15-826: Multimedia Databases and Data Mining

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

Must-Read Material

- MM-Textbook, Chapter 4
- 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
- ...
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

Sec. key indexing

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

Solution?

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Job-code</th>
<th>Salary</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>PGM</td>
<td>70</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Jones</td>
<td>ADMIN</td>
<td>50</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Tomson</td>
<td>ENG</td>
<td>50</td>
<td>SALES</td>
</tr>
</tbody>
</table>
Solution

• A#1: eg., with postings lists

<table>
<thead>
<tr>
<th>Name</th>
<th>Job-code</th>
<th>Salary</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>PGM</td>
<td>70</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Jones</td>
<td>ADMIN</td>
<td>50</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tomson</td>
<td>ENG</td>
<td>50</td>
<td>SALES</td>
</tr>
</tbody>
</table>

How to handle Boolean Queries?

• eg., ‘sal=50 AND job-code=PGM’?

<table>
<thead>
<tr>
<th>Name</th>
<th>Job-code</th>
<th>Salary</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>PGM</td>
<td>70</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Jones</td>
<td>ADMIN</td>
<td>50</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tomson</td>
<td>ENG</td>
<td>50</td>
<td>SALES</td>
</tr>
</tbody>
</table>

Solution

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Job-code</th>
<th>Salary</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>PGM</td>
<td>70</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>Jones</td>
<td>ADMIN</td>
<td>50</td>
<td>R&amp;D</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tomson</td>
<td>ENG</td>
<td>50</td>
<td>SALES</td>
</tr>
</tbody>
</table>

How to handle Boolean Queries?

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

• easily solved in commercial DBMS:
  
  
  ```sql
  create index sal-index on EMPLOYEE (salary);
  select * from EMPLOYEE
  where salary > 50 and job-code = ‘ADMIN’
  ```

Sec. key indexing

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

Indexing - Detailed outline

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

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.:

Quad-trees

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

Quad-trees - search?

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

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

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);
    }
  }

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

Quad-trees - discussion

• great for 2- and 3-d spaces
• several variations, like fixed decomposition:
  ‘adaptive’
  ‘fixed’
  z-ordering (later)
Quad-trees - discussion

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

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
- ...

k-d-trees

- Binary trees, with alternating ‘discriminators’

```
   PHL
   |
   PGH
   |
  ---
  |   |
  |   |
  ATL
   |
   SW
```

10

30

quad-tree
**k-d-trees**

- Binary trees, with alternating ‘discriminators’

(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
- ...

k-d-trees - insertion

- Binary trees, with alternating ‘discriminators’

k-d-trees - deletion

- How?
k-d-trees - deletion
• Tricky! ‘delete-and-promote’ (or ‘mark as deleted’)

k-d-trees - range query
• similar to quad-trees: check the root; proceed to appropriate child(ren).

k-d-trees - 1-nn query
• e.g., 1-nn: closest city to ‘X’
k-d-trees - k-nn query

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

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

k-d trees - discussion

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

Conclusions

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

References

• Applet: eg., http://donar.umiacs.umd.edu/quadtree/points/kdtree.html