Database Systems
CSE 414

Lectures 16 – 17:
Basics of Query Optimization and
Cost Estimation
(Ch. 15.{1,3,4,6,6} & 16.4-5)
Announcements

• WQ4 is due Friday 11pm
• HW3 is due next Tuesday 11pm
• Midterm is next Monday
Motivation

• To understand performance, need to understand a bit about how a DBMS works
  – my database application is too slow… why?
  – one of the queries is very slow… why?

• Under your direct control: index choice
  – understand how that affects query performance
Recap: Query Evaluation

1. **Parse & Check Query**
   - Translate query string into internal representation
   - Check syntax, access control, table names, etc.

2. **Decide how best to answer query: query optimization**
   - Logical plan → physical plan

3. **Query Execution**

4. **Return Results**

---

CSE 414 - Fall 2017
Query Optimizer Overview

• **Input**: Parsed & checked SQL
• **Output**: A good physical query plan
• **Basic query optimization algorithm**:  
  – Enumerate alternative plans (logical and physical)  
  – Compute estimated cost of each plan  
    • Compute number of I/Os  
    • Optionally take into account other resources  
  – Choose plan with lowest cost  
  – This is called cost-based optimization
Query Optimizer Overview

- There are exponentially many query plans
  - exponential in the size of the query
  - simple SFW with 3 joins does not have too many
- Optimizer will consider many, many of them
- Worth substantial cost to avoid bad plans
Rest of Today

• Cost of reading from disk

• Cost of single RA operators

• Cost of query plans
Cost of Reading Data From Disk
Cost Parameters

• Cost = Disk I/O + CPU + Network I/O
  – We will focus on Disk I/O

• Parameters:
  – $B(R)$ = # of blocks (i.e., pages) for relation $R$
  – $T(R)$ = # of tuples in relation $R$
  – $V(R, A)$ = # of distinct values of attribute $A$
    • When $A$ is a key, $V(R, A) = T(R)$
    • When $A$ is not a key, $V(R, A)$ can be anything < $T(R)$

• Where do these values come from?
  – DBMS collects statistics about data on disk
Selectivity Factors for Conditions

- **A = c**  
  \[ */ \sigma_{A=c}(R) */ \]  
  - Selectivity = \(1/V(R, A)\)

- **A < c**  
  \[ */ \sigma_{A<c}(R)*/ \]  
  - Selectivity = \((c - \text{Low}(R, A))/(\text{High}(R, A) - \text{Low}(R, A))\)

- **c1 < A < c2**  
  \[ */ \sigma_{c1<A<c2}(R)*/ \]  
  - Selectivity = \((c2 - c1)/(\text{High}(R, A) - \text{Low}(R, A))\)
Example: Selectivity of $\sigma_{A=c}(R)$

$T(R) = 100,000$
$V(R, A) = 20$

How many records are returned by $\sigma_{A=c}(R) = ?$

Answer: $X \times T(R)$, where $X$ = selectivity...
... $X = 1/V(R, A) = 1/20$

Number of records returned = $100,000/20 = 5,000$
Cost of Index-based Selection

- Sequential scan for relation R costs $B(R)$

- Index-based selection
  - Estimate selectivity factor $X$ (see previous slide)
  - Clustered index: $X*B(R)$
  - Unclustered index $X*T(R)$

Note: we are ignoring I/O cost for index pages
Example: Cost of $\sigma_{A=c}(R)$

- Example:
  
  $B(R) = 2000$
  $T(R) = 100,000$
  $V(R, A) = 20$

- Table scan: $B(R) = 2,000$ I/Os

- Index based selection:
  - If index is clustered: $B(R)/V(R, A) = 100$ I/Os
  - If index is unclustered: $T(R)/V(R, A) = 5,000$ I/Os

Lesson: Don’t build unclustered indexes when $V(R, A)$ is small!
Cost of Executing Operators (Focus on Joins)
Outline

• **Join operator algorithms**
  – One-pass algorithms (Sec. 15.2 and 15.3)
  – Index-based algorithms (Sec 15.6)

