Spatial Indexing

Ramakrishnan/Gehrke Ch. 28

Applications of Multidimensional Data

- Geographic Information Systems (GIS)
  - Geospatial information; service standards by Open Geospatial Consortium (OGC)
  - Vendors: ESRI, Intergraph, SmallWorld, …, Oracle, …; open-source: Grass, PostGIS, …
  - All classes of spatial queries and data are common
- Computer-Aided Design / Manufacturing
  - spatial objects, ex: surface of airplane fuselage
  - Range queries and spatial join queries are common
- Multimedia Databases
  - Images, video, text, etc. stored and retrieved by content
  - First converted to *feature vector* form; high dimensionality
  - Nearest-neighbor queries are the most common

Multidimensional Data

- Point Data
  - = points in a multidimensional space
  - Ex: geographic locations; feature vectors extracted from text
- Region Data
  - = objects having spatial extent with location and boundary
  - typically geometric approximations: polygons etc. → vector data
- What about raster data such as satellite imagery?
  - = each pixel stores a measured value
  - Here vector data; raster data different story

Multidimensional Queries

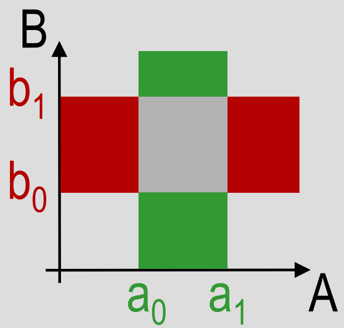
- Point queries
  - *"show Bremen"*
- Spatial Range queries
  - *"Find all hotels within a radius of 5 miles from the conference venue"*
  - *"Find all cities that lie on the Nile in Egypt"*
  - 50 < age < 55 AND 80K < sal < 90K
  - 50 < Lat < 55 AND 80 < Long < 90
- Nearest-Neighbor queries
  - *"Find the 10 cities nearest to Bremen"*
  - *"Find the city with population 500,000 or more that is nearest to Kalamazoo, MI"*
- Spatial Join queries
  - *"Find all cities near a lake"*
  - *"Find all parts that touch the fuselage" (in airplane design)*
  - Expensive; join condition involves regions and proximity!
- Similarity queries
  - "Given a face, find the five most similar faces"
- *…plus aggregation, and more*

Multiple B+ Trees

- Query example:

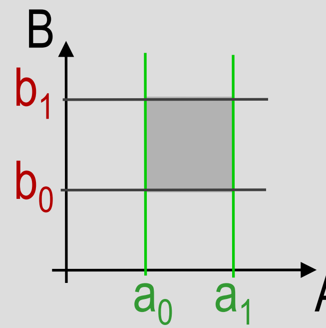
  select * from R where $a_0 < A < a_1$ **and** $b_0 < B < b_1$

Several conventional indexes:



- read tuple with $a_0<A<a_1$
- read tuple with $b_0<B<b_1$
- intersect

wanted:



read only tuples with $a_0<A<a_1$ <u>and</u> $b_0<B<b_1$

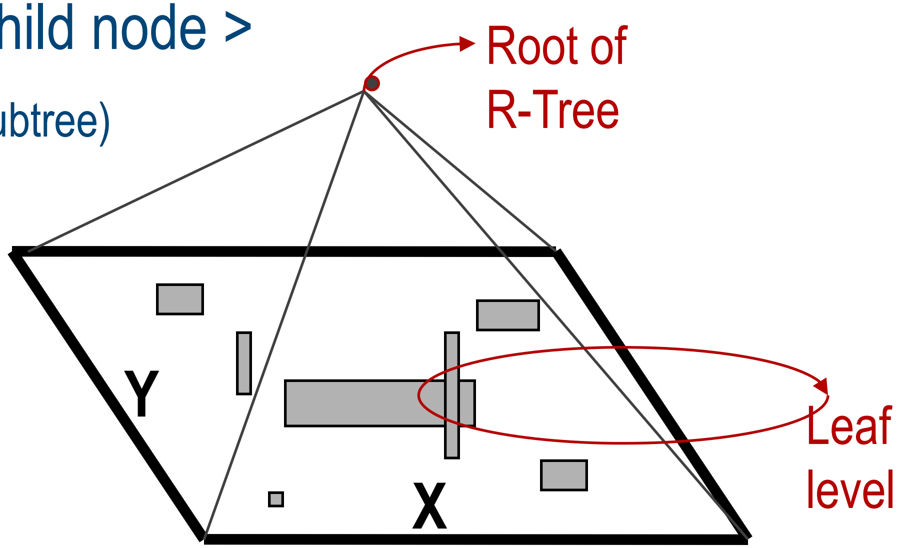
- Problems:
  - Selects way too much data
  - Index space grows with dimensionality

