R-trees

COSC 404 – Database System Implementation

R-Trees Introduction

R-trees (or region tree) is a generalized B-tree suitable for processing spatial queries. Unlike B-trees where the keys have only one dimension, R-trees can handle multidimensional data.

The basic R-tree was proposed by Guttman in 1984 and extensions and modifications have been later developed.

- R+-tree (Sellis *et al.* 1987)
- R*-tree (Beckmann *et al.* 1990)

We begin by looking at the properties of spatial data and spatial query processing.

Types of Spatial Data

Spatial data includes multidimensional points, lines, rectangles, and other geometric objects.

A spatial data object occupies a region of space, called its spatial extent, which is defined by its location and boundary.

Point Data - points in multidimensional space

Region Data - objects occupy a region (spatial extent) with a location and a boundary.

Types of Spatial Queries

Spatial Range Queries - query has associated region and asks to find matches within that region

- e.g. Find all cities within 50 miles of Kelowna.
- Answer to query may include *overlapping* or *contained* regions.

Nearest Neighbor Queries - find closest region to a region.

- e.g. Find the 5 closest cities to Kelowna.
- Results are ordered by proximity (distance from given region).

Spatial Join Queries - join two types of regions

- e.g. Find all cities near a lake.
- Expensive to compute as join condition involves regions and proximity.

Spatial Data Applications

Geographic Information Systems (GIS) use spatial data for modeling cities, roads, buildings, and terrain.

Computer-aided design and manufacturing (CAD/CAM) process spatial objects when designing systems.

- Spatial constraints: "There must be at least 6 inches between the light switch and turn signal."

Multimedia databases storing images, text, and video require spatial data management to answer queries like "Return the images similar to this one." Involves use of feature vectors.

- Similarity query converted into nearest neighbor query.

Spatial Queries

Question: What type of spatial query is: "Find the city closest to Chicago?"

A) Spatial Range Query

B) Nearest Neighbor Query

C) Spatial Join Query

D) Not a spatial query

Spatial Indexing

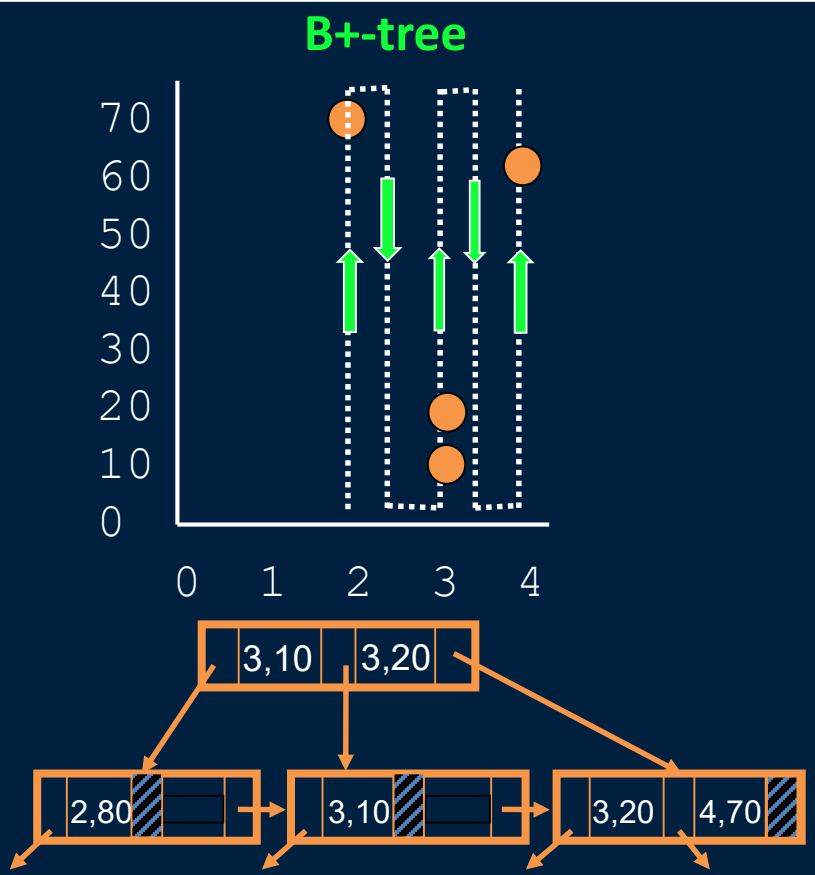
A multidimensional or spatial index utilizes some kind of spatial relationship to organize data entries. Each key value in the index is a point (or region) in *k*-dimensional space, where *k* is the number of fields in the search key.

