Introduction to Database Systems
CSE 414

Lecture 28: Intro to Query Optimization

Final Exam
• Thursday 6/7, 2:30-4:20pm
• Location: here
• Comprehensive exam
  • Covers all lectures, sections, web quizzes, HWs, and readings
  • Can bring 2 letter-size sheets of notes
    — Handwritten or printed
  • More info on course website
• Review session:
  — Sunday 6/3, 2:30-5pm, SMI 102

Big Picture
• How to choose the “best” query plan to run? (aka query optimization)
• To answer this question we need to understand:
  − Data organization on the disk
  − Index structures and how they are used in queries
  − A way to model query “costs”
    − Compute cost for each query operator
    − Compute cost for each physical plan

Review: Join Algorithms
• Nested loop join
• Hash join
• Sort-merge join

Hash Join
Hash join: \( R \bowtie S \)
• Scan R, build hash table in main memory
• Then scan S and join
• Cost: \( B(R) \times B(S) \)
• Which relation to build the hash table on?
• One-pass algorithm when \( B(R) \leq M \)
  − \( M \) = number of memory pages available
Hash Join Example

Patient(pid, name, address)
Insurance(pid, provider, policy_nb)

Patient \Join \text{Insurance}

Two tuples per page

Patient

<table>
<thead>
<tr>
<th>1</th>
<th>Bob</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Ela</td>
<td>Everett</td>
</tr>
<tr>
<td>3</td>
<td>Jim</td>
<td>Kent</td>
</tr>
<tr>
<td>4</td>
<td>Joe</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

Insurance

<table>
<thead>
<tr>
<th>2</th>
<th>'Blue'</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>'Prem'</td>
<td>432</td>
</tr>
<tr>
<td>3</td>
<td>'Prem'</td>
<td>343</td>
</tr>
<tr>
<td>3</td>
<td>'GrpH'</td>
<td>554</td>
</tr>
</tbody>
</table>

Step 1: Scan Patient and build hash table in memory

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

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

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages

Hash h: pid % 5

Input buffer

Output buffer

Write to disk or pass to next operator

Keep going until read all of Insurance

Cost: B(R) + B(S)

Showing pid only

This is one page with two tuples

Some large-enough #
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 sort in memory

- Disk
  - Patient
    - 1 2 3 4
    - 6 8
  - Insurance
    - 1 2 4 6
    - 8 9

**Step 2:** Scan Insurance and sort in memory

- Disk
  - Patient
    - 1 2 3 4
    - 6 8
  - Insurance
    - 1 2 4 6
    - 8 9

**Step 3:** Merge Patient and Insurance

- Disk
  - Patient
    - 1 2 3 4
    - 6 8
  - Insurance
    - 1 2 3 4
    - 6 8
  - Output buffer
    - 1

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 \( S \)

\[
\text{for } r \text{ in } R \\
\text{// use index to lookup} \\
\text{for } s' \text{ in } S \text{ that should be joined with } r \\
s = \text{fetch } S \text{ tuple pointed to by } s' \text{ from disk} \\
\text{output } (r, s)
\]

Cost:
- If index on \( S \) is clustered:
  \[ B(R) + T(R) \times (B(S) \times 1/V(S,a)) \]
- If index on \( S \) is unclustered:
  \[ B(R) + T(R) \times (T(S) \times 1/V(S,a)) \]

Review: Physical Query Plan 1

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

(On the fly) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

(Nested loop) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

Supplier (File scan) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

Supply (File scan) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

A physical query plan is a logical query plan annotated with physical implementation details.

Review: Physical Query Plan 2

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

(On the fly) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

(Hash join) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

Supplier (File scan) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

Supply (File scan) \[ \sigma_{\text{scity} = \text{Seattle} \text{ and sstate} = \text{WA} \text{ and pno} = 2} \]

Same logical query plan
Different physical plan
Review: Physical Query Plan 3

Different but equivalent logical query plan; different physical plan

Query Optimization: Overview

- Compute cost of each operator, which depends on:
  - Table statistics (# of tuples produced)
  - Algorithm used to implement each operator
- Cost of a physical plan = sum(each operator cost)
- Cost each plan and choose the one with lowest cost

Estimating Table Statistics
Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]

Logical Query Plan 2

\[ \pi_{\text{sname}} \\
\sigma_{\text{scity} = \text{Seattle}} \sigma_{\text{sstate} = \text{WA}} \]

\[ \text{SELECT sname FROM Supplier x, Supply y WHERE} \]
\[ x.\text{sid} = y.\text{sid} \]
\[ y.pno = 2 \]
\[ x.\text{scity} = \text{Seattle} \]
\[ x.\text{sstate} = \text{WA} \]

\[ \text{T(Supplier) = 1000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, state) = 10} \]

\[ \text{M=11} \]
Physical Plan 3

\[ \pi_{\text{sname}} \] \hspace{1cm} \sigma_{\text{pno} = 2} \hspace{1cm} \sigma_{\text{scity} = \text{Seattle} \land \text{sstate} = \text{WA}} \hspace{1cm} \text{Cost of \ Supply(pno) = 4} \hspace{1cm} \text{Cost of Index join = 4} \hspace{1cm} \text{Total cost: 8} \]

\[ T = 4 \]

\[ T = 4 \]

\[ M = 11 \]

\[ \text{Supply} \]

\[ \text{Supplier} \]

\[ \text{Unclustered index lookup} \]

\[ \text{Supply(pno)} \]

\[ \text{Cost of \ Supply(pno) = 4} \]

\[ \text{Cost of Index join = 4} \]

\[ \text{Total cost: 8} \]

\[ \text{Clustered index join} \]

\[ \text{Cost of \ Supply(pno) = 4} \]

\[ \text{Cost of Index join = 4} \]

\[ \text{Total cost: 8} \]

Query Optimizer Summary

- Input: A logical query plan
- Output: A good physical query plan
- Basic query optimization algorithm
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
  - Choose plan with lowest cost

- This is called cost-based optimization
  - More in CSE 444