Wanted: a Multi-Dimensional Index

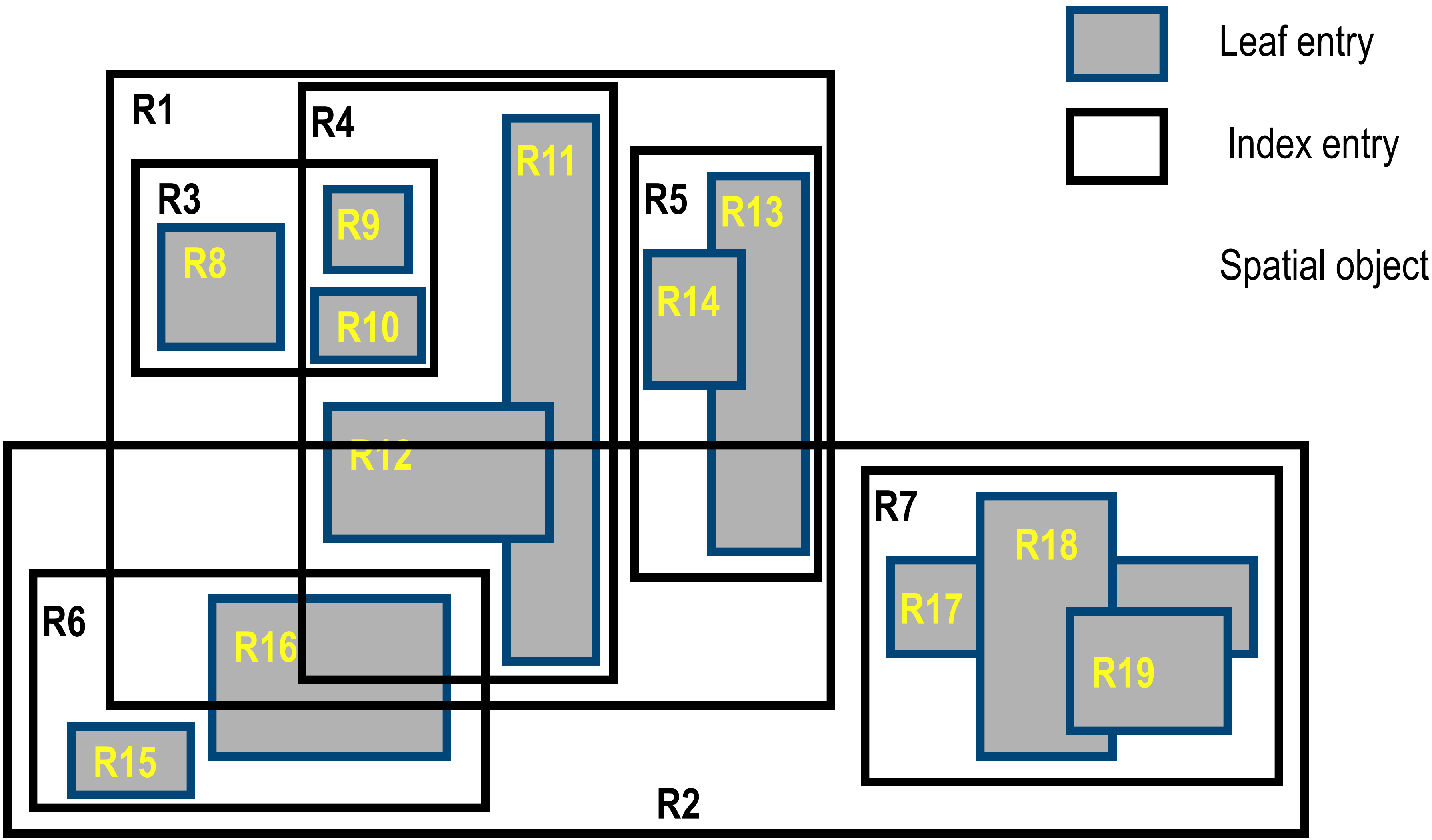
- Requirements:
  - any number of dimensions
  - Symmetric behavior for all dimensions
  - supports inserts and deletes gracefully
  - Ideally, want to support non-point data as well (e.g., lines, shapes)
- Zillions of approaches and variants in literature
  - Grid file, Quad/Oct-tree, kdb-tree, space-filling curves, …
  - Core idea always: spatial clustering of entries on disk
- we look into R-tree
  - widely used, in many variants

The R-Tree

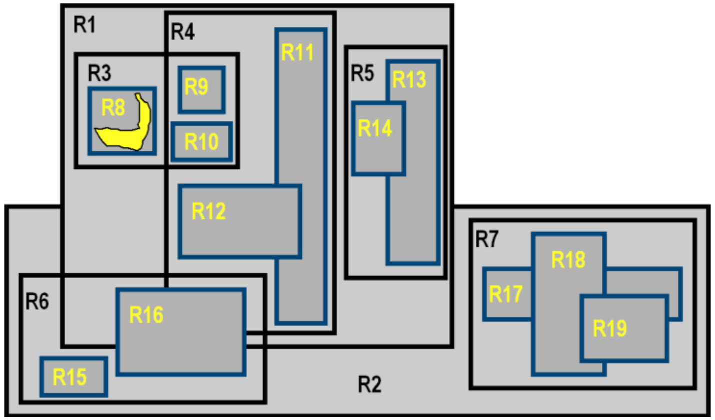
- R-tree = tree-structured n-D index [Guttman 1984]
  - Discriminating value of B+-Tree substituted by bounding intervals (bbox)
  - Index search by bbox, not by exact (polygon) shape
- Leaf entry = < n-dimensional box, rid >
  - tightest bounding box for object
- Non-leaf entry = < n-dim box, ptr to child node >
  - Box covers all boxes in child node (in fact, subtree)
- 2-D sketch:

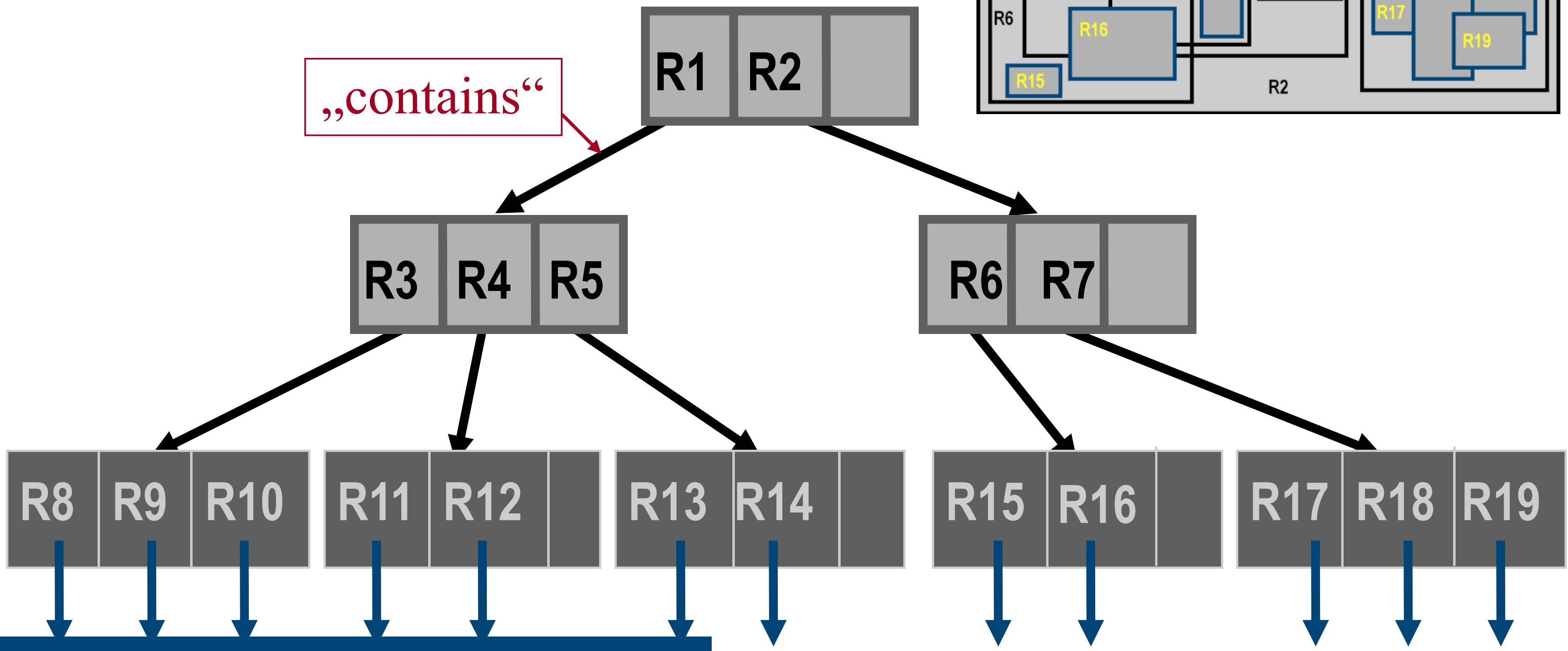


Sample R-Tree

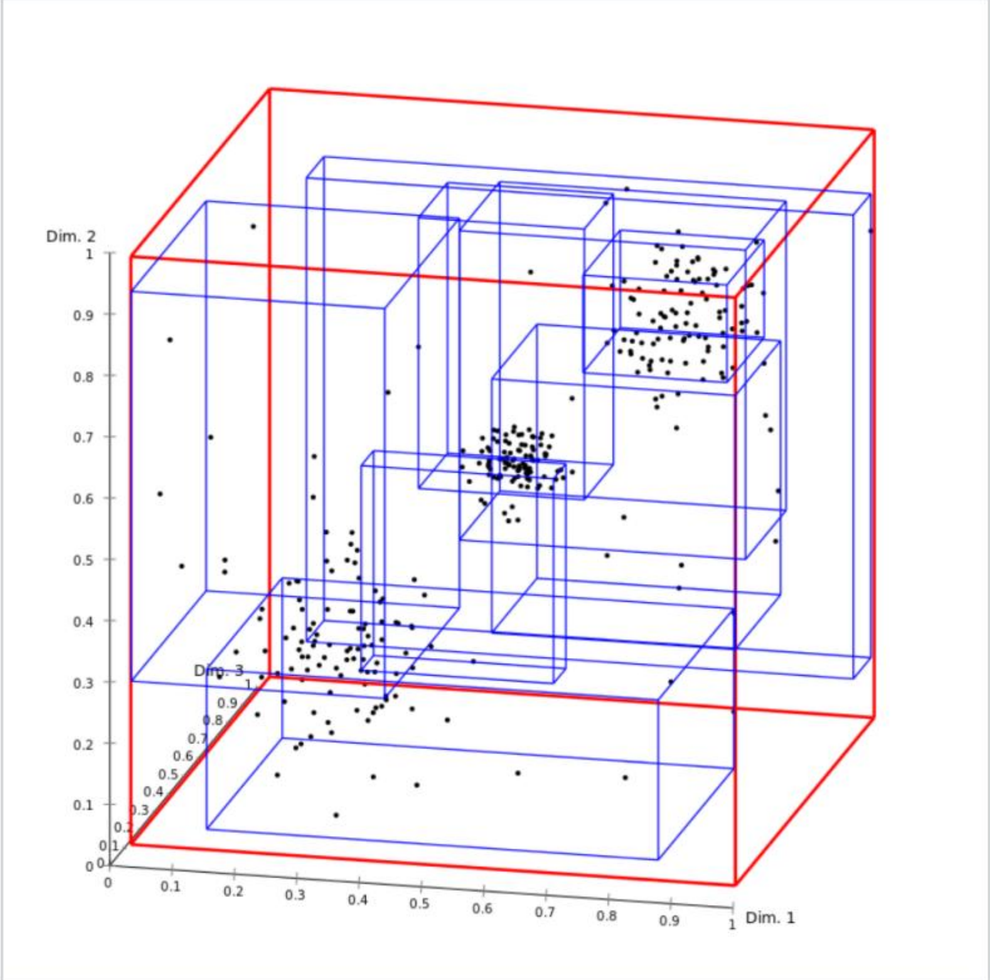


Sample R-Tree (contd.)





Sample 3D R+-Tree [Wikipedia]



Visualization of an R*-tree for 3D points using ELKI (the cubes are directory pages)

Search for Objects Overlapping Box Q

Current node := root;

1. If current node is non-leaf:

for each entry <E, ptr>:

    if *box* E overlaps Q

    then search subtree identified by ptr;

2. If current node is leaf:

for each entry <E, rid>:

    if box E overlaps Q

    then

        rid identifies an object that might overlap Q.

May have to search several subtrees at each node!
(B-tree equality search goes to just one leaf)

Summary

- Index support for multi-dimensional queries has many applications
  - GIS, CAD/CAM, …: spatio-temporal, 2..4-D
  - multimedia indexing, statistical databases:
    non-spatial dimensions, 3-D..12-D..10,000-D…
- Main multidimensional query types:
  - Point, overlap, containment, nearest-neighbor
- Fundamental difference between space/time and feature spaces
  - <4D vs 1000s of dimensions
  - R-tree worse than sequential scan for 12+ D

Summary (contd.)

- Many approaches to multi-dimensional indexing
- R-tree approach widely used in GIS
  - Overall, works quite well for 2..4-D datasets
  - Several variants
- Issues
  - Not for high-dimensional datasets