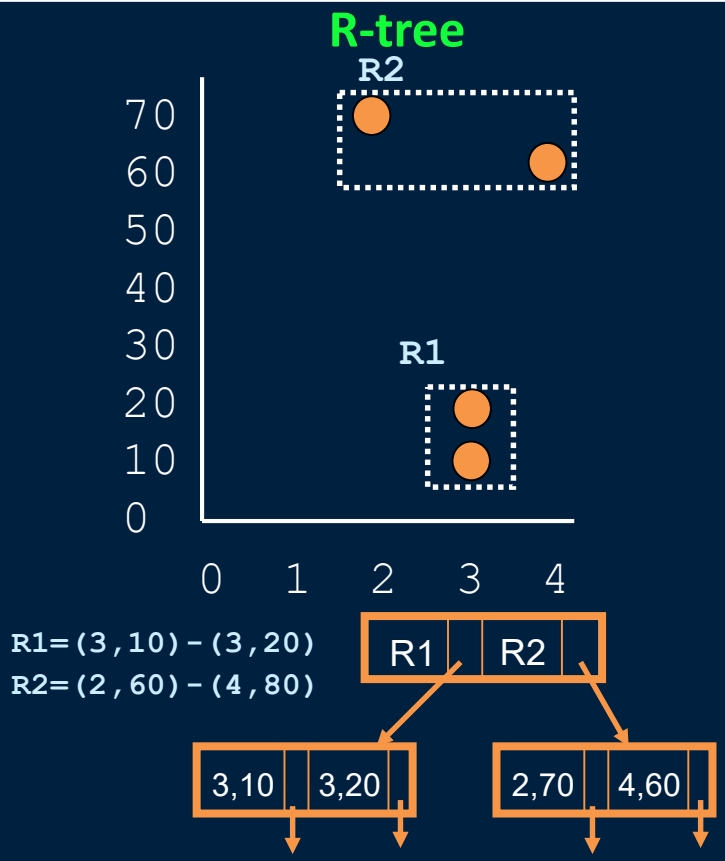
Although multidimensions (multiple key fields) can be handled in a B+-tree, this is accomplished by imposing a total ordering on the data as B+-trees are single-dimensional indexes.

For instance, B+-tree index on <*x*,*y*> would sort the points by *x* then by *y*.

- I.e. <2,70>, <3,10>, <3,20>, <4,60>

B+-tree versus R-tree





B+-tree versus R-tree Querying

Consider these three queries on *x* and *y*:

- 1) Return all points with $x < 3$.
  - Works well on B+-tree and R-tree. Most efficient on B+-tree.
- 2) Return all points with $y < 50$.
  - Cannot be efficiently processed with B+-tree as data sorted on *x* first.
  - Can be efficiently processed on R+-tree.
- 3) Return all points with $x < 3$ and $y < 50$.
  - B+-tree is only useful for selection on *x*. Not very good if many points satisfy this criteria.
  - Efficient for R-tree as only search regions that may contain points that satisfy both criteria.

R-Tree Structure

R-tree is adaptation of B+-tree to handle spatial data.

The search key for an R tree is a collection of intervals with one interval per dimension. Search keys are referred to as ***bounding boxes*** or ***minimum bounding rectangles*** (***MBR***s).

- Example:

Each entry in a node consists of a pair <*n-dimensional box*, *id*> where the *id* identifies the object and the *box* is its MBR.

Data entries are stored in leaf nodes and non-leaf nodes contain entries consisting of <*n-dimensional box*, *node pointer*>.

The box at a non-leaf node is the smallest box that contains all the boxes associated with the child nodes.

R-Tree Notes

The bounding box for two children of a given node can overlap.

- Thus, more than one leaf node could potentially store a given data region.

A data point (region) is only stored in one leaf node.

R-Tree Searching

Start at the root.

- If current node is non-leaf, for each entry <*E*, *ptr*> if box *E* overlaps *Q*, search subtree identified by *ptr*.
- If current node is leaf, for each entry <*E*, *id*>, if *E* overlaps *Q*, *id* identifies an object that might overlap *Q*.

Note that you may have to search several subtrees at each node. In comparison, a B-tree equality search goes to just one leaf.

R-Tree Searching Improvements

Although it is more convenient to store boxes to represent regions because they can be represented compactly, it is possible to get more precise bounding regions by using convex polygons.

Although testing overlap is more complicated and slower, this is done in main-memory so it can be done quite efficiently. This often leads to an improvement.

R-Tree Insertion Algorithm

Start at root and go down to "best-fit" leaf *L*.

- Go to child whose box needs ***least enlargement*** to cover *B*; resolve ties by going to smallest area child.

If best-fit leaf *L* has space, insert entry and stop.

Otherwise, split *L* into *L1* and *L2*.

- Adjust entry for *L* in its parent so that the box now covers (only) *L1*.
- Add an entry (in the parent node of *L*) for *L2*. (This could cause the parent node to recursively split.)

R-Tree Insertion Algorithm
Splitting a Node

The existing entries in node *L* plus the newly inserted entry must be distributed between *L1* and *L2*.

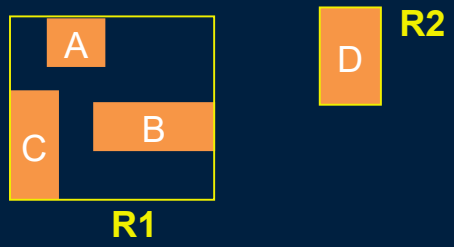
Goal is to reduce likelihood of both *L1* and *L2* being searched on subsequent queries.

