R-Trees and applications

Efficient grid queries in the Dynamo reservoir modelling software

R-Tree Project

A.N. Yzelman (UU)

Supervisors

A.N. Swart (Alten)

G.L.G. Sleijpen (UU)

Project Owners

Eric Haesen (Alten)

Hans Molenaar (Shell)

Overview

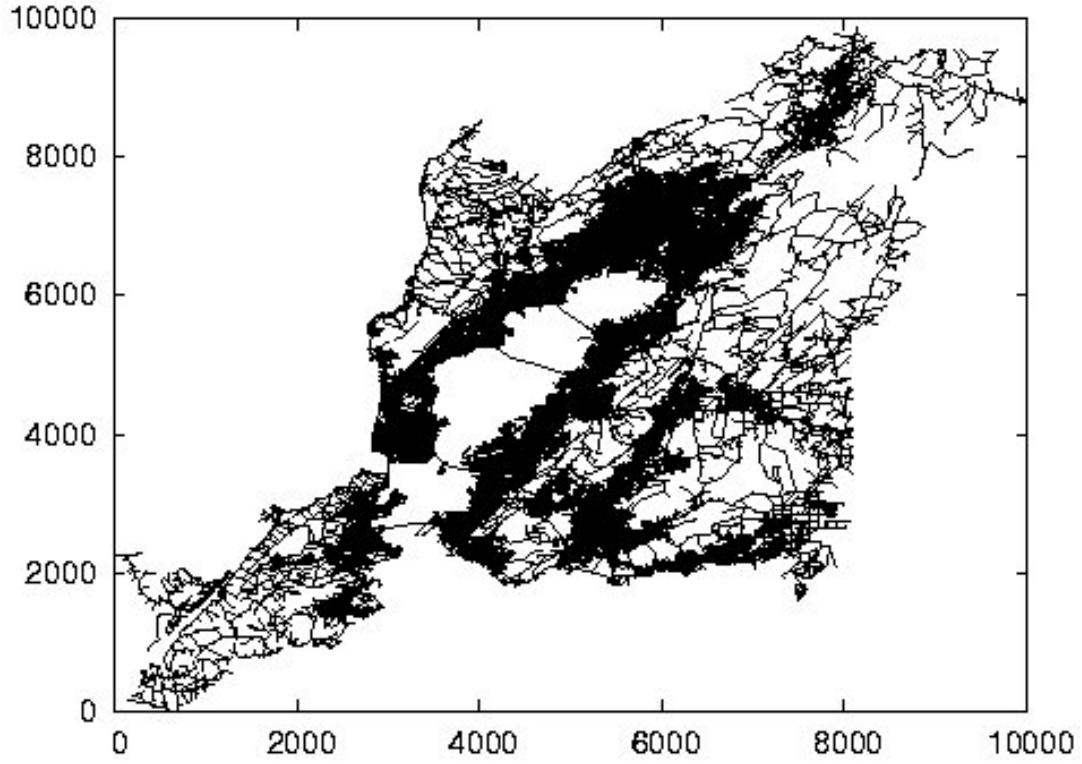
- Problem: Querying large spatial databases.
- Application in Dynamo reservoir simulation software.
- R-Trees.
  - Example
  - Testing
  - Implementation

Querying spatial data

- Needed in areas of:
  1. Image processing
     (Ray-tracing / CAD)
  2. Geographical information systems (e.g., TomTom)
  3. Robotics

Example

The San Francisco road network



Typical Queries

- Containment
  “What objects are contained in ...”
- Neighbours
  “What objects are close to...”
- Intersection
  “What objects cross...”

