Summary of Spatial access methods

By S. Xiao Fernández Marín

1 Summary

Spatial data
• 2 or 3 dimensions
• Points
• Linestrings
• Polygons
– With and without holes
• Collections
– Multipoints
– Multilinestrings
– Multipolygons
– Collections with mixed types

Spatial functions
• Spatial relations (boolean)
– Within, overlaps, intersects, contains, covers, covered by, equals, etc.
– Typical use cases for spatial indexes
• Set operations
– Intersection, union, difference, symmetric difference
• Geometric properties
– Distance, length, area, angle measures, buffer, etc.

Spatial indexes
• Point queries; ST_GeomFromText('POINT(0 0)'); LINESTRING(0 0, 10 10)
– Which geometries contain the query point?
• Window queries; polygon((0.5 0.5, 1.5 0.5, 1.5 1.5, 0.5 0.5))
– Which geometries are contained in (or overlap, etc.) the query geometry?
• Bounding boxes
– Also known as minimal bounding box (MBB), minimal bounding rectangle (MBR)
– Box sides are parallel to axes
• Indexes are often based on bounding boxes
– Faster math
• Index lookup returns superset of actual result
– Must filter through exact shape to get actual result
– Still much faster than table scan
• Not necessarily a total order of keys
– Not always possible to keep geometrically close objects close in the index structure
• Index construction
– Static (create index and use that) or dynamic (inserting and deleting stuff)-> Self-balancing for dynamic
– Packing
• Search operations
– Point queries
– Window queries
– kNN and other more advanced queries

• Time complexity
– Sublinear point queries
– Sublinear window queries

• Space complexity
– Size comparable to indexed data

• Space-driven (más ordenado,, todo a la izq)
– Pre-define a division of the data domain into partitions
– Regular
– Less adaptive to data set
• Data-driven (junto toda la data)
– Divide the data set
– Adaptive to data set
– More irregular divisions

Grid files No cells splits, point P falls in a page not full ; cell split and no dir split, point P falls in a full page but referenced (no need to split as already done) ; and cell split and dir split, dplit grid and new page

Areal geometries
Grids
• fixed grid: duplication in neighbor cells = cell split likely to occur more often
• grid files: more dynamic

• Relatively simple
– Predefined split points
• Not fully adaptive to data set
• Requires pre-defined range boundaries
• Duplication of areal geometries increases with amount of data
• Performance relies on having the dictionary in memory
– Not feasible for large data sets

Quad-trees
• Split the data space into four quadrants
– NW, NE, SW, SE
• When a quadrant is full, split it into four quadrants
• Simple
• Main-memory structure
• Low fan-out
– Deep trees
• Duplication of entries!!!

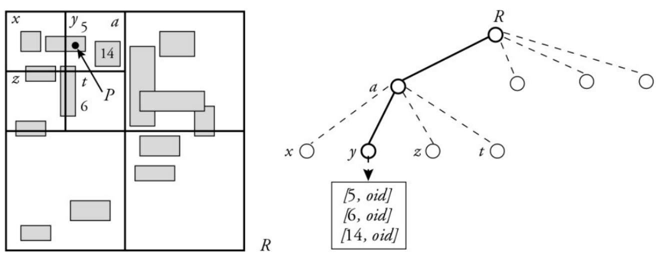

Figure 1: Quad-Tree

Space filling curves
• Define a total order of grid cells

– Linearizing the data space
• Store index in a B-tree
• Reuse of existing 1d index methods
• Loss of clustering
– Less efficient window queries
• Duplication of entries

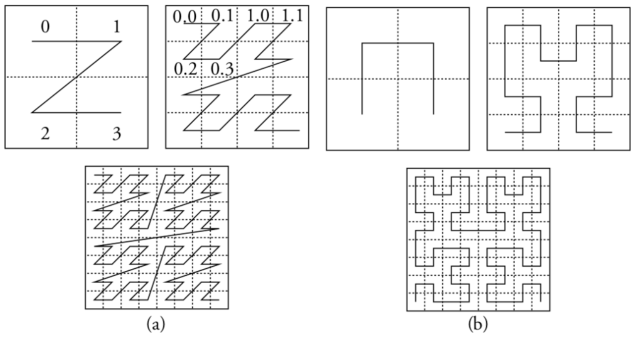

Figure 6.13 Space-filling curves: z-ordering (a) and Hilbert curve (b).