Idea: Redistribute so as to minimize area of *L1* plus area of *L2*.

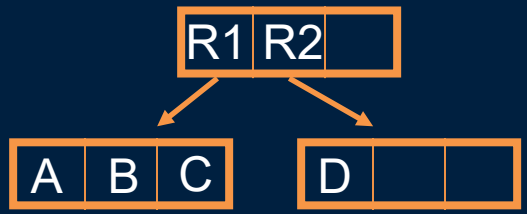
An exhaustive search of possibilities is too slow so quadratic and linear heuristics are used.

Insertion Example

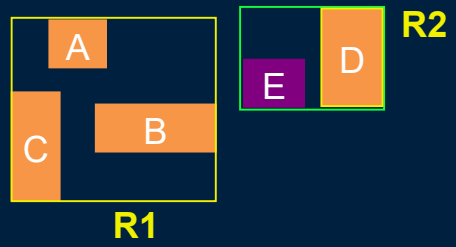
Spatial Data



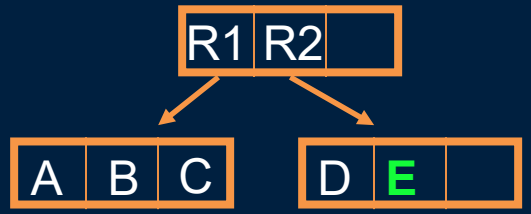
R-tree degree=3



Insert E



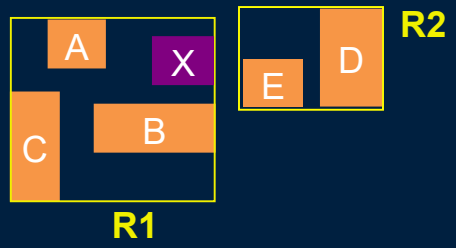
New R-tree



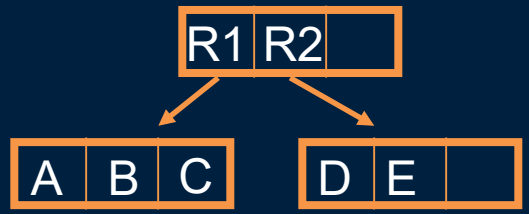
Extended region R2 to hold E.

Insertion Example 2

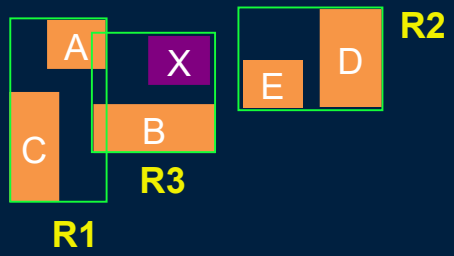
Insert X



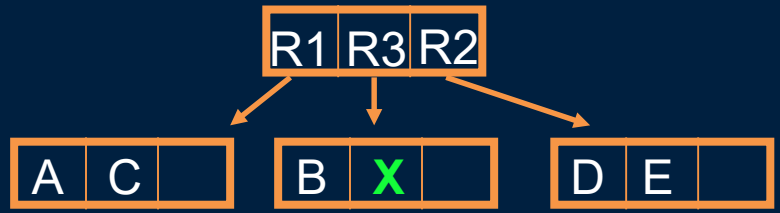
Original R-tree



Updated Regions



New R-tree



Split R1 into R1 and R3.

R+-Tree

R+-tree avoids overlap by inserting an object into multiple leaves if necessary.

Reduces search cost as now take a single path to leaf.

R*-Tree

R*-tree uses the concept of forced reinserts to reduce overlap in tree nodes.

When a node overflows, instead of splitting:

- Remove some (say 30%) of the entries and reinsert them into the tree.
- Could result in all reinserted entries fitting on some existing pages, avoiding a split.

R*-trees also use a different heuristic, minimizing box parameters, rather than box areas during insertion.

GiST

The Generalized Search Tree (GiST) abstracts the "tree" nature of a class of index including B+-trees and R-tree variants.

- Striking similarities in insert/delete/search and even concurrency control algorithms make it possible to provide "templates" for these algorithms that can be customized to obtain the many different tree index structures.
- B+ trees are so important (and simple enough to allow further specialization) that they are implemented specifically in all DBMSs.
- GiST provides an alternative for implementing other index types.
- Implemented in PostgreSQL. Make your own index!

R-Tree Variants

Question: Select a true statement.

A) Searching in a R-tree always follows a single path.

B) R-tree variants may have different ways for splitting nodes during insertion.

C) A R+-tree search always follows a single path to a leaf node.

D) None of the above

R-Trees Summary

An R-tree is useful for indexing and searching spatial data.

Variants of R-trees are used in commercial databases.

Major Objectives

The "One Thing":

- Be able to explain the difference between an R-tree and a B+-tree.

Other objectives:

- List some types of spatial data.
- List some types of spatial queries.
- List some applications of spatial data and queries.
- Understand the idea of insertion in a R-tree.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA