CMSC424: Database Design

Instructor: Amol Deshpande

amol@cs.umd.edu
Overview

User

select *
from R, S
where ...

Query Parser

Query Optimizer

Query Processor

R, B+Tree on R.a
S, Hash Index on S.a

Resolve the references, Syntax errors etc.
Converting the query to an internal format

relational algebra like

Find the best way to evaluate the query
Which index to use?
What join method to use?
...

Read the data from the files
Do the query processing
joins, selections, aggregates
...

Overview

query → parser and translator → relational-algebra expression

query output → evaluation engine → execution plan

data → statistics about data

parser and translator

optimizer
“Cost”

- Complicated to compute
  - Too many factors
  - Total resource consumptions vs response time
- We will focus on disk:
  - Number of I/Os not sufficient by itself
    - Number of seeks matters a lot... why?
  - $t_T$ – time to transfer one block
  - $t_S$ – time for one seek
  - Cost for $b$ block transfers plus $S$ seeks
    $$b \times t_T + S \times t_S$$
  - Measured in seconds
Query Processing

- Overview
- Selection operation
- Join operators
- Sorting
- Other operators
- Putting it all together…
Selection Operation

- select * from person where SSN = “123”

**Option 1: Sequential Scan**
- Read the relation start to end and look for “123”
  - Can always be used (not true for the other options)
- Cost?
  - Let $b_r = \text{Number of relation blocks}$
  - Then:
    - 1 seek and $b_r$ block transfers
  - So:
    - $t_S + b_r \times t_T \text{ sec}$
- Improvements:
  - If SSN is a key, then can stop when found
    - So on average, $b_r/2 \text{ blocks accessed}$
Selection Operation

- select * from person where SSN = “123”

- **Option 2 : Binary Search:**
  - Pre-condition:
    - *The relation is sorted on SSN*
    - *Selection condition is an equality*
      - E.g. can’t apply to “Name like ‘%424%’”
  - Do binary search
    - Cost of finding the *first* tuple that matches
      - \( [\log_2(b_r)] \times (t_T + t_S) \)
      - All I/Os are random, so need a seek for all
        - The last few are closeby, but we ignore such small effects
  - Not quite: What if 10000 tuples match the condition?
    - Incurs additional cost
Selection Operation

- select * from person where SSN = “123”

**Option 3 : Use Index**

- Pre-condition:
  - *An appropriate index must exist*

- Use the index
  - Find the first leaf page that contains the search key
  - Retrieve all the tuples that match by following the pointers
    - If primary index, the relation is sorted by the search key
      - Go to the relation and read blocks sequentially
    - If secondary index, must follow all pointers using the index
## Selection w/ B+-Tree Indexes

<table>
<thead>
<tr>
<th>Index Type</th>
<th>Cost of Finding the First Leaf</th>
<th>Cost of Retrieving the Tuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary index, candidate key, equality</td>
<td>( h_i \times (t_T + t_S) )</td>
<td>( 1 \times (t_T + t_S) )</td>
</tr>
<tr>
<td>Primary index, not a key, equality</td>
<td>( h_i \times (t_T + t_S) )</td>
<td>( 1 \times (t_T + t_S) + (b - 1) \times t_T )</td>
</tr>
<tr>
<td>Note: primary == sorted</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( b = \text{number of pages that contain the matches} )</td>
</tr>
<tr>
<td>Secondary index, candidate key, equality</td>
<td>( h_i \times (t_T + t_S) )</td>
<td>( 1 \times (t_T + t_S) )</td>
</tr>
<tr>
<td>Secondary index, not a key, equality</td>
<td>( h_i \times (t_T + t_S) )</td>
<td>( n \times (t_T + t_S) )</td>
</tr>
<tr>
<td>( n = \text{number of records that match} )</td>
<td></td>
<td>( \text{This can be bad} )</td>
</tr>
</tbody>
</table>

\( h_i = \text{height of the index} \)