Today:

- More multidimensional structures: quad-trees, grid files, partitioned hash, R-trees.
- Overview of query optimization.

Soon:

- Algebraic query optimization.
- Join methods.

Grid Files

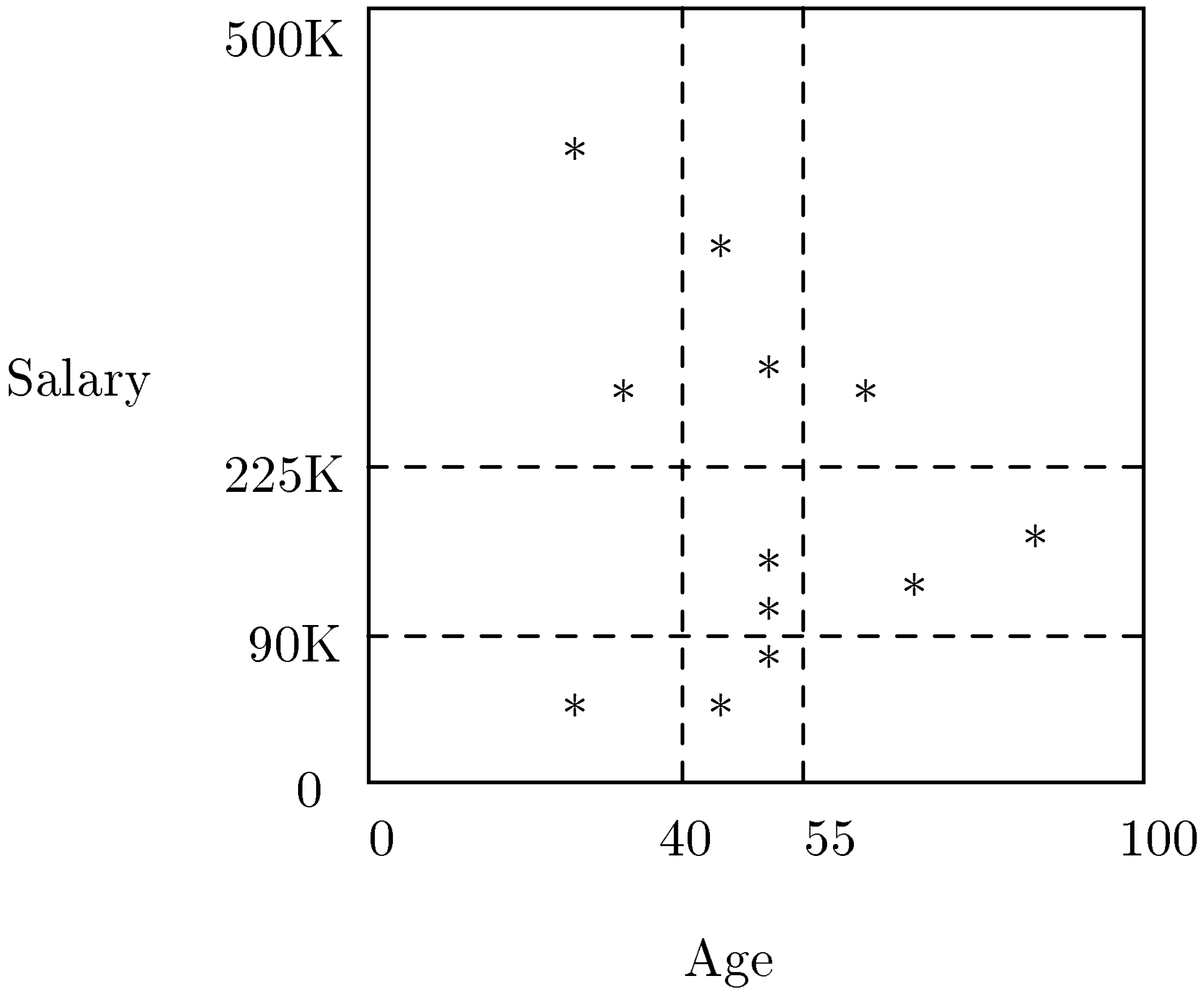
Select dividers along each dimension. Partition points into buckets according to their rectangular region.