• **Note about readings:**
  – In class, we discuss only algorithms for joins
  – Other operators are easier: read the book
Join Algorithms

- Hash join
- Nested loop join
- Sort-merge join
Hash Join

Hash join:  \( R \bowtie S \)
- Scan \( R \), build buckets in main memory
- Then scan \( S \) and join
- Cost: \( B(R) + B(S) \)

- One-pass algorithm when \( B(R) \leq M \) (memory size)
  - more disk access also when \( B(R) > M \)
Hash Join Example

Patient(pid, name, address)

Insurance(pid, provider, policy_nb)

Patient $\bowtie$ Insurance

Two tuples per page
Hash Join Example

Patient ▷ Insurance

Disk

Patient | Insurance
---|---
1 | 2
3 | 4
9 | 6
8 | 5

2 | 4
4 | 3
2 | 8
8 | 9

6 | 6
1 | 3

Showing pid only

Memory M = 21 pages

Large enough

This is one page with two tuples

CSE 414 - Fall 2017
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Memory M = 21 pages

Hash h: pid % 5

Input buffer
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages

Hash h: pid % 5

Disk

Patient | Insurance
---|---
1 2 | 2 4
3 4 | 4 3
9 6 | 2 8
8 5 | 8 9

Input buffer

Output buffer

Write to disk
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages
Hash h: pid % 5

Disk

Patient | Insurance
---|---
1 | 2
3 | 4
9 | 6
8 | 5

Input buffer

Output buffer

CSE 414 - Fall 2017
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Keep going until read all of Insurance

Cost: B(R) + B(S)
Nested Loop Joins

- Tuple-based nested loop \( R \bowtie S \)
- \( R \) is the outer relation, \( S \) is the inner relation

```plaintext
for each tuple \( t_1 \) in \( R \) do
    for each tuple \( t_2 \) in \( S \) do
        if \( t_1 \) and \( t_2 \) join then output \( (t_1, t_2) \)
```

- **Cost**: \( B(R) + T(R) \cdot B(S) \)
- Multiple-pass because \( S \) is read many times
Block-at-a-time Refinement

for each block of tuples r in R do
  for each block of tuples s in S do
    for all pairs of tuples $t_1$ in r, $t_2$ in s
      if $t_1$ and $t_2$ join then output ($t_1$, $t_2$)

• Cost: $B(R) + B(R) \times B(S)$
Block-at-a-time Refinement

Disk

Patient

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Insurance

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Input buffer for Patient

Input buffer for Insurance

Output buffer
Block-at-a-time Refinement

Disk

Patient | Insurance
------- | -------
1 2     | 2 4 6 6
3 4     | 4 3 1 3
9 6     | 2 8
8 5     | 8 9

1 2     Input buffer for Patient
4 3     Input buffer for Insurance

Output buffer
Page-at-a-time Refinement

Disk

Patient

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Insurance

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

Input buffer for Patient

Input buffer for Insurance

Keep going until read all of Insurance

Output buffer

CSE 414 - Fall 2017
Block-at-a-time Refinement

Input buffer for Patient

Input buffer for Insurance

Keep going until read all of Insurance

Then repeat for next page of Patient… until end of Patient

Cost: $B(R) + B(R)B(S)$
Block-Nested-Loop Refinement

for each group of M-1 pages r in R do
    for each page of tuples s in S do
        for all pairs of tuples t₁ in r, t₂ in s
            if t₁ and t₂ join then output (t₁,t₂)

• Cost: \( B(R) + \frac{B(R)B(S)}{(M-1)} \)

What is the Cost?
Sort-Merge Join

Sort-merge join: \( R \bowtie S \)
- Scan \( R \) and sort in main memory
- Scan \( S \) and sort in main memory
- Merge \( R \) and \( S \)

- Cost: \( B(R) + B(S) \)
- One pass algorithm when \( B(S) + B(R) \leq M \)
- Typically, this is NOT a one pass algorithm
Sort-Merge Join Example

