Database Systems
CSE 414

Lectures 11 – 12:
Basics of Query Optimization and Cost Estimation
(Ch. 15.(1,3,4,6,6) & 16.4-5)

Announcements

• HW3 is due Tuesday
• WQ4 is due Thursday
• Midterm on Friday
  – we’ll talk more about it on Monday
• Husky Football spring game tomorrow

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

SQL query

Parse & Check Query

Decide how best to answer query: query optimization

Logical plan → physical plan

Query Execution

Return Results

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

• There are exponentially many query plans
  – exponential in the size of the query
  – simple SFW with 3 joins has not 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

- $A = c$
  - Selectivity = $1/V(R, A)$
- $A < c$
  - Selectivity = $(c - \text{Low}(R, A))/(\text{High}(R, A) - \text{Low}(R, A))$
- $c_1 < A < c_2$
  - Selectivity = $(c_2 - c_1)/(\text{High}(R, A) - \text{Low}(R, A))$

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 $X = \text{selectivity} \ldots$

$X = 1/V(R, A) = 1/20$

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: $X \times B(R)$
  - Unclustered index $X \times T(R)$

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

- Example:
  
  \[
  \begin{align*}
  B(R) &= 2000 \\
  T(R) &= 100,000 \\
  V(R, A) &= 20
  \end{align*}
  \]

  \[\text{cost of } \sigma_{A=c}(R) = ?\]

- Table scan: $B(R) = 2,000$ I/Os
- Index based selection:
  - If index is clustered: $B(R)\sqrt{V(R, A)} = 100$ I/Os
  - If index is unclustered: $T(R)\sqrt{V(R, A)} = 5,000$ 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 $B(R) \leq M$
  - More disk access also when $B(R) > M$

Hash Join Example

Patient(pid, name, address)
 Insurance(pid, provider, policy_nb)
Patient $\bowtie$ Insurance
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Step 2: Scan Insurance and probe into hash table

Step 2: Scan Insurance and probe into hash table

Step 2: Scan Insurance and probe into hash table

Disk

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Disk

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Disk

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Disk

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Nested Loop Joins

- Tuple-based nested loop R \bowtie S
- R is the outer relation, S is the inner relation
  
  for each tuple t in R do
  for each tuple t_j in S do
  if t and t_j join then output (t, t_j)

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

\[
\text{Cost: } B(R) + B(R)B(S)
\]

Page-at-a-time Refinement

\[
\text{Cost: } B(R) + B(R)B(S)/(M-1)
\]

Block-Nested-Loop Refinement

\[
\text{Cost: } B(R) + B(R)B(S)
\]
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 and sort in memory

Disk

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

Memory \( M = 21 \) pages

Using PK, so only one can match
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

(On the fly) \( \pi_{\text{name}} \)
Selection and project on-the-fly
\( \rightarrow \) No additional cost.

(On the fly) \( \sigma_{\text{scity='Seattle'} \land \text{state='WA'} \land pno=2} \)
Total cost of plan is thus cost of join:
\( = B(Supplier) + B(Supplier) \cdot B(Supply) \)
\( = 100 \cdot 100 = 10,100 \text{ I/Os} \)

Physical Query Plan 2

(On the fly) \( \pi_{\text{name}} \)

(d) Total cost
\( = 100 + 100 \cdot 1/20 \cdot 1/10 \) (a)
\( + 100 + 100 \cdot 1/2500 \) (b)
\( + 2 \) (c)
\( + 0 \) (d)
Total cost = 204 I/Os
Physical Query Plan 3

(On the fly) (d) \( \pi_{\text{name}} \) \( \text{Total cost} = 1 \) (a) + 4 (b) + 0 (c) + 0 (d) Total cost = 5 I/Os

(On the fly) (c) \( \sigma_{\text{city}='Seattle' \land \text{state}='WA'} \)

(Use hash index) (b) 4 tuples (Index nested loop)

(a) \( \sigma_{\text{pno}=2} \) (Index on pno)

(b) \( \sigma_{\text{sno} = \text{sno}} \) (Index on sno)

Assume: clustered Clustering does not matter

\( T(\text{Supplier}) = 1000 \) \( B(\text{Supplier}) = 100 \) \( V(\text{Supplier}) = 20 \) \( V(\text{Supplier}) = 100 \) \( M = 11 \)