Example

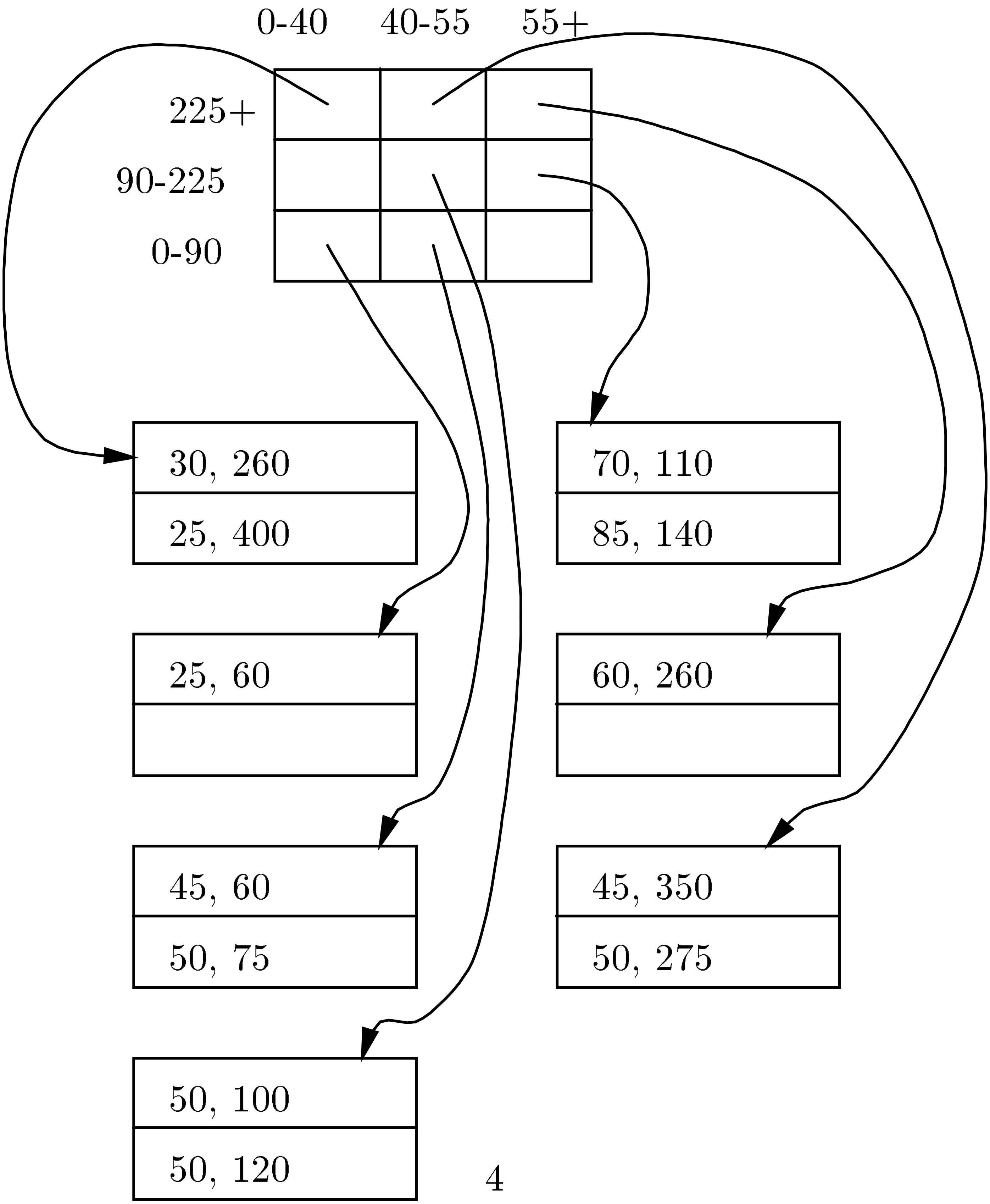
Here are the "gold-jewelry" points again:

$$\begin{array}{llll}
(25, 60) & (45, 60) & (50, 75) & (50, 100) \\
(50, 120) & (70, 110) & (85, 140) & (30, 260) \\
(25, 400) & (45, 350) & (50, 275) & (60, 260)
\end{array}$$

The Grid



The “Hash Table”



Lookup

Find coordinates of point in each dimension — gives you a bucket to search.

Nearest Neighbor

Lookup point P. Consider points in that bucket.

- Problem: there could be points in adjacent buckets that are closer.
  - Example: NN of $(45, 200)$.
- Problem: there could be no points at all in the bucket: widen search?

Range Queries

Ranges define a region of buckets.