Step 1: Scan Patient and \textcolor{red}{sort} in memory

Memory M = 21 pages
Sort-Merge Join Example

Step 2: Scan Insurance and sort in memory

Memory $M = 21$ pages

Disk

Patient  Insurance

\[
\begin{array}{c}
\text{1} \ 2 \\
\text{3} \ 4 \\
\text{9} \ 6 \\
\text{8} \ 5 \\
\end{array}
\quad
\begin{array}{c}
\text{2} \ 4 \\
\text{4} \ 3 \\
\text{2} \ 8 \\
\text{8} \ 9 \\
\end{array}
\quad
\begin{array}{c}
\text{6} \ 6 \\
\text{1} \ 3 \\
\text{1} \ 3 \\
\text{8} \ 9 \\
\end{array}
\]
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Disk

Patient  Insurance

1 2  
3 4  
9 6  
8 5  

Output buffer

1 2 3 4 5 6 8 9

1 2 2 3 3 4 4 6

6 8 8 9
Sort-Merge Join Example

Step 3: **Merge** Patient and Insurance

Memory $M = 21$ pages

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 4</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3</td>
</tr>
<tr>
<td>9 6</td>
<td>2 8</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Output buffer

<table>
<thead>
<tr>
<th>1 2</th>
<th>3 4</th>
<th>5 6</th>
<th>8 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 3</td>
<td>3 4</td>
<td>4 6</td>
</tr>
<tr>
<td>6 8</td>
<td>8 9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Using PK, so only one can match
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Disk

Patient  |  Insurance
-------  |  -------
1  2     |  2  4    |  6  6
3  4     |  4  3    |  1  3
9  6     |  2  8    
8  5     |  8  9    

Output buffer

CSE 414 - Fall 2017
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Keep going until end of first relation
Index Nested Loop Join

R \bowtie S

- Assume S has an index on the join attribute
- Iterate over R, for each tuple, fetch corresponding tuple(s) from S

- **Cost:**
  - If index on S is clustered: \( B(R) + T(R)B(S)/V(S, A) \)
  - If index on S is unclustered: \( B(R) + T(R)T(S)/V(S, A) \)
Cost of Query Plans
Physical Query Plan 1

\[ \sigma \text{ scity='Seattle' } \wedge \text{ sstate='WA' } \wedge \text{ pno=2} \]

\[ \pi \text{ sname} \]

Selection and project on-the-fly

\[ \text{No additional cost.} \]

\[ \text{Total cost of plan is thus cost of join:} \]
\[ = B(\text{Supplier}) + B(\text{Supplier}) \times B(\text{Supply}) \]
\[ = 100 + 100 \times 100 \]
\[ = 10,100 \text{ I/Os} \]

\[ \text{T(Supplier)} = 1000 \]
\[ \text{B(Supplier)} = 100 \]
\[ \text{V(Supplier, scity)} = 20 \]
\[ \text{M = 11} \]
\[ \text{T(Supply)} = 10,000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supplier, state)} = 10 \]
\[ \text{V(Supply, pno)} = 2,500 \]
Physical Query Plan 2

(On the fly) \[ \pi_{\text{sname}} \]

(Sort-merge join)

(Scan, write to T1)

(a) \( \sigma_{\text{scity}='Seattle' \land \text{sstate}='WA'} \)

(b) \( \sigma_{\text{pno}=2} \)

(Scan, write to T2)

Supplier (File scan)

Total cost

\[ = 100 + 100 \times \frac{1}{20} \times \frac{1}{10} \ (a) \]

\[ + 100 + 100 \times \frac{1}{2500} \ (b) \]

\[ + 2 \ (c) \]

\[ + 0 \ (d) \]

Total cost \( \approx 204 \) I/Os

\begin{align*}
\text{T(Supplier)} &= 1000 \\
\text{B(Supplier)} &= 100 \\
\text{V(Supplier, scity)} &= 20 \\
\text{M} &= 11 \\
\text{T(Supply)} &= 10,000 \\
\text{B(Supply)} &= 100 \\
\text{V(Supplier, state)} &= 10 \\
\text{V(Supply, pno)} &= 2,500
\end{align*}
Physical Query Plan 3

Total cost
≈ 5 I/Os