

CarnegieMellon

# 15-826: Multimedia Databases and Data Mining

Lecture #4: Multi-key and  
Spatial Access Methods - I

*C. Faloutsos*

CarnegieMellon

## Must-Read Material

- MM-Textbook, Chapter 4
- [Bentley75] J.L. Bentley: *Multidimensional Binary Search Trees Used for Associative Searching*, CACM, 18,9, Sept. 1975.
- Ramakrishnan+Gehrke, Chapter 28.1-3

## Outline


Goal: 'Find similar / interesting things'

- Intro to DB
- ➔ • Indexing - similarity search
- Data Mining

## Indexing - Detailed outline

- primary key indexing
- ➔ • secondary key / multi-key indexing
- spatial access methods
- text
- ...

CarnegieMellon




## Problem

- Find employees with
  - Salary in (\$10K, \$20K) and
  - Years-in-company in (5,7)

15-826 Copyright: C. Faloutsos (2019) 5

CarnegieMellon



## Conclusions

- sec. keys: B-tree indices (+ postings lists)
- multi-key, main memory methods:
  - quad-trees
  - k-d-trees

15-826 Copyright: C. Faloutsos (2019) 6

## Sec. key indexing

- attributes w/ duplicates (eg., EMPLOYEES, with 'job-code', 'salary', 'dept')
- Query types:

## Sec. key indexing

- attributes w/ duplicates (eg., EMPLOYEES, with 'job-code', 'salary', 'dept')
- Query types:
  - exact match
  - partial match
    - 'job-code' = 'PGM' and 'dept' = 'R&D'
  - range queries
    - 'job-code' = 'ADMIN' and salary < 50K

## Sec. key indexing

- Query types - cont' d
  - boolean
    - 'job-code' = 'ADMIN' or salary>20K
  - nn
    - salary ~ 30K

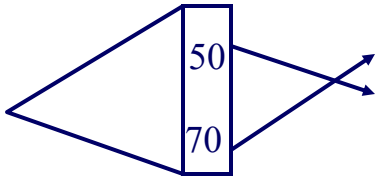
## Solution?

CarnegieMellon

## Solution?

- Inverted indices (usually, w/ B-trees)
- Q: how to handle duplicates?

salary-index



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
....			
Tomson	ENG	50	SALES

15-826 Copyright: C. Faloutsos (2019) 11

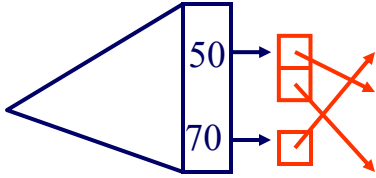
CarnegieMellon

## Solution

- A#1: eg., with postings lists

salary-index

postings lists



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
....			
Tomson	ENG	50	SALES

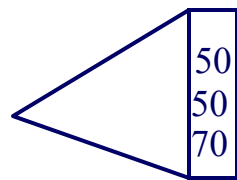
15-826 Copyright: C. Faloutsos (2019) 12

CarnegieMellon

## Solution

- A#2: modify B-tree code, to handle dup' s

salary-index



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
....			
Tomson	ENG	50	SALES

15-826

Copyright: C. Faloutsos (2019)

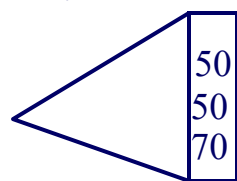
13

CarnegieMellon

## How to handle Boolean Queries?

- eg., 'sal=50 AND job-code=PGM' ?

salary-index



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
....			
Tomson	ENG	50	SALES

15-826

Copyright: C. Faloutsos (2019)

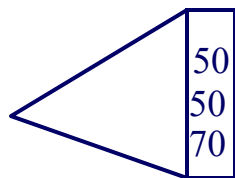
14

CarnegieMellon

## How to handle Boolean Queries?

- from indices, find lists of qual. record-ids
- merge lists (or check real records)

salary-index



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
....			
Tomson	ENG	50	SALES

15-826

Copyright: C. Faloutsos (2019)

15

CarnegieMellon

## Sec. key indexing

- easily solved in commercial DBMS:  

```
create index sal-index on
EMPLOYEE (salary);
select * from EMPLOYEE
where salary > 50 and
job-code = 'ADMIN'
```

15-826

Copyright: C. Faloutsos (2019)

16



CarnegieMellon

## Sec. key indexing

- can create combined indices:  
`create index sj on EMPLOYEE(  
salary, job-code);`

15-826

Copyright: C. Faloutsos (2019)

17

CarnegieMellon

## Sec. key indexing

- can create combined indices:  
`create index sj on EMPLOYEE(  
salary, job-code);`

Q: Drawback?