- Buckets on border may contain points not in range.
- Example: $35 < age \leq 45$; $50 < salary \leq 100$.

Queries Specifying Only One Attribute

Problem: must search a whole row or column of buckets.

Insertion

1. If bucket's block is full, add overflow blocks.

   ✦ Note: the SKS version of grid files seems to assume there are a very large number of grid lines, so overflow is not a problem. Rather, buckets need to share blocks, which would be split if they got crowded.

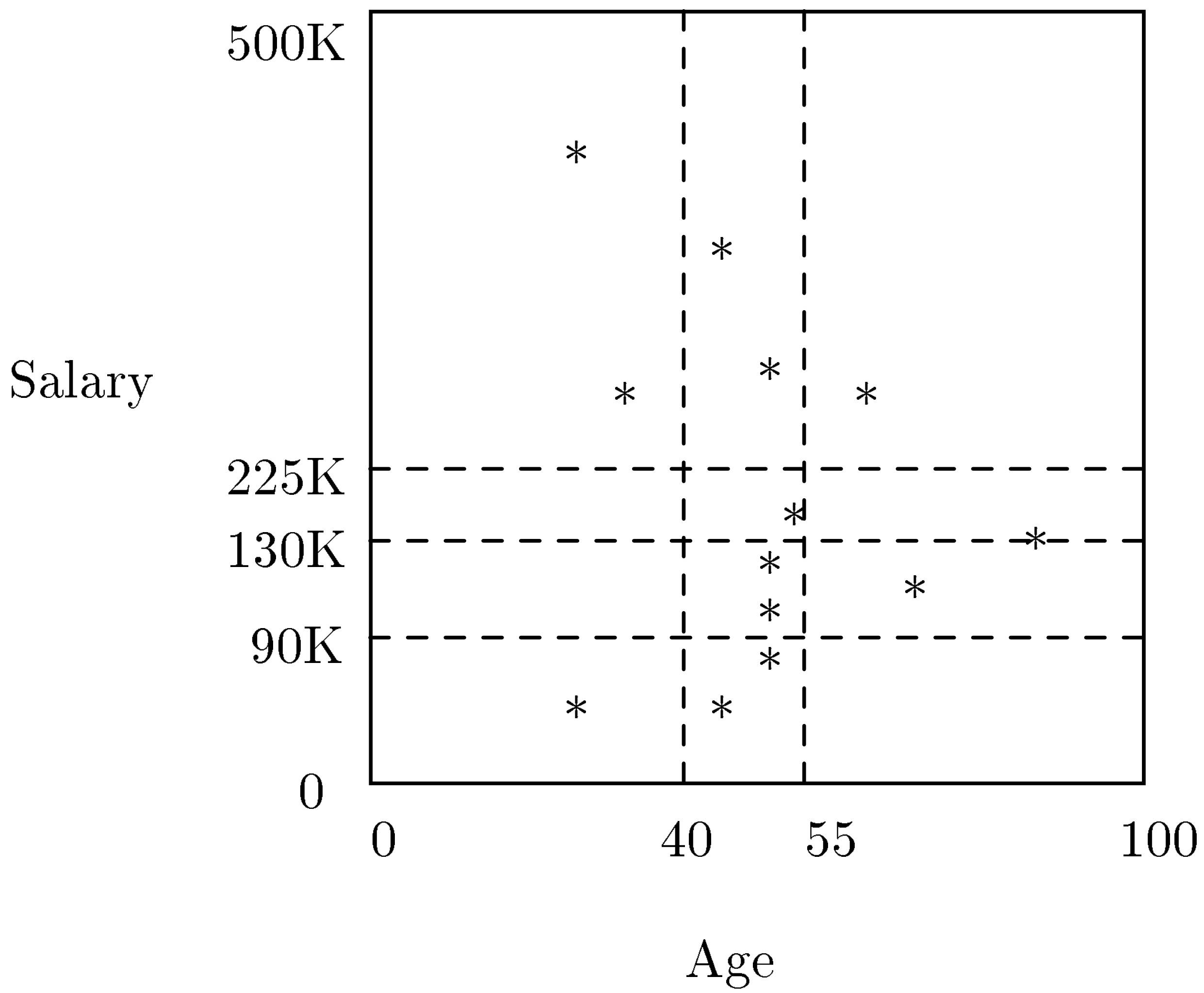
2. Split some buckets by introducing a new divider.

   ✦ Which dimension?