The Naive Approach

-> Test all objects

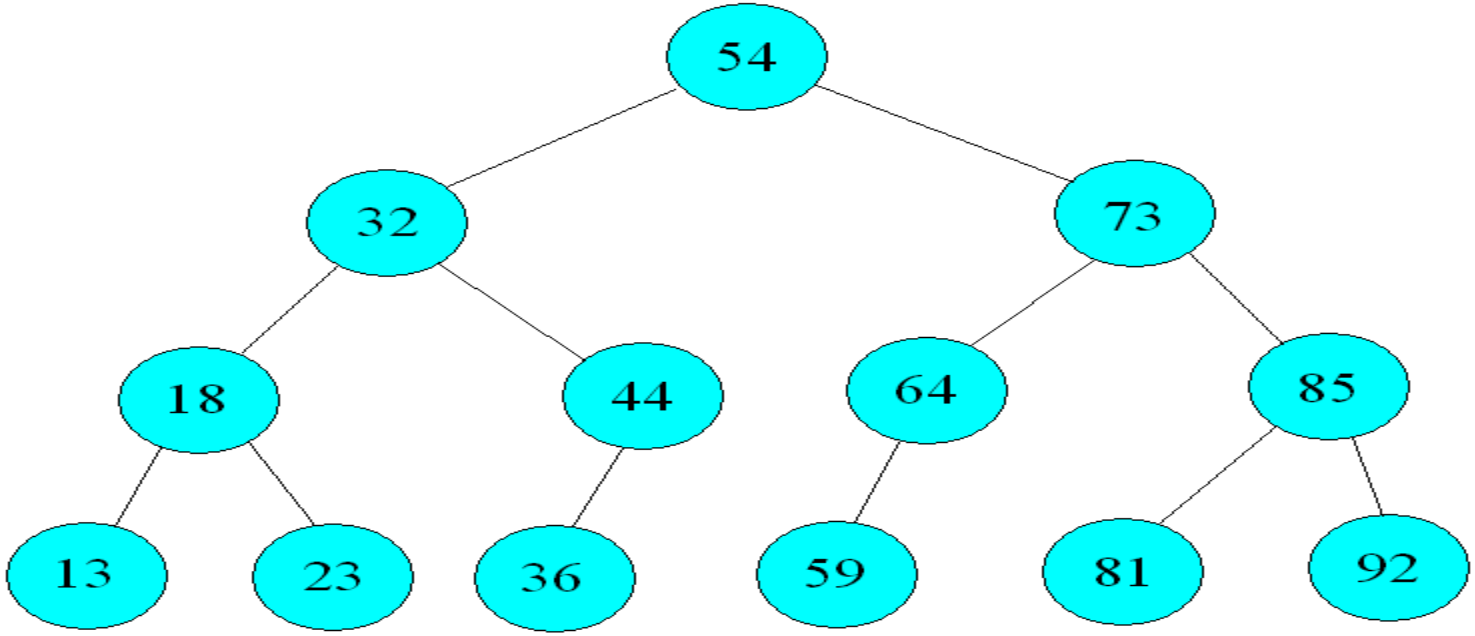
May take enormous amount of time:
174599 nodes for San Francisco Map

Conclusion: need a clever data structure

Trees

Example: Binary Search tree

Each node has max. two children



Trees

- Searching takes O(log(n)) time (2048 entries take twice the time of 1024 entries!)
- Insertion/Deletion may be slower

Main issue for spatial data:

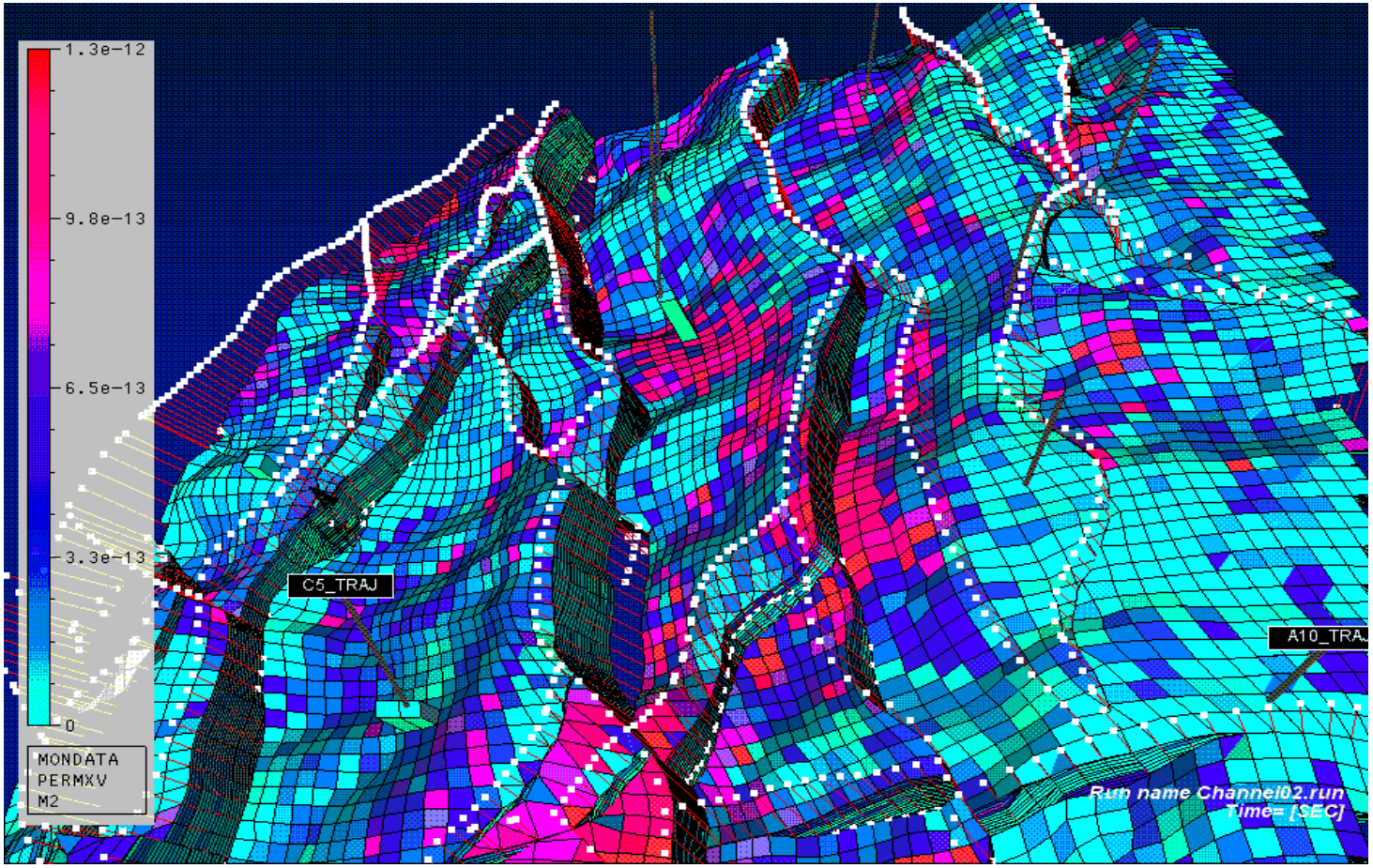
need an ordering

Dynamo

- Dynamo uses high resolution grids
- Queries: neighbours, intersections, overlap
- Currently uses a bisection method, which may be sub-optimal

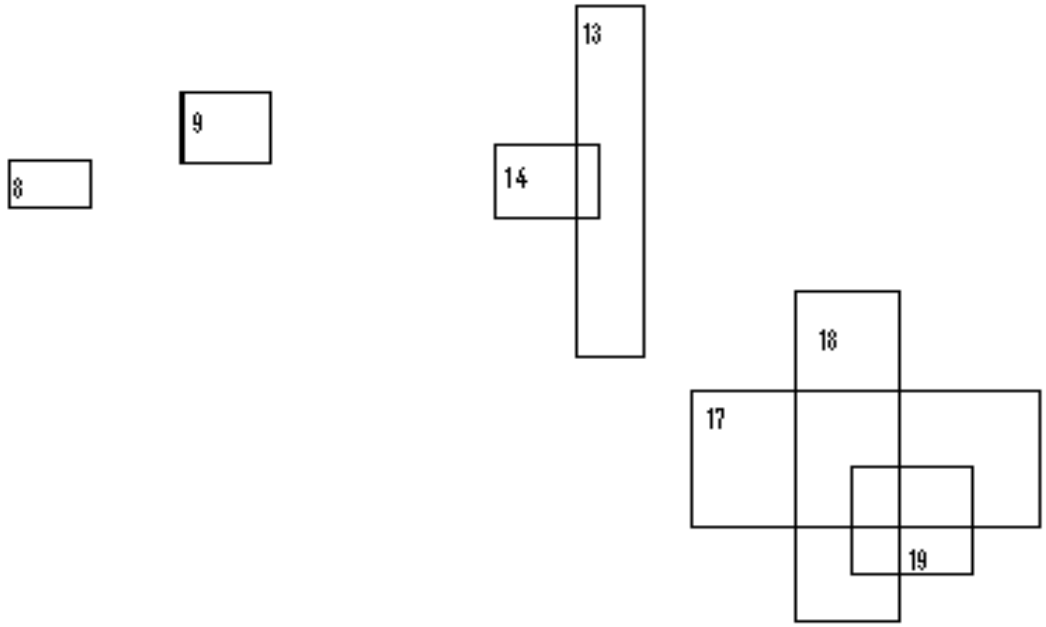
Dynamo may benefit from a custum taylored R-Tree implementation

