Evaluation of Relational Operations

Yanlei Diao
Overview of Query Processing

Query Processor

Query Parser
Query Rewriter
Query Optimizer
Query Executor

Lock Manager
Access Methods
Buffer Manager
Log Manager

Transaction Manager

Disk Manager

Disk Space Manager

DB
Relational Operations

- We will consider how to implement
  - **Selection** ($\sigma$) Selects a subset of rows from relation.
  - **Join** ($\bowtie$) Allows us to combine two relations.
  - **Projection** ($\pi$) Deletes unwanted columns from relation.
  - **Union** ($\cup$) Tuples in either reln. 1 or reln. 2.
  - **Intersection** ($\cap$) Tuples in both reln. 1 and reln. 2.
  - **Set-difference** ($\neg$) Tuples in reln. 1, but not in reln. 2.

- **GROUP BY** and **Aggregation** (SUM, MIN, etc.)
  
  with cost estimation, which leads to **cost-based optimization**
For more details, see the textbook

Database Management Systems
3rd Edition
Ramakrishnan and Gehrke

Amazon:
- Buy new: $43-$147.09 (hardcover); paperback, $23; Kindle, rent options are also available…
Background on Algorithms

Introduction to Algorithms
3rd Edition
Thomas Cormen,
Charles E. Leiserson
Ronald L. Rivest
Clifford Stein

Amazon:
Buy new: $66.32(hardcover);
rent options are also available…
Outline

- Selection
- Sorting routine
- Join
- Projection
- Set operators
- Group By aggregation
Schema for Examples

Sailors (\textit{sid}: integer, \textit{sname}: string, \textit{rating}: integer, \textit{age}: real)
Reserves (\textit{sid}: integer, \textit{bid}: integer, \textit{day}: date, \textit{rname}: string)

\begin{itemize}
  \item Sailors:
    \begin{itemize}
      \item Each tuple is 50 bytes long,
      \item 80 tuples per page,
      \item 500 pages.
    \end{itemize}
  \item Reserves:
    \begin{itemize}
      \item Each tuple is 40 bytes long,
      \item 100 tuples per page,
      \item 1000 pages.
    \end{itemize}
  \item Cost metric: \# I/Os
\end{itemize}
Using an Index for Selections

```
SELECT  *
FROM    Sailors S
WHERE   S.rating > 8
```

- Cost of selection includes:
  1) top down search in the index
  2) scan the relevant leaf nodes
  3) retrieve records from file (could be large w/o clustering)
Using an Index for Selections

- Cost of selection includes:
  1) top down search in the index
  2) scan the relevant leaf nodes
  3) retrieve records from file (could be large w/o clustering)

- **Step 1) top down search:** = the height of the tree H (and buffer management), ≤ 3-4 I/Os

```
SELECT * 
FROM   Sailors S 
WHERE  S.rating > 8 
```
Cost Factors of Steps 2 and 3

- Cost of selection includes:
  1) top down search in the index
  2) scan the relevant leaf nodes
  3) retrieve records from file (could be large w/o clustering)

- Step 2 scanning leaf nodes: cost factors include
  - Num. of leaf nodes: Either is given, or can be estimated as follows: if a data entry is 1/5 of a tuple, there are 500 / 5 = 100 leaf nodes.
  - Reduction factor (% of qualifying tuples): rating > 8: 20% of tuples qualify
  - So, 100 * 20% = 20 I/Os.

```
SELECT * 
FROM   Sailors S 
WHERE  S.rating > 8
```
Cost Factors of Selection (contd.)

- **Step 3 retrieving records from file:** cost factors include
  - **Num. of qualifying tuples:** rating > 8: 20% of tuples, 500*20%=100 data pages, 80*100=8,000 tuples
  - **Clustering:** retrieving records from file ≈
    - **Clustered index:** 100 I/Os.
    - **Unclustered index:** worst case 1 I/O per tuple; 8,000 I/Os here!
    - **Unclustered index + Sorting or Bitmap on rid:** ≤ 500 I/Os.