   ✦ What position?

Example

Insert $(52, 200)$; split central bucket. One possibility:



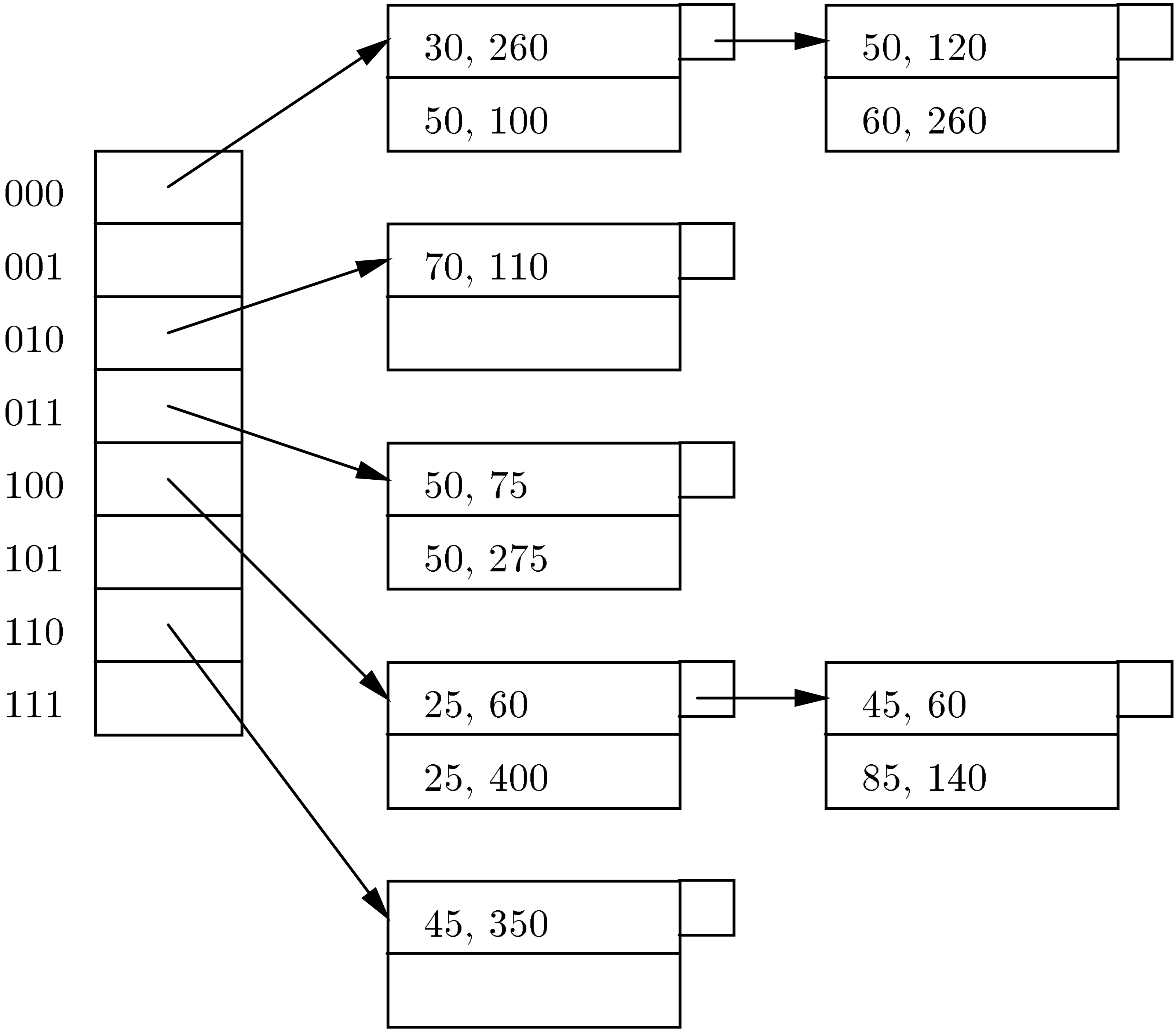
Partitioned Hashing

- A hash function that combines several attributes is great as long as *all* the attributes have values specified.
- A *partitioned* hash function devotes some bits to each attribute independently.

Example

Gold-jewelry data with:

- One bit from age (age modulo 2).
- Two bits from salary (salary modulo 4).



