

15-826: Multimedia Databases and Data Mining

Multi-key and Spatial Access Methods - I
C. Faloutsos




Outline

Goal: 'Find similar / interesting things'

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


15-826 Copyright: C. Faloutsos (2006) 2



Indexing - Detailed outline

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


15-826 Copyright: C. Faloutsos (2006) 3



Sec. key indexing

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


15-826 Copyright: C. Faloutsos (2006) 4



Sec. key indexing

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

15-826 Copyright: C. Faloutsos (2006) 5



Solution?

15-826 Copyright: C. Faloutsos (2006) 6

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 (2006) 7

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 (2006) 8

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 (2006) 9

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 (2006) 10

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 (2006) 11

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 (2006) 12

CMU SCS

Sec. key indexing

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

15-826 Copyright: C. Faloutsos (2006) 13

CMU SCS

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 (2006) 14

CMU SCS

Quad-trees

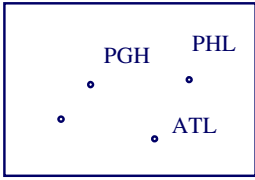
- problem: find cities within 100mi from Pittsburgh
- assumption: all fit in main memory
- Q: how to answer such queries quickly?

15-826 Copyright: C. Faloutsos (2006) 15

CMU SCS

Quad-trees

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

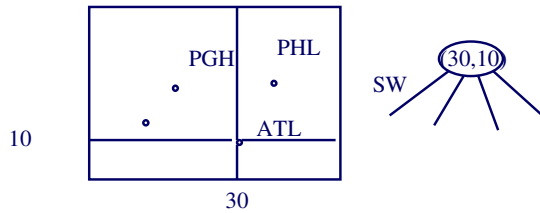


15-826 Copyright: C. Faloutsos (2006) 16

CMU SCS

Quad-trees

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

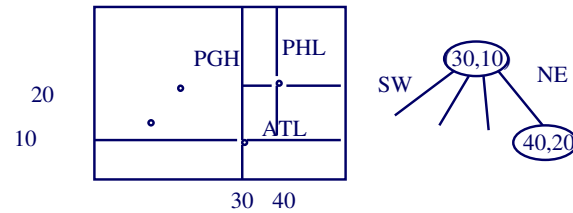


15-826 Copyright: C. Faloutsos (2006) 17

CMU SCS

Quad-trees

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



15-826 Copyright: C. Faloutsos (2006) 18

Quad-trees - search?

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

15-826 Copyright: C. Faloutsos (2006) 19

Quad-trees - search?

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

15-826 Copyright: C. Faloutsos (2006) 20

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 (2006) 21

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 (2006) 22

Quad-trees - discussion

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

‘adaptive’

‘fixed’

15-826 Copyright: C. Faloutsos (2006) 23

Quad-trees - discussion

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

15-826 Copyright: C. Faloutsos (2006) 24

CMU SCS

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!

15-826 Copyright: C. Faloutsos (2006) 25

CMU SCS

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 (2006) 26

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2006) 27

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2006) 28

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2006) 29

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

15-826 Copyright: C. Faloutsos (2006) 30

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
- ...

15-826 Copyright: C. Faloutsos (2006) 31

k-d-trees - insertion

- Binary trees, with alternating ‘discriminators’

15-826 Copyright: C. Faloutsos (2006) 32

k-d-trees - insertion

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

15-826 Copyright: C. Faloutsos (2006) 33

k-d-trees - deletion

- Tricky! ‘delete-and-promote’ (or ‘mark as deleted’)

15-826 Copyright: C. Faloutsos (2006) 34

k-d-trees - range query

15-826 Copyright: C. Faloutsos (2006) 35

k-d-trees - range query

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

15-826 Copyright: C. Faloutsos (2006) 36

k-d-trees - k-nn query

- e.g., 1-nn: closest city to 'X'

15-826 Copyright: C. Faloutsos (2006) 37

k-d-trees - k-nn query

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

15-826 Copyright: C. Faloutsos (2006) 38

k-d-trees - k-nn query

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

15-826 Copyright: C. Faloutsos (2006) 39

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 (2006) 40

k-d trees - discussion


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

15-826 Copyright: C. Faloutsos (2006) 41

k-d trees - discussion

- great for main memory & low 'd' (~<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 (2006) 42



Conclusions

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

15-826 Copyright: C. Faloutsos (2006) 43



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

15-826 Copyright: C. Faloutsos (2006) 44