Department
tutk@cs.cmu.edu

Questions

- What is spatial data ?
- How to represent spatial data?
- What are the queries?
- How to process the query?

Spatial Data:Definition

- Data that pertains to the space occupied by objects
- Conceptually, points, lines, rectangles, surfaces, volumes and etc.
- Physically, cities, rivers, roads, states, crop coverage, mountain ranges etc.
- Applications include environmental monitoring, geographic information systems, earthquake research etc.

Spatial Data:Features

- Geometric and varied
- Naturally high dimensional
- Can be either discrete or continuous
- May associate with *non-spatial attributes*

Spatial Data and DBMS

- How: turn spatial data into tuples
  - Termed *representative point*
  - Parameterized representation
  - Non-spatial attributes stored together
  - It works for simple retrieval and queries
- Why not: unnatural
  - Dimension of representative point too high
  - Deducing dimensionality lose important physical information such as proximity
  - Clumsy for queries involving space

A Motivating Example

- Pittsburgh road database
  - Suppose the roads are straight lines (forget about Forbes for this example)
  - A representative point consists of two endpoints, that is, a tuple of four four items
  - Mapping from a two-dimensional (*drawing*) space to a four-dimensional (*transformed*) space
- Queries
  - What are the roads originating from Point State Park?
  - Which road is the closest to Wean Hall?
  - Which roads go through CMU?

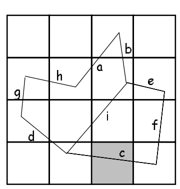
Spatial Data Representation

- Follow the natural way: spatial occupancy
- Decompose the space from which the data is drawn into *buckets*
- Four principal approaches
  - Minimum bounding rectangles: data-dependent
    - e.g. R-tree [Guttman 1984], R^*-tree [Beckmann et al. 1990]
  - Disjoint cells: data-dependent
    - e.g. R^+-tree [Sellis et al. 1987], Cell tree [Günther 1988]
  - Uniform size blocks: data-independent
    - e.g. Uniform grid [Franklin 1984]
  - Adaptive regular blocks: data-independent
    - e.g. Quadtree-based approach [Same et al. 1985]

Spatial Indexing

- All the spatial occupancy methods are characterized as employing *spatial indexing*
- Each block/cell/rectangle only contains information about whether or not it's occupied by the object or part of the object
- The information is usually in the form of a pointer to a descriptor of the object

Spatial Indexing (cont.)



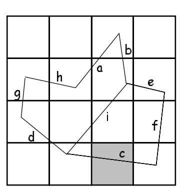
Example collection of line segments embedded in a 4x4 grid

- The shaded block only records the fact line segment **c** crosses it
- The part of the line that passes through or terminates in a block is termed *q-edge*
- Each q-edge in the block is represented by a pointer to a record containing the end points of the line segment
- No information about what part the line crosses it

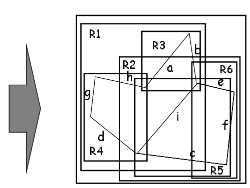
Case Study:R-Tree

- Goal: efficient retrieval of objects according to their spatial location
- Conventional DBMS indexing structure
  - Hashing: cannot handle range query
  - B-tree: cannot handle multi-dimensional data
- Key ideas:
  - Maintain balanced hierarchical tree structure
  - Represent objects by intervals in several dimensions
  - Take care of secondary memory paging

R-Tree:Example

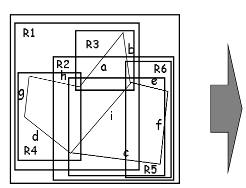


Example collection of line segments embedded in a 4x4 grid

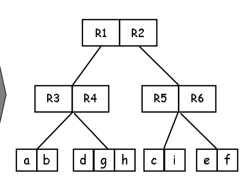


Spatial extents of the bounding rectangles for R-tree

R-Tree:Example (cont.)