15-826

Copyright: C. Faloutsos (2019)

18

CarnegieMellon

## Sec. key indexing

- can create combined indices:  
`create index sj on EMPLOYEE(  
salary, job-code);`

Q: Drawback?

A: can not answer queries on job-code

15-826

Copyright: C. Faloutsos (2019)

19

CarnegieMellon

## Indexing - Detailed outline

- primary key indexing
- ➔ • secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
- spatial access methods
- text
- ...

15-826

Copyright: C. Faloutsos (2019)

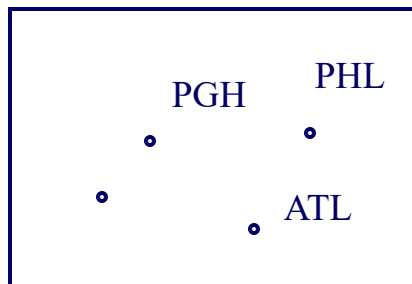
20

## Quad-trees

- problem: find cities within 100mi from Pittsburgh
- assumption: all fit in main memory
- Q: how to answer such queries quickly?

## Quad-trees

- A: recursive decomposition of space, e.g.:



CarnegieMellon

## Quad-trees

- A: recursive decomposition of space, e.g.:

The diagram shows a 2D space with a vertical line at x=30 and a horizontal line at y=10. The quadrants are labeled PGH (top-left), PHL (top-right), SW (bottom-left), and ATL (bottom-right). Three points are plotted: one in PGH, one in PHL, and one in ATL. To the right, a tree structure shows a root node labeled (30,10) with four children representing the quadrants SW, PHL, PGH, and ATL.

15-826 Copyright: C. Faloutsos (2019) 23

CarnegieMellon

## Quad-trees

- A: recursive decomposition of space, e.g.:

The diagram shows a 2D space with a vertical line at x=30 and a horizontal line at y=10. The quadrants are labeled PGH (top-left), PHL (top-right), SW (bottom-left), and ATL (bottom-right). Three points are plotted: one in PGH, one in PHL, and one in ATL. A second vertical line is drawn at x=40, and a second horizontal line is drawn at y=20, further partitioning the PHL quadrant. To the right, a tree structure shows a root node labeled (30,10) with four children. The SW child is labeled SW, the NE child is labeled NE, and the NE child has a further child node labeled (40,20). The PGH and ATL children are not explicitly labeled in the tree.

15-826 Copyright: C. Faloutsos (2019) 24

CarnegieMellon

## Quad-trees - search?

- find cities with  $(35 < x < 45, 15 < y < 25)$ :

15-826 Copyright: C. Faloutsos (2019) 25

CarnegieMellon

## Quad-trees - search?

- find cities with  $(35 < x < 45, 15 < y < 25)$ :

15-826 Copyright: C. Faloutsos (2019) 26

## Quad-trees - search?

- pseudocode:

```
range-query( tree-ptr, range)
  if (tree-ptr == NULL) exit;
  if (tree-ptr->point within range){
    print tree-ptr->point}
  for each quadrant {
    if ( range intersects quadrant ) {
      range-query( tree-ptr->quadrant-ptr, range);
    }
  }
```

15-826

Copyright: C. Faloutsos (2019)

27

## Quad-trees - k-nn search?

- k-nearest neighbor algo - more complicated:
  - find ‘good’ neighbors and put them in a stack
  - go to the most promising quadrant, and update the stack of neighbors
  - until we hit the leaves

15-826

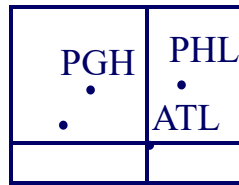
Copyright: C. Faloutsos (2019)

28

## Quad-trees - discussion

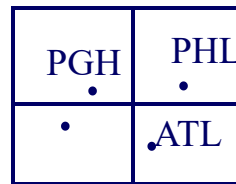
- great for 2- and 3-d spaces
- several variations, like fixed decomposition:

‘adaptive’



‘fixed’

*z-ordering (later)*



middle

15-826

Copyright: C. Faloutsos (2019)

29

## Quad-trees - discussion

- but: unsuitable for higher-d spaces (why?)