Example Grid



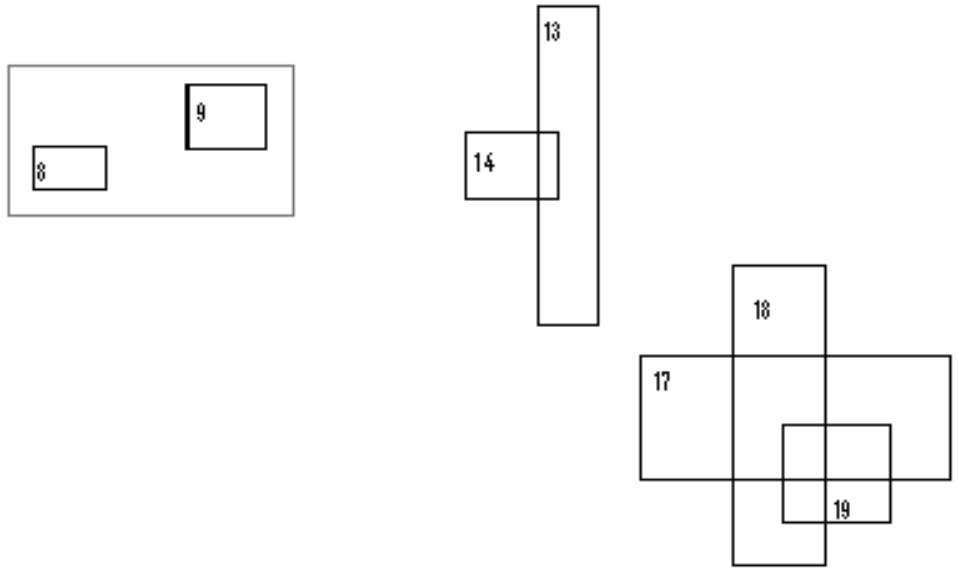
R-Trees

- Consider a simple two-dimensional case:



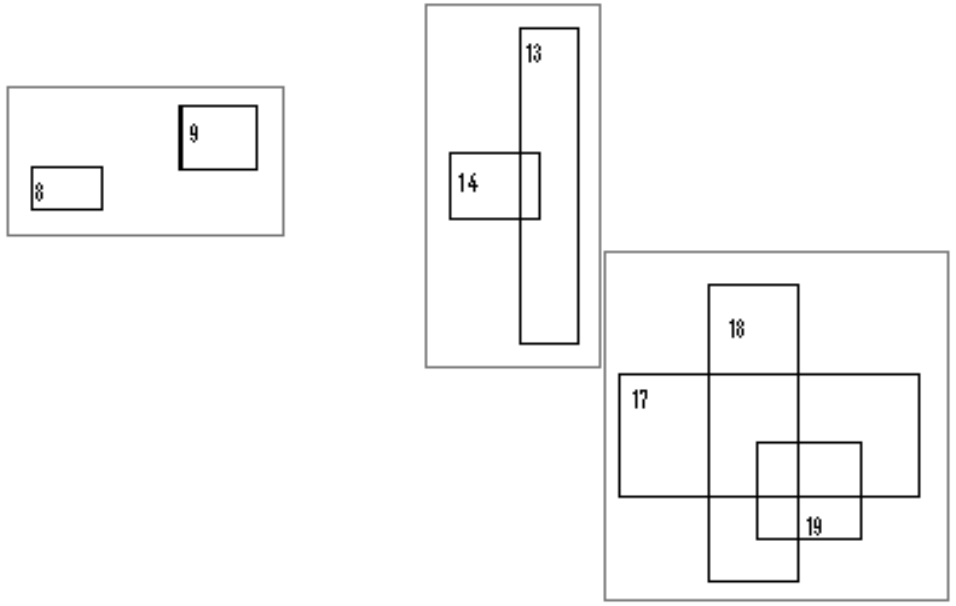
R-Trees

- Add Bounding Boxes



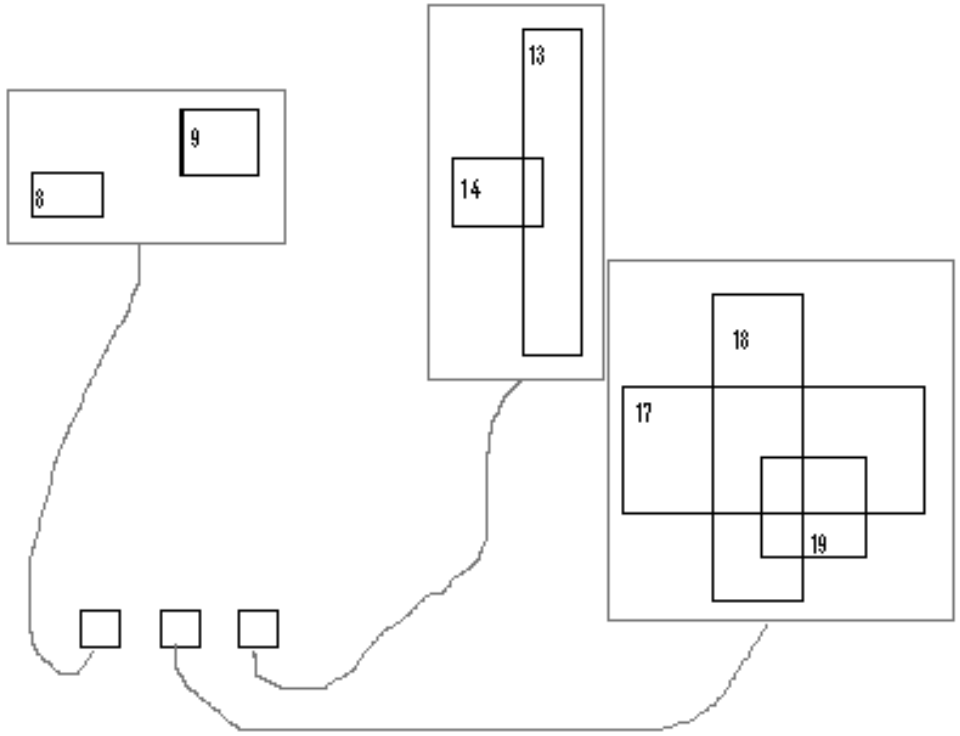
R-Trees

- Add Bounding Boxes



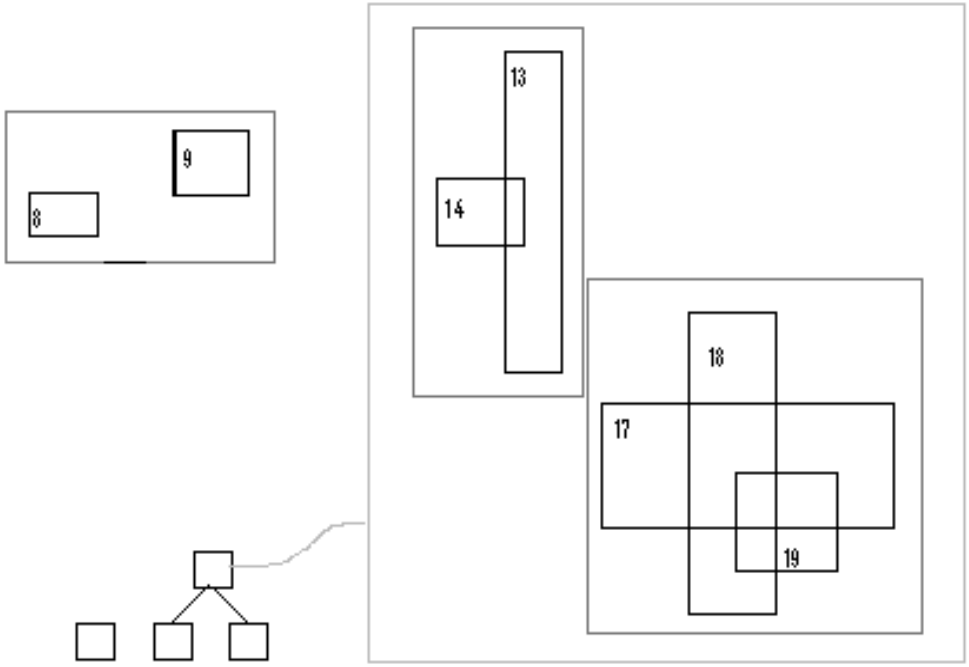
R-Trees

- Add Bounding Boxes



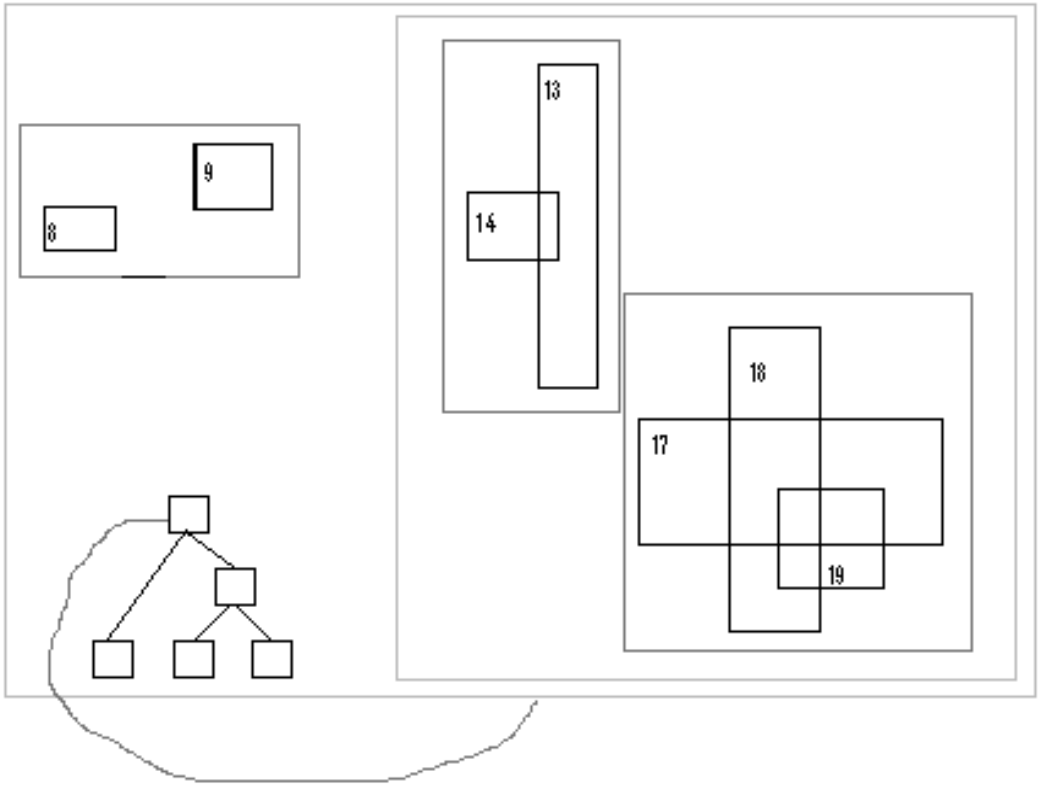
R-Trees

- Add Bounding Boxes



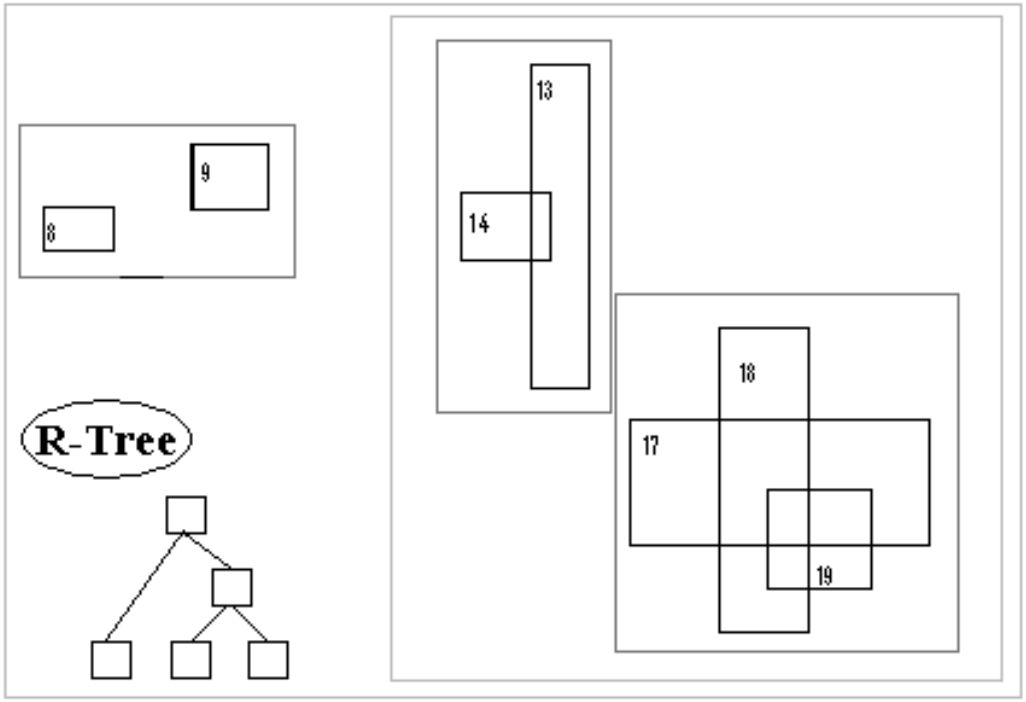
R-Trees

- Add Bounding Boxes



R-Trees