Figure 2: Space filling curves

Linearized quad-tree Far in the tree, close in the grid file

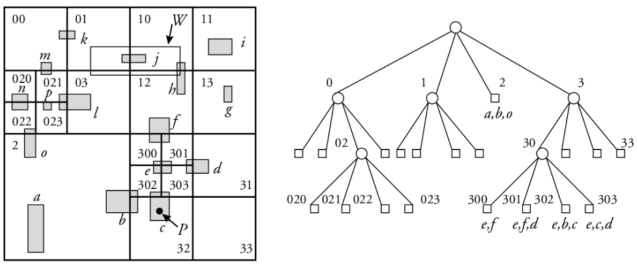

Figure 6.14 A quadtree labeled with z-order.

Figure 3: Linearized quad-tree

Decomposition If there is no need of smaller boxes, don't. For avoiding duplication and visit few nodes. Cons -> strings not same length

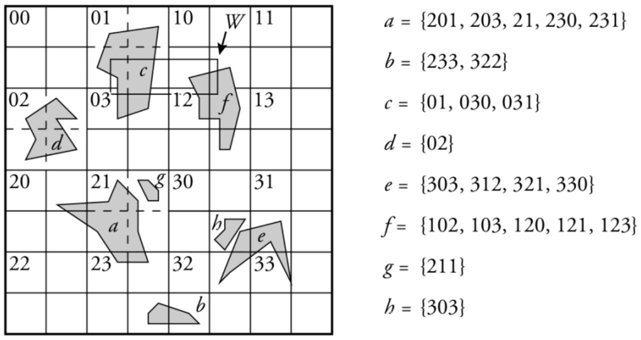

Figure 6.17 A set of objects with z-ordering decomposition.

Figure 4: Decomposition

R-trees

- Data-driven

– Adaptive to data set
- Balanced tree

– All leaves are at the same level
- Leaf nodes contain bounding box of single data entry
- Intermediate nodes contain bounding box of whole subtree
- Root node contains bounding box of entire data set
- Cons->Sibling nodes may overlap
- Cons->Tree structure depends on insertion order

Each bounding box in each leave but if want to check 12,, check nodes c and d bc they are the ones implicated

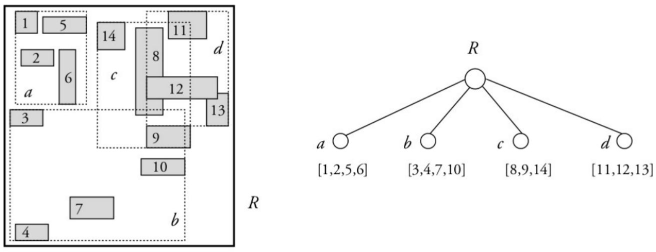

Figure 6.22 An R-tree.

Figure 5: R-trees

Inserting into R-trees

- Traverse the tree through nodes which bounding boxes contain that of the new entry

– If the new entry's bounding box is not contained by any nodes at a certain level, pick the one that will be expanded the least
- When reaching a leaf node, insert the entry

– If the leaf is full, split it: (a) Minimize the total area of the two nodes, (b) Minimize the overlapping of the two nodes

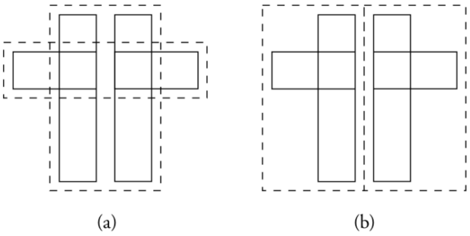

Figure 6.27 Minimal area and minimal overlap: a split with minimal area (a) and a split with minimal overlap (b).

Figure 6: Inserting

R*trees
• Improvement on the original R-tree
– Node overlapping
– Area covered by a node
– Perimeter of a bounding box: Given a certain area, the square has the minimal bounding box perimeter
• Aims at improving node splitting

R+trees
• Sibling bounding boxes don't overlap
– Single path from root to leaf for point queries
• Duplication of areal entries

R-tree packing