15-826


Copyright: C. Faloutsos (2019)

30

## Quad-trees - discussion

- but: unsuitable for higher-d spaces (why?)
- A:  $2^d$  pointers, per node!
- Q: how to solve this problem?
- A: k-d-trees!

## Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  -  – main memory: k-d-trees
- spatial access methods
- text
- ...



CarnegieMellon

## k-d-trees

- Binary trees, with alternating ‘discriminators’

10

30

quad-tree

15-826 Copyright: C. Faloutsos (2019) 33

CarnegieMellon

## k-d-trees

- Binary trees, with alternating ‘discriminators’

10

30

k-d-tree

15-826 Copyright: C. Faloutsos (2019) 34

CarnegieMellon

## k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2019) 35

CarnegieMellon

## k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2019) 36

## (Several demos/applets, e.g.)

- <http://donar.umiacs.umd.edu/quadtree/points/kdtree.html>

## Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
    - insertion; deletion
    - range query; k-nn query
- spatial access methods
- text
- ...

CarnegieMellon

## k-d-trees - insertion

- Binary trees, with alternating ‘discriminators’

The diagram shows a 2D plot with points PGH, PHL, and ATL. The x-axis has markers at 30 and 40, and the y-axis at 10 and 20. A vertical line at x=30 and a horizontal line at y=20 partition the space. To the right, a binary tree has root node (40,20) with children (30,10) and another node. The root node is labeled with 'PHL' and its children with 'ATL'. Discriminators are shown as  $x \leq 30$  and  $x > 30$  for the root, and  $y \leq 20$  and  $y > 20$  for the child nodes. Red lines indicate the axes x and y.

15-826 Copyright: C. Faloutsos (2019) 39

CarnegieMellon

## k-d-trees - insertion

- discriminators: may cycle, or ....
- Q: which should we put first?

The diagram shows a 2D plot with points PGH, PHL, and ATL. The x-axis has markers at 30 and 40, and the y-axis at 10 and 20. A vertical line at x=30 and a horizontal line at y=20 partition the space. To the right, a binary tree has root node (40,20) with children (30,10) and another node. The root node is labeled with 'PHL' and its children with 'ATL'. Discriminators are shown as  $x \leq 30$  and  $x > 30$  for the root, and  $y \leq 20$  and  $y > 20$  for the child nodes. Red lines indicate the axes x and y.

15-826 Copyright: C. Faloutsos (2019) 40

CarnegieMellon

## k-d-trees - deletion

- How?

The diagram shows a 2D plot on the left and a k-d tree on the right. The plot has a vertical axis labeled 'y' with values 10 and 20, and a horizontal axis labeled 'x' with values 30 and 40. Three points are plotted: PGH at approximately (25, 15), PHL at (35, 20), and ATL at (35, 10). The plot is divided into four quadrants by a vertical line at x=30 and a horizontal line at y=20. The k-d tree on the right has a root node (30,10) with a split line x=30. The left child is labeled 'ATL' and the right child is labeled 'PHL'. The 'PHL' node has a split line y=20, with a left child labeled 'PHL' and a right child labeled 'ATL'. Red lines and labels 'x' and 'y' indicate the split lines.