000
001
010
011
100
101
110
111
30, 260
50, 100
50, 120
60, 260
70, 110
50, 75
50, 275
25, 60
25, 400
45, 60
85, 140
45, 350

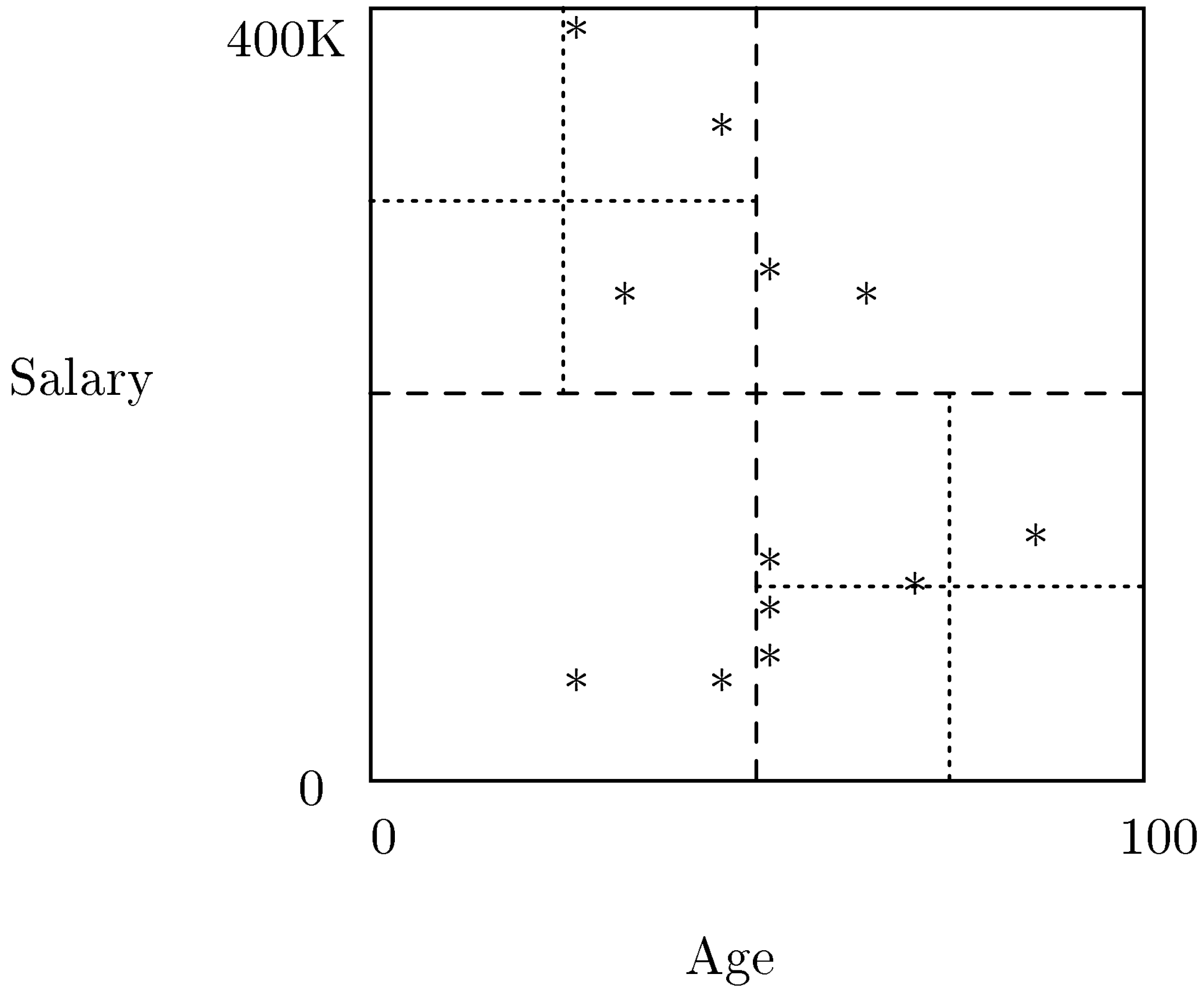
Comparison

- Grid files suffer from sparseness if dimension is high. Partitioned hash tables tend to distribute data evenly among buckets.
- Both support (somewhat) queries in which only some attributes are specified.
- Grid files good for NN and range queries; partitioned hash not.

