

CMSC 724: Indexes

Amol Deshpande

University of Maryland, College Park

February 20, 2007

*Adapted from Joe Hellerstein's Notes (<http://redbook.cs.berkeley.edu/redbook3/lec4.html>)

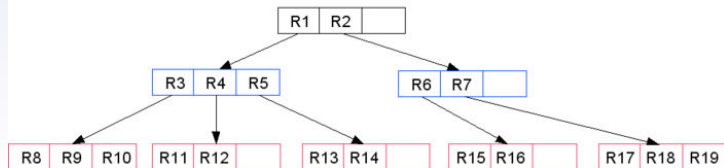
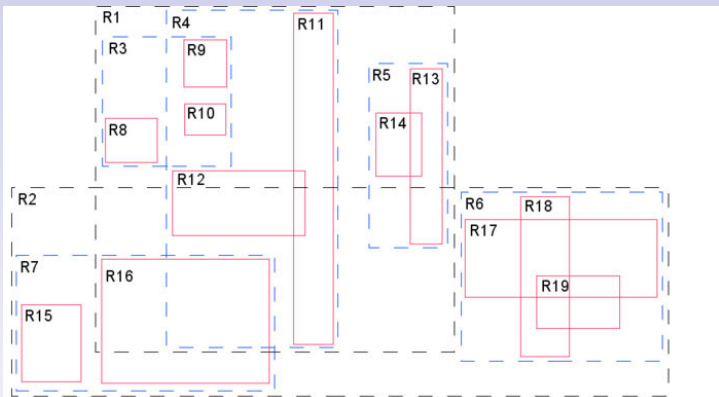
Access Methods

- Heap files and indexes
- Support iterator interface:
 - open (possibly with selection condition)
 - `get_next`, `close`, `insert`, `delete`, `update_field`
- At a high level:
 - *partition*: partition a dataset or domain into buckets
 - *label*: provide a label for each bucket
- Performance goals:
 - Query, insert, delete times (or disk I/Os)
 - Compare to sequential (seek times improving much slower)
- Implementation Issues:
 - Concurrency & recovery
 - Cost estimation
 - Bulk loading

B+-Tree

- Balanced, 50% utilization (in practice, allow getting lower)
- $O(\log_d(n))$ search, update, delete costs
- Speeding up counts; key compression; bulk loading
- Concurrency: not 2PL - too slow
- Optimal for one-dimensional data
- B-Trees ?
- Multi-dimensional data
($20 < \textit{age} < 30 \wedge \$100,000 < \textit{salary}$)
 - Grid-files, Quad-trees etc. . .
 - Space-filling curves. . .

R-Tree



R-Tree

- Multi-dimensional, spatial data (points, rectangles)
- Queries: point in polygon, polygon in polygon, overlaps polygon, contains polygon
- *labels*: bounding rectangles
- Bulk loading ? Hard...
- Search: Follow all paths.
- Insert: Driven by minimizing *area enlargement*
- Split algorithms: exhaustive, quadratic, linear
- Delete: re-insert if too small (why ?)

R*-Tree

- Analysis: four optimization metrics ?
 - Minimize area covered by a directory rectangle.
 - Minimize overlap; Minimize margin; Maximize storage *utilization*
- Conflict with each other
 - E.g., minimizing area covered conflicts with maximizing storage utilization.
- Should really be driven by a query workload
- Changes:
 - Insertion algorithm slightly different (minimizes “overlap” at leaf level)
 - Aggressive re-insertion (30% entries re-inserted at the same level)
- Lots of heuristics. . . backed by experimental analysis. . .

More...

- Space partitioning vs data partitioning
- R+-Tree: Uses replication (non-overlapping keys)
- Quadtrees, K-D-B Trees
- Trees for nearest-neighbor or similarity searches
- Interval trees (several optimality results exist)
- GiST: Generalized Search Trees (will cover later)