15-826 Copyright: C. Faloutsos (2019) 41

CarnegieMellon

## k-d-trees - deletion

- Tricky! 'delete-and-promote' (or 'tombstone' = 'mark as deleted')

The diagram is identical to the one above, showing a 2D plot and a k-d tree. The plot has a vertical axis labeled 'y' with values 10 and 20, and a horizontal axis labeled 'x' with values 30 and 40. Three points are plotted: PGH at approximately (25, 15), PHL at (35, 20), and ATL at (35, 10). The plot is divided into four quadrants by a vertical line at x=30 and a horizontal line at y=20. The k-d tree on the right has a root node (30,10) with a split line x=30. The left child is labeled 'ATL' and the right child is labeled 'PHL'. The 'PHL' node has a split line y=20, with a left child labeled 'PHL' and a right child labeled 'ATL'. Red lines and labels 'x' and 'y' indicate the split lines.

15-826 Copyright: C. Faloutsos (2019) 42

CarnegieMellon

## k-d-trees - range query

The diagram shows a 2D plot with x-axis values 30 and 40, and y-axis values 10 and 20. Points are labeled PGH, PHL, and ATL. A red box highlights the PHL point. To the right, a k-d tree structure is shown with root node (30,10) and child nodes (40,20) and ATL. The PHL node is highlighted in the tree.

15-826 Copyright: C. Faloutsos (2019) 43

CarnegieMellon

## k-d-trees - range query

- similar to quad-trees: check the root; proceed to appropriate child(ren).

The diagram shows a 2D plot with x-axis values 30 and 40, and y-axis values 10 and 20. Points are labeled PGH, PHL, and ATL. A red box highlights the PHL point. To the right, a k-d tree structure is shown with root node (30,10) and child nodes (40,20) and ATL. The PHL node is highlighted in the tree.

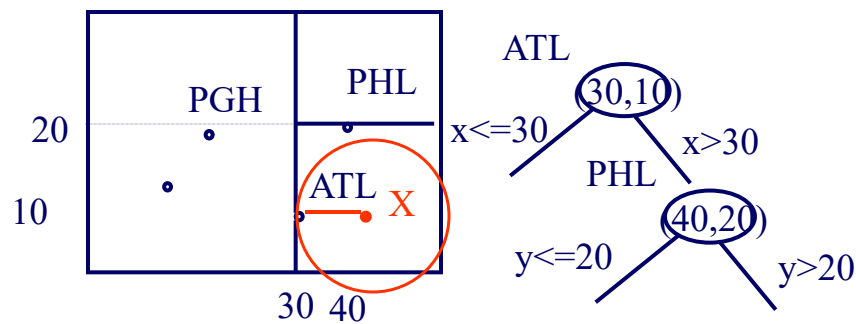
15-826 Copyright: C. Faloutsos (2019) 44



CarnegieMellon

## k-d-trees - k-nn query

- A: check root; put in stack; proceed to child



15-826

Copyright: C. Faloutsos (2019)

47

CarnegieMellon

## Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
    - insertion; deletion
    - range query; k-nn query
    - discussion
- ➔ • spatial access methods
- text

15-826

Copyright: C. Faloutsos (2019)

48



CarnegieMellon

## k-d trees - discussion

- great for main memory & low 'd' ( $\sim < 10$ )
- Q: what about high-d?
- A:
- Q: what about disk
- A:

15-826

Copyright: C. Faloutsos (2019)

49

CarnegieMellon

## k-d trees - discussion


- great for main memory & low 'd' ( $\sim < 10$ )
- Q: what about high-d?
- A: most attributes don't ever become discriminators
- Q: what about disk?
- A: Pagination problems, after ins./del.  
(solutions: next!)

15-826

Copyright: C. Faloutsos (2019)

50

CarnegieMellon



## Conclusions

- sec. keys: B-tree indices (+ postings lists)
- multi-key, main memory methods:
  - quad-trees
  - k-d-trees

15-826 Copyright: C. Faloutsos (2019) 51

CarnegieMellon

## References

- [Bentley75] J.L. Bentley: *Multidimensional Binary Search Trees Used for Associative Searching*, CACM, 18,9, Sept. 1975.
- [Finkel74] R.A. Finkel, J.L. Bentley: *Quadtrees: A data structure for retrieval on composite keys*, ACTA Informatica,4,1, 1974
- Applet: eg.,  
<http://donar.umiacs.umd.edu/quadtree/points/kdtree.html>

15-826 Copyright: C. Faloutsos (2019) 52