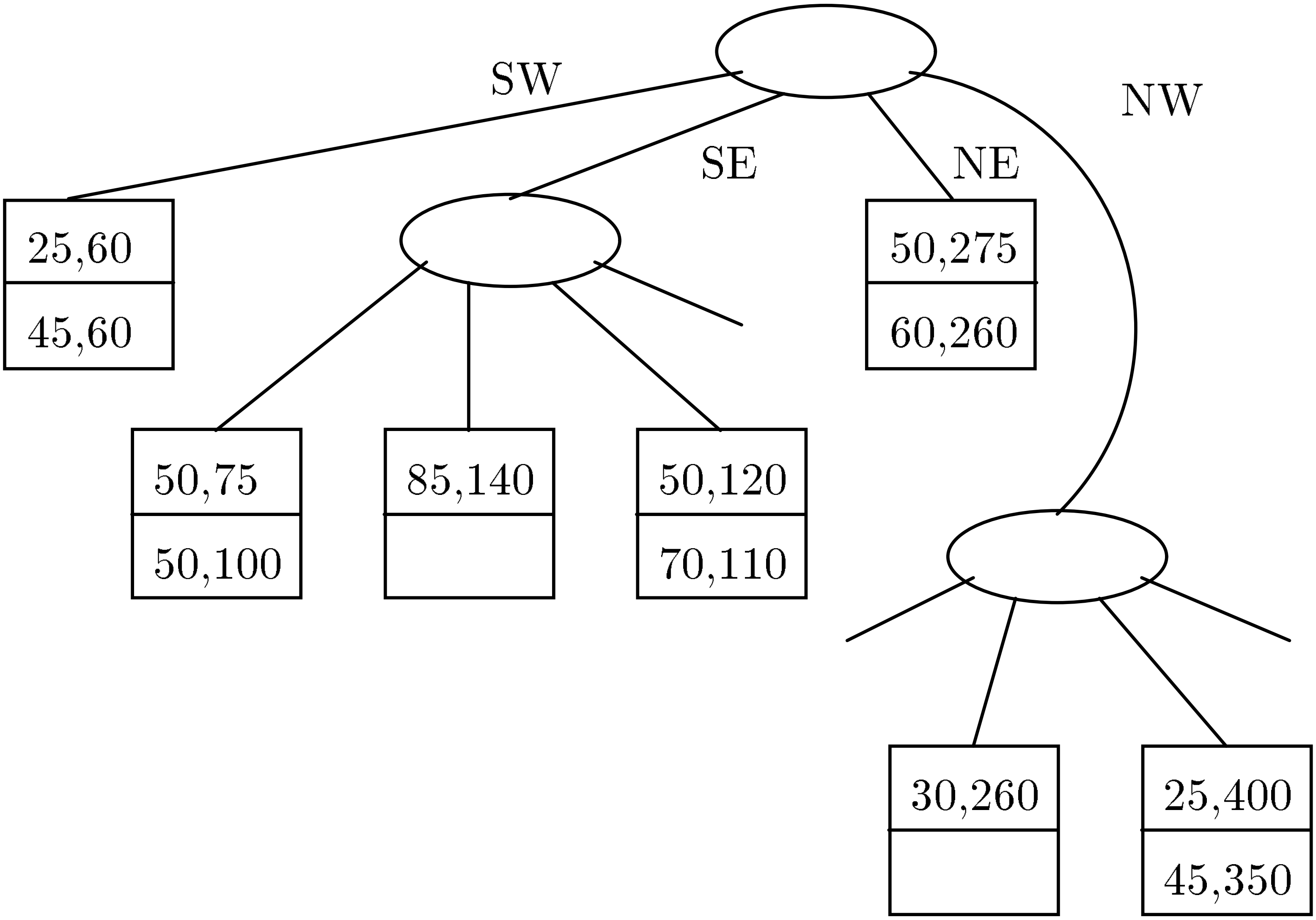
Quad Trees

- Nodes split along all dimensions at once.
- Division fixed: by quadrants.
- As for kd-trees, we cannot make the number of levels uniform.

Example



- Idea generalizes to any number of dimensions.


SW
SE
NE
NW
25,60
45,60
50,275
60,260
50,75
50,100
85,140
50,120
70,110
30,260
25,400
45,350

Quad-Tree Insert

- Find leaf node in which new point belongs.
- If room, put it there.
- If not, make the leaf an interior node and give it leaves for each quadrant. Split the points among the new leaves.
- Problem: may make lots of null pointers, especially in high-dimensions.