Spatial extents of the bounding rectangles for R-tree



R-tree representation of line segments

R-Tree:Properties

- Height-balanced tree similar to B-tree
  - All leaves appear on the same level
- Leaf nodes contain index record **entries** of the form *(I,tuple-identifier)*
  - *tuple-identifier*: a tuple in database
  - *I :* an n-dimensional rectangle that bounds the spatial object indexed
- Non-leaf nodes contain **entries** of the form *(I,child-pointers)*
  - *child-pointer:* the address of a lower node in the R-tree
  - *I* : bounds all rectangles in the lower node's entries

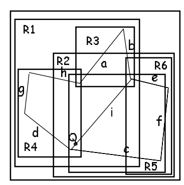
R-Tree:Properties (cont.)

- Every node contains between *m* and *M* entries unless it is the root
  - *M* : maximum number of entries that fit in one node.
  - *m:* minimum number of entries in a node ($m \leq M/2$)
  - Nodes correspond to disk pages
- Root node has at least two children unless it is a leaf

R-Tree:Search

- Search Algorithm
  - Given an R-tree rooted at *T*, find all records whose rectangles overlap search rectangle *S*
  - Denote the rectangle part of an index entry *E* by *EI*, and the *tuple-identifier* or *child-pointers* by *Ep*
  - Search subtree: If *T* is not a leaf, check each entry E to determine whether *EI* overlaps *S*; For all overlapping entries, invoke Search subtree on the tree rooted at *Ep*
  - Search leaf node: If *T* is a leaf, check all entries *E* to determine whether *EI* overlaps *S*. If so, E is a qualifying record

R-Tree:Search (cont.)



Spatial extents of the bounding rectangles for R-tree

- Acclaimed strength
  - Eliminate irrelevant regions of the indexed space and examine only data near the search area, thus visit only a small number of nodes
- Criticized drawback:
  - More than one subtree under a node may be searched, thus potential search the entire spatial database
  - Bad example:find the line segment passing through Q

R-Tree:Insert

- Similar to insertion in B-tree
- New index records are added to the leaves, nodes overflow are split, and split propagate up the tree
- Key issues:
  - Choose leaf node for the first insertion
  - Adjust nodes as changes propagate up
  - Split node if it becomes too full (> *M*)

R-Tree:Insert (cont.)

- How to choose leaf node
  - Descend the tree
  - At each non-leaf node, choose the entry that needs least enlargement to include the new index entry; follow the *child-pointer* link of this entry to find next level non-leaf node
- How to adjust parent nodes
  - Enlarge *I* of the entry in the parent node still tightly encloses all entry rectangles in child node

R-Tree:Insert (cont.)

- How to split full node
  - Objective: reduce the number of node to be examined on subsequent searches
  - Principle: the total area of the two covering rectangles after a split should be minimized
  - Exhaustive algorithm
    - Try all possible grouping and choose the best
    - Prohibitively slow
  - Quadratic split algorithm: eager strategy
  - Linear algorithm: fast and almost as good as more expensive ones as per experiments

R-Tree:Deletion

- Different from deletion in B-tree in how to handle *under-full* (< *m*) nodes
- Deleted index may result in shrinking of containing rectangle at higher levels and nodes may become under-full
- Key issues:
  - Shrink the rectangles all the way up
  - Handle under-full nodes

R-Tree:Deletion (cont.)

- How to shrink rectangle
  - Shrink to tightly contain all the entry rectangles in the child node
- How to handle under-full node
  - Not merged with sibling
  - Removed aside till the end and then entries are reinserted into the R-tree
    - Functionally equivalent with improve disk cache hit
    - Incrementally refine the spatial structure and prevent gradual deterioration due to permanently fixed parent-children relationship

R-Tree:Update

- Update is simple
  - Delete original index from R-tree
  - Update the index
  - Reinsert the index into the R-tree
- Other search operations are simply variants of the one described

R-Tree:Wrap-up

- Exploit strength of B-tree and geometry of underlying objects
- Dynamically update index
- Use heuristic optimization(linear algorithm) to bound overhead
- Many variants have been proposed to improve R-tree
- Hard to model continuous geo-spatial data such as a field (personal opinion)

Conclusion

- Spatial application is important, esp. in science and engineering research
- Conventional database technique failed to capture the essential nature of spatial application
- New methods and algorithms have been proposed, each trying to solve part of the problem
- No cure-all solution is perceived, which results in very active research in this area