

R-Trees: A Dynamic Index Structure for Spatial Searching

Presenter: Nahi Ojeil

10/15/09

EECS 584, Fall 2009

1

Outline

1. Motivation
2. Problem Definition
3. R-Tree Description
4. Operations on R-Trees
5. Results
6. Conclusion

10/15/09

EECS 584, Fall 2009

2

Motivation

- Need to handle spatial data efficiently:
 - Multi-dimensional
 - Range search
- Applications:
 - computer aided design (CAD)
 - geo-data applications
- Traditional indexing methods based on point locations.

10/15/09

EECS 584, Fall 2009

3

Motivation

- Alternatives:
 - B Tree is a generalized binary tree.
 - B+ Tree is a B-Tree with values in leaf nodes only.
 - Both use total ordering on elements.
 - How do we order: (0,0), (0,1), (1,0), (1,1)?
 - How about the ranges [0:3], [2:3], [1:4]?
 - How to search for [2.25:2.75]?

10/15/09

EECS 584, Fall 2009

4

Problem Definition

- Need a new index structure:
 - for multi-dimensional objects
 - with efficient range searches

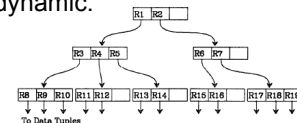
10/15/09

EECS 584, Fall 2009

5

R-Tree Description

- Height balanced tree.
- Pointers to index records in leaf nodes.
- Index is dynamic.



10/15/09

EECS 584, Fall 2009

6

R-Tree Description

- Leaf nodes (I , tuple-identifier)
 - I bounding box of form $(I_0, I_1, \dots, I_{n-1})$
 - n is number of dimensions
 - I_i is a closed bounded interval $[a_i, b_i]$
- Non-leaf nodes (I , child-pointer)
 - child pointer address of lower node
 - M max child nodes per non-leaf node
 - $m \leq M/2$ min child nodes per non-leaf node

10/15/09

EECS 584, Fall 2009

7

R-Tree Description (Properties)

1. Every leaf node contains between m and M index records, except root.
2. For each (I , *tuple-identifier*), I is smallest rectangle that spatially contains the n -dimensional data object represented by this *tuple-identifier*.
3. Every non-leaf node has between m and M children unless it is the root.

10/15/09

EECS 584, Fall 2009

8

R-Tree Description (Properties)

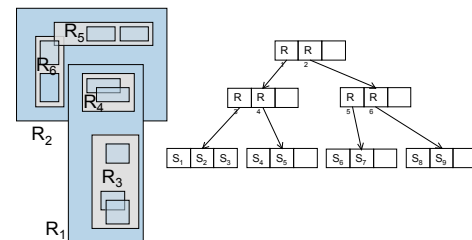
4. For each (I , *child-pointer*), I is smallest rectangle that spatially contains the rectangles in the child node.
5. The root has at least 2 children unless it is a leaf.
6. All leaves appear on the same level.

10/15/09

EECS 584, Fall 2009

9

R-Tree Description



10/15/09

EECS 584, Fall 2009

10

Searching

- S1 [Search subtrees]
 - If T is not a leaf, check each entry E to determine whether E_i overlaps S . For all overlapping entries, invoke Search on the tree whose root node is pointed to by E_i .
- S2 [Search leaf node]
 - If T is a leaf, check all entries E to determine whether E_i overlaps S . If so, E is a qualifying record.

10/15/09

EECS 584, Fall 2009

11

Searching

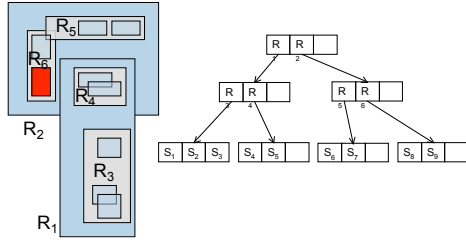
- Issues:
 - Search either intersection, or containment.
 - Might follow multiple paths down the tree.
 - Need to examine all children of every non-leaf node we encounter.
- Worst Case Complexity:
 - Time: $O(N)$
 - Space: $O(N)$

10/15/09

EECS 584, Fall 2009

12

R-Tree Description

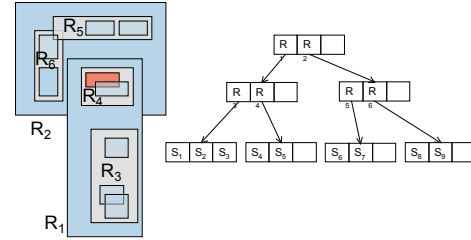


10/15/09

EECS 584, Fall 2009

13

R-Tree Description



10/15/09

EECS 584, Fall 2009

14

Insertion

- I1 [Find position for new record]
 - Select a leaf node L in which to place E.
- I2 [Add record to leaf node]
 - If no room for entry, split node.
- I3 [Propagate changes upward]
 - Only if a split occurs.
- I4 [Grow tree taller]
 - Only if propagation caused root split.