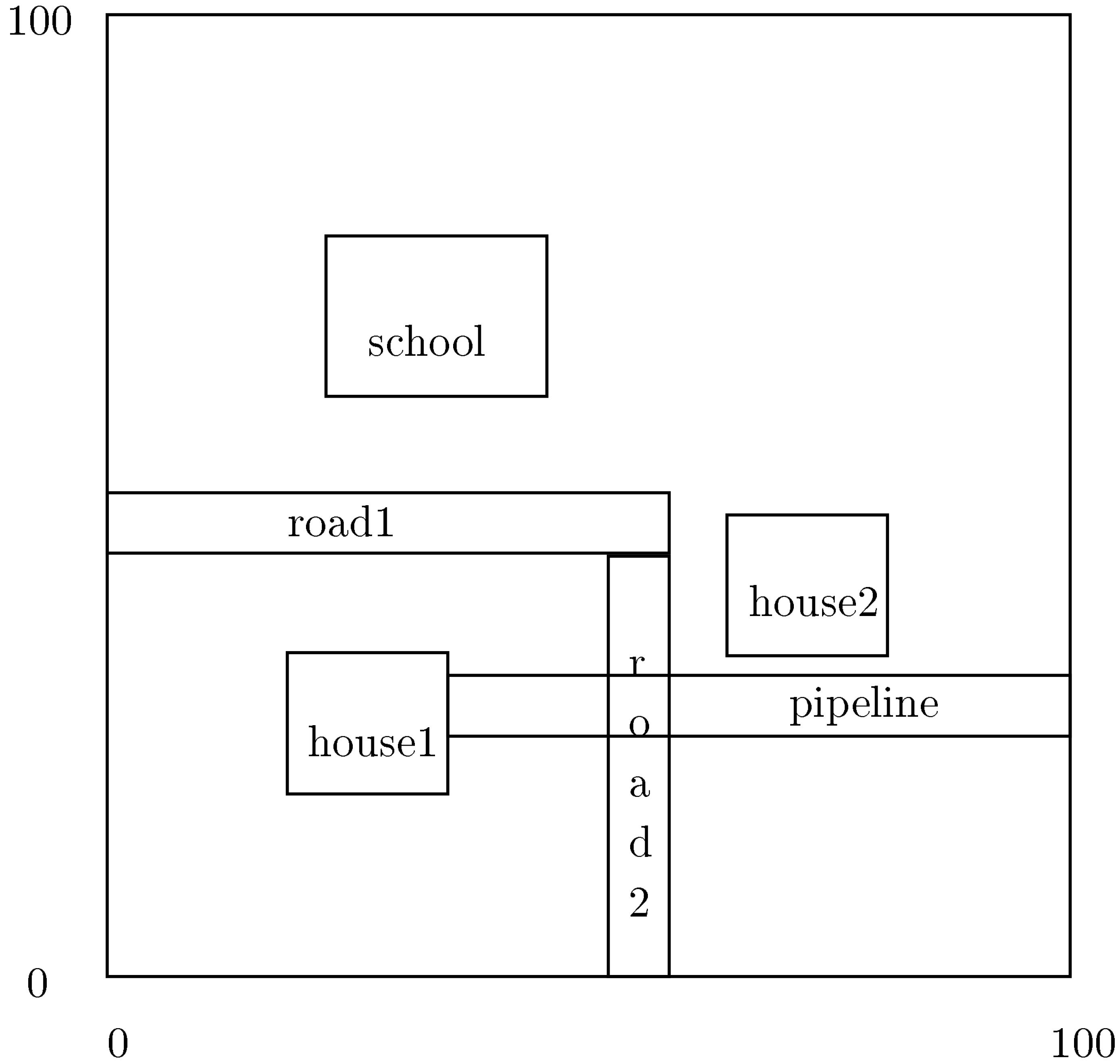
Quad-Tree Queries

- Single point queries: easy; just go down the tree to proper leaf.
- Range queries: varies by position of range.
  - ❖ Example: a range like $45 < age < 55$; $180 < salary < 220$ requires search of four leaves, none of which is guaranteed to produce any answers.
  - ❖ But if range covers a large subtree of the quad tree, then even if we have to search a large number of leaves, we know that everything we find is an answer.
- Nearest neighbor: Problems and strategies similar to grid files.