  (a) Retrieve matching data entries; sort by page_id; retrieve records in order of page id
  (b) Bitmap Index Scan + Bitmap Heap Scan in PostgreSQL: replace sorting w. a bitmap on page ids; retrieve records from pages with the bit set to 1
General Selections

- Boolean combination of predicates using AND and OR.
  - Conjunctive Normal Form (CNF), e.g.,
    
    \[ \text{pred1 AND (pred3 OR pred4)} \]
    
    \[ (\text{pred1 OR pred2}) \text{ AND (pred3 OR pred4)} \]

- \textit{File scan} always works for general selections.

- \textit{Index scan} works when it matches a predicate that is a conjunct of CNF.
  - E.g., an index matching \textit{‘pred1’} can be used for
    
    \[ \text{pred1 AND (pred3 OR pred4)} \]
Outline

- Selection
- Sorting routine
- Join
- Projection
- Set operators
- Group By aggregation
Why Sort?

- Important utility in DBMS:
  - Request data in sorted order (e.g., ORDER BY)
    - e.g., find students in decreasing order of gpa
  - Sort-merge join algorithm involves sorting.
  - Eliminate duplicates in a collection of records (e.g., SELECT DISTINCT)
  - Sorting is first step in bulk loading B+ tree index.

- Problem: sort 1TB of data with 1GB of RAM
  - Limited memory \( \rightarrow \) key is to minimize \# I/Os!
  - Methodology for algorithm design: from simple to complex
What is the **minimum memory size** (in num. of buffer pages) needed to sort a large file on disk?
2-Way Sort: Requires 3 Buffer Pages

- Pass 1: Read a page, sort it, write it as a sorted subfile
  - only one buffer page (from the memory) is used

- Pass 2, 3, …, etc.: Merge two sorted subfiles
  - three buffer pages are used
Two-Way External Merge Sort

- **Divide and conquer:** sort subfiles (runs) and then merge

A file of N pages:
- Pass 1: N sorted runs of 1 page each
- Pass 2: N/2 sorted runs of 2 pages each
- Pass 3: N/4 sorted runs of 4 pages each
- ... 
- Pass P+1: 1 sorted run of $2^P$ pages

$2^P \geq N \implies P \geq \log_2 N$
Divide and conquer: sort subfiles (runs) and then merge

- Each pass, read + write $N$ pages in file $\rightarrow 2N$.
- Number of passes is: $\lceil \log_2 N \rceil + 1$
- So total cost is: $2N(\lceil \log_2 N \rceil + 1)$
General External Merge Sort

Given $B (>3)$ buffer pages. How can we utilize them?

- **Pass 1:** Use $B$ buffer pages. Produce $\lceil \frac{N}{B} \rceil$ sorted runs of $B$ pages each.
- **Pass 2, 3..., etc.:** Merge $B-1$ runs.
Cost of External Merge Sort

- E.g., with 5 (B) buffer pages, sort 108 (N) page file:

<table>
<thead>
<tr>
<th>Pass 1</th>
<th>[\lceil 108/5 \rceil = 22] sorted runs of 5 pages each (last run is only 3 pages)</th>
<th>[\lceil N/B \rceil] sorted runs of B pages each</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass 2</td>
<td>[\lceil 22/4 \rceil = 6] sorted runs of 20 pages each (last run is only 8 pages)</td>
<td>[\lceil N/B \rceil / (B-1)] sorted runs of B(B-1) pages each</td>
</tr>
<tr>
<td>Pass 3</td>
<td>2 sorted runs, 80 pages and 28 pages</td>
<td>[\lceil N/B \rceil / (B-1)^2] sorted runs of B(B-1)^2 pages</td>
</tr>
<tr>
<td>Pass 4</td>
<td>Sorted file of 108 pages</td>
<td>[\lceil N/B \rceil / (B-1)^3] sorted runs of B(B-1)^3 (\geq N) pages</td>
</tr>
</tbody>
</table>

- Number of passes = \(1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil\)
- Cost = \(2N \times (1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil)\)