- Add Bounding Boxes



R-Tree Properties

- Every node contains between *m* and *M* data elements unless it is the root.
- All leaves appear at the same level, and contain all the data elements.
- The bounding boxes used tightly encloses the objects within.

R-Tree Properties

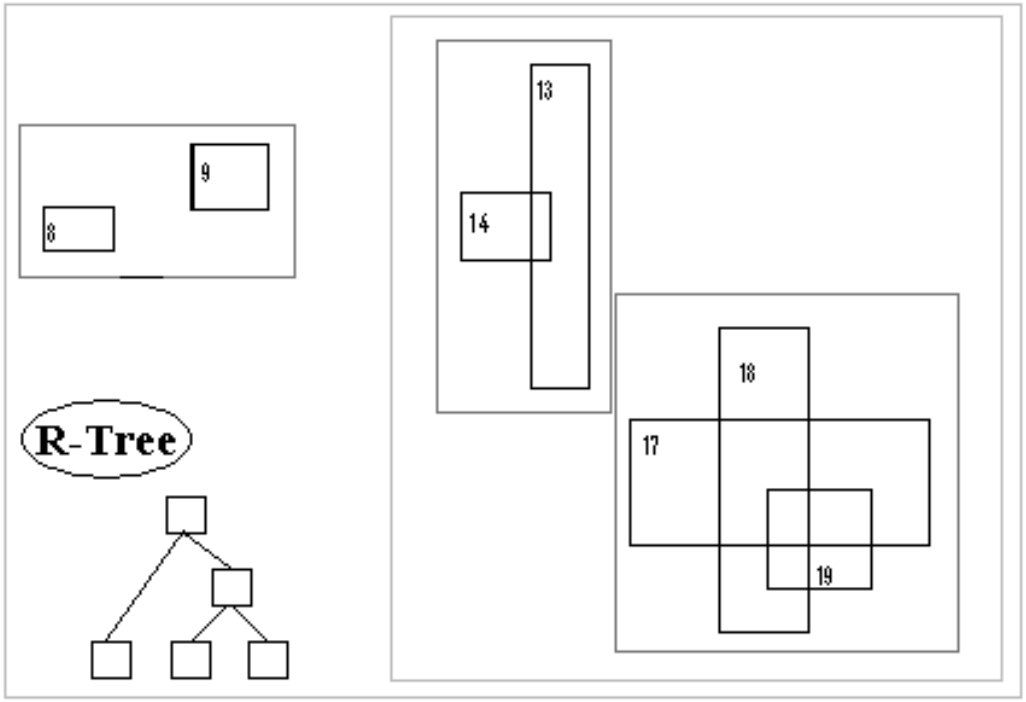
- Every node contains between *m* and *M* data elements unless it is the root.

  m and *M* may be chosen with memory or disk cache size in mind. In the latter case, R-trees reduce to the well known B-trees.

Subtleties

- By the previous example, an R-tree can be built in many different ways.

R-Trees



Variants

- By the previous example, we saw an R-tree can be built in many different ways.
- Leads to many different R-tree variations.