R-Trees

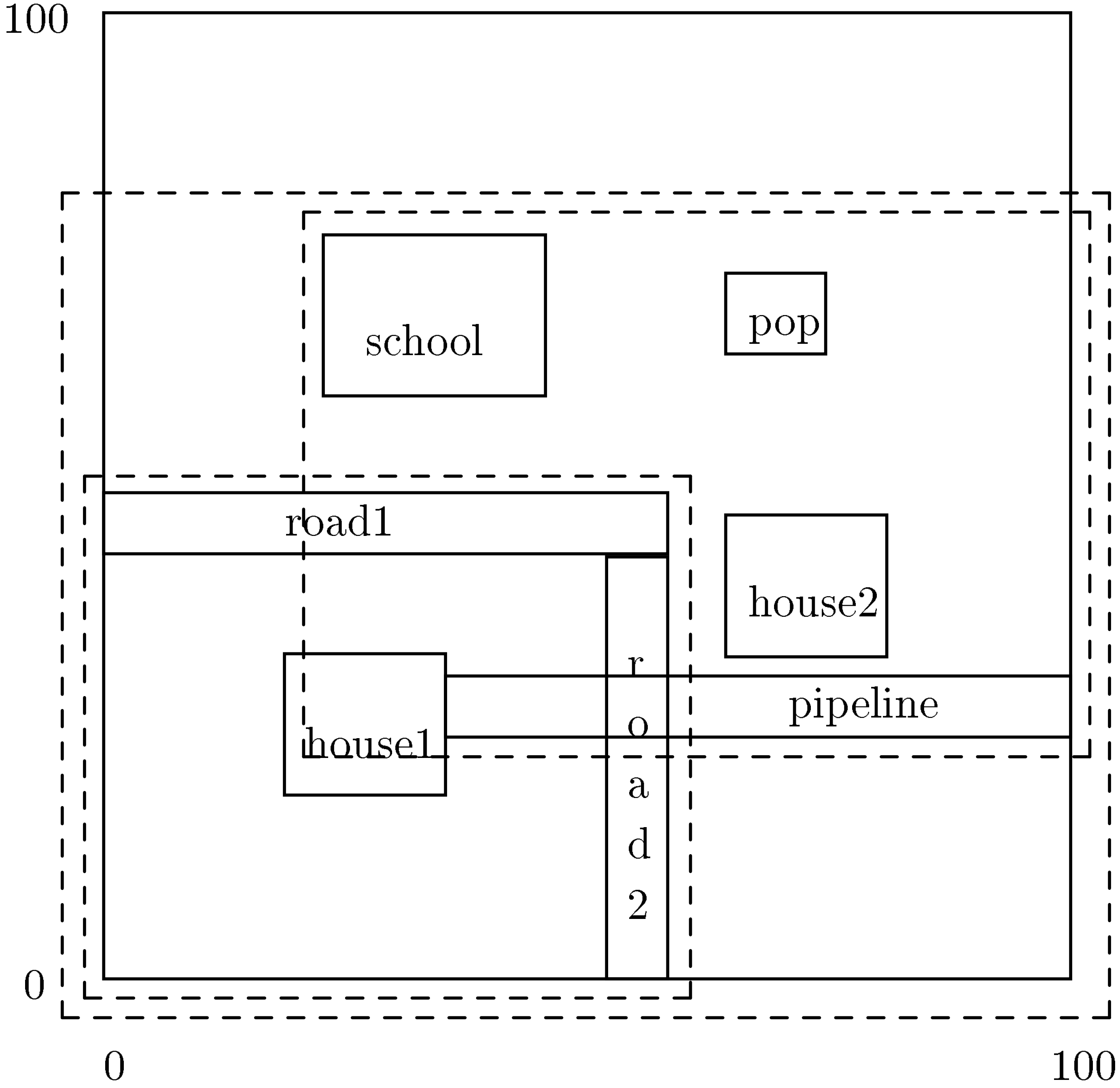
- For “regions” (typically rectangles) but can represent points.
- Supports NN, “where-am-I” queries.
- Generalizes B-tree to multidimensional case.
  - Problem: no ideal way to partition children without overlap.
  - In place of B-tree’s key-pointer pairs, R-tree has region-pointer pairs.

Example



R-Tree for Example

- Root represents largest region.
- Two children represent subregions.
- Region records themselves shown at leaves (B-tree style), but leaves could have pointers to region records with just the corners of the region in the leaf block itself (B+ tree style).


100
0
0
100
school
pop
road1
house2
r
o
a
d
2
pipeline
house1