10/15/09

EECS 584, Fall 2009

15

Insertion

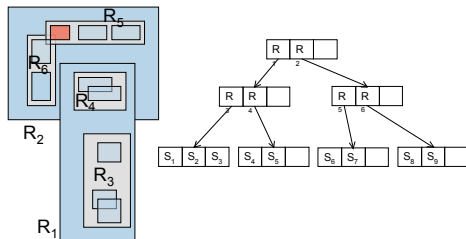
- Issues:
 - Selecting leaf node in I1 depends on least amount of area increase.
 - Split node if node has $M + 1$ children.
 - Propagation takes $\text{Height}(\text{R-Tree})$ steps.
 - Root splits very rare.
- Worst Case Complexity:
 - Time: $O(m \log_m N)$

10/15/09

EECS 584, Fall 2009

16

R-Tree Description

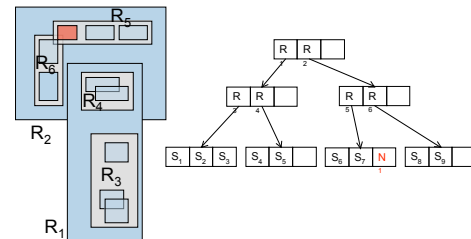


10/15/09

EECS 584, Fall 2009

17

R-Tree Description

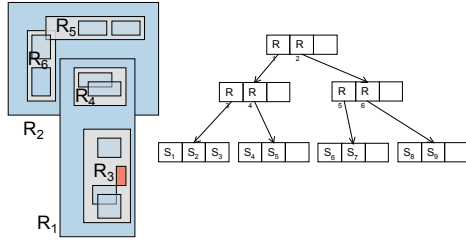


10/15/09

EECS 584, Fall 2009

18

R-Tree Description

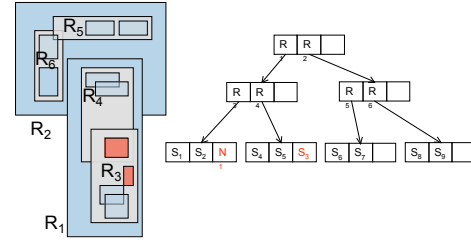


10/15/09

EECS 584, Fall 2009

19

R-Tree Description



10/15/09

EECS 584, Fall 2009

20

Deletion

- D1 [Find node containing record]
 - Locate the leaf node L containing E.
- D2 [Delete record]
 - Remove E from L.
- D3 [Propagate changes]
 - Condense the tree whenever possible.
- D4 [Shorten tree]
 - If root node has one child left, make it root.

10/15/09

EECS 584, Fall 2009

21

Deletion

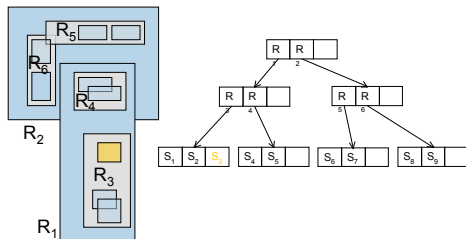
- Issues:
 - If can't find entry E, abort.
 - Propagation happens if node has less than m entries.
- Worst Case Complexity:
 - Time: $O(N)$
 - Space: $O(N)$ due to search

10/15/09

EECS 584, Fall 2009

22

R-Tree Description

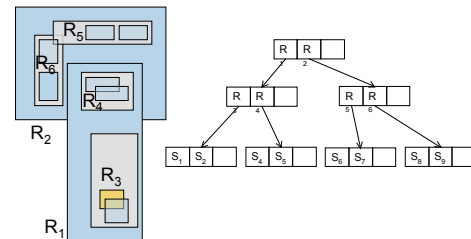


10/15/09

EECS 584, Fall 2009

23

R-Tree Description

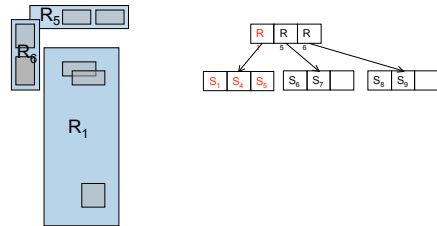


10/15/09

EECS 584, Fall 2009

24

R-Tree Description



10/15/09

EECS 584, Fall 2009

25

Update

- Update is equivalent to:
 - Delete its index record from tree.
 - Update its index record.
 - Insert back into tree.

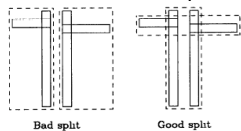
10/15/09

EECS 584, Fall 2009

26

Node Splitting

- Exhaustive Algorithm (Optimal)
 - Examine all possible splits and choose the one with smallest area.
 - Exponential number of possible splits.



10/15/09

EECS 584, Fall 2009

27

Node Splitting

- Quadratic-Cost Algorithm (Greedy non-optimal)
 - Pick two rectangles that cause most area waste.
 - Put them in different splits.
 - Pick each remaining entry and insert in split with least area addition.

10/15/09

EECS 584, Fall 2009

28

Node Splitting

- Linear-Cost Algorithm (Greedy non-optimal)
 - Uses heuristic to choose the first rectangle in each split.
 - Pick each remaining entry and insert in split with least area addition.

10/15/09

EECS 584, Fall 2009

29


Empirical Results

- Evaluate by varying:
 - Different bytes per page.
 - $m=2$, $m=M/3$, $m=M/2$
 - Node Splitting Algorithm: E, Q, L
 - Number of records in index.

10/15/09

EECS 584, Fall 2009


30



Empirical Results

- Check for:
 - CPU time per operation.
 - Pages touched per record.
 - Storage space required.


10/15/09 EECS 584, Fall 2009 31



Empirical Results

- Findings:
 - L node splitting has min cost on insert.
 - Delete time affected by under-full nodes.
 - Search performance almost the same.
 - Storage space better for higher m.
 - Search time and space same for L and Q as records increase.


10/15/09 EECS 584, Fall 2009 32



Conclusion

- R-Tree structure useful for spatial data.
- Linear node split algorithm fast and does not affect search performance noticeably.
- R-Trees should be easy to add to conventional relational database systems.

10/15/09 EECS 584, Fall 2009 33



Questions?

10/15/09 EECS 584, Fall 2009 34