Variants

- By the previous example, we saw an R-tree can be built in many different ways.
- Leads to many different R-tree variations.
- For example:

  R*-tree, PR-tree, Hilbert-tree, R+-tree,

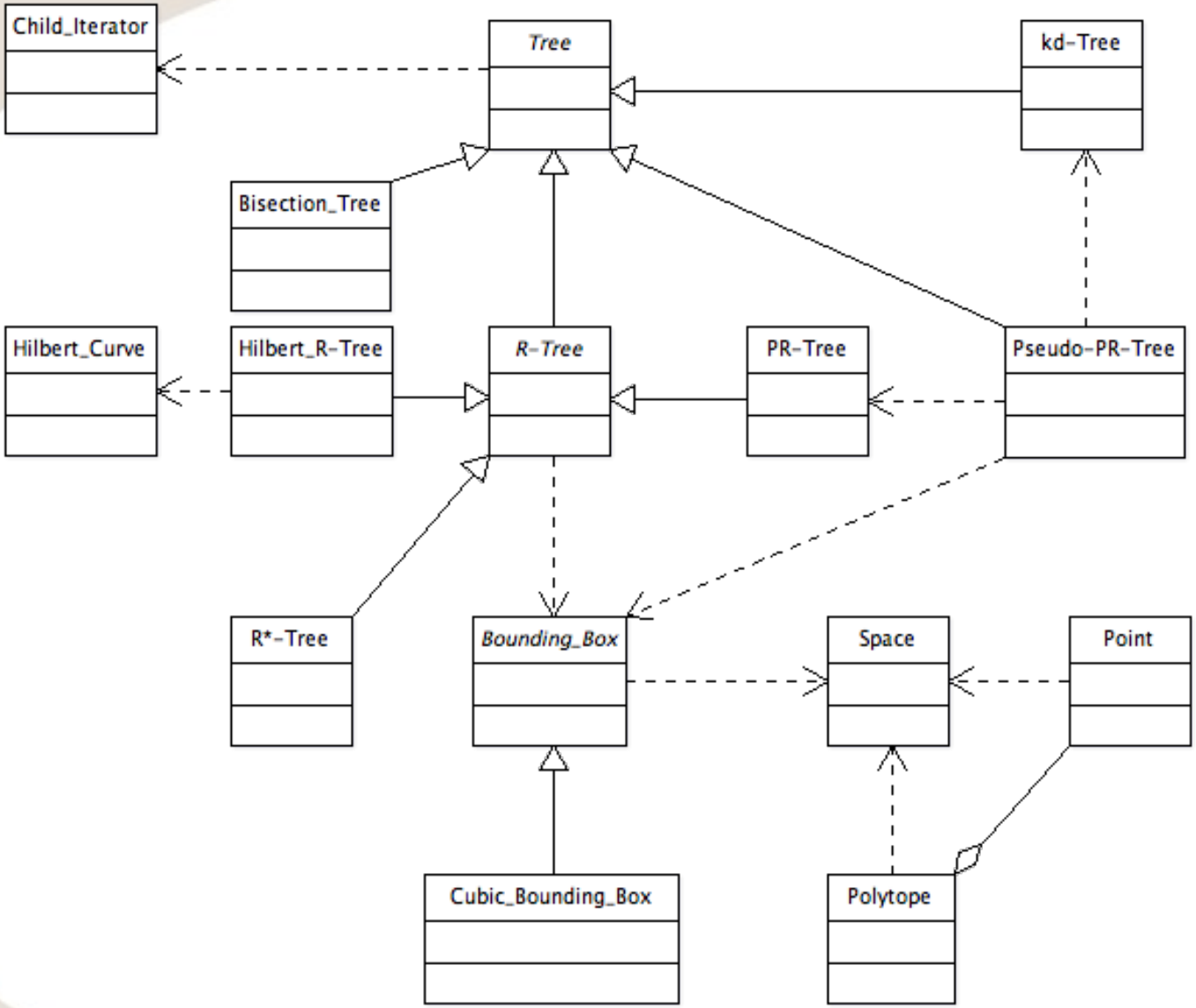
  and more...

Testing

- Use real life geographical data supplied by Shell.
- Test several R-tree variations.
- Tune tree parameters to the specifics of Shell data.
- Quantify results in terms of query time, memory usage and query response quality.

Implementation

- Object-Oriented C++
- Easily extended
- “Plug & Play” solution for spatial data storage
- Generic design targeted at *n*-dimensional polygons in different spaces


Child_Iterator
Tree
kd-Tree
Bisection_Tree
Hilbert_Curve
Hilbert_R-Tree
R-Tree
PR-Tree
Pseudo-PR-Tree
R*-Tree
Bounding_Box
Space
Point
Cubic_Bounding_Box
Polytope

ALTEN

Questions?