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

Last topics this quarter!
Review: Join Algorithms

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

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

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 'Bob'</td>
<td>2 'Blue' 123</td>
</tr>
<tr>
<td>2 'Ela'</td>
<td>4 'Prem' 432</td>
</tr>
<tr>
<td>3 'Jill'</td>
<td>4 'Prem' 343</td>
</tr>
<tr>
<td>4 'Joe'</td>
<td>3 'GrpH' 554</td>
</tr>
</tbody>
</table>

Two tuples per page
# Hash Join Example

## Patient ⊙ Insurance

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

Disk

Memory $M = 21$ pages

Showing pid only

This is one page with two tuples

Some large-enough #
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Memory M = 21 pages

Hash h: pid % 5

= 0 = 1 = 2 = 3 = 4
5  1  6  2  3  8  4  9

Disk

Patient  Insurance

1  2  2  4  6  6
3  4  4  3  1  3
9  6  2  8
8  5  8  9
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages

Hash h: pid % 5

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>
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages

Hash h: pid % 5

Disk

Patient  Insurance

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

Input buffer

Output buffer

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

= 0  = 1  = 2  = 3  = 4
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory $M = 21$ pages

Hash $h$: pid % 5

<table>
<thead>
<tr>
<th>Input buffer</th>
<th>Output buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 3</td>
<td>4 4</td>
</tr>
</tbody>
</table>

Keep going until read all of Insurance

Cost: $B(R) + B(S)$
Sort-Merge Join
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

Memory M = 21 pages

Disk

Patient

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

Insurance

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

Step 2: Scan Insurance and sort in memory

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  |

Sorted!
Sort-Merge Join Example

Step 3: **Merge** Patient and Insurance

**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>

**Memory M = 21 pages**

```
1 2 3 4 5 6 8 9
1 2 2 3 3 4 4 6
6 8 8 9
```

Output buffer: 1 1

17
Sort-Merge Join Example

Step 3: **Merge** Patient and Insurance

**Disk**

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

**Memory M = 21 pages**

```
1 2 3 4 5 6 8 9
1 2 2 3 3 4 4 6
6 8 8 9
```

**Output buffer**

2 2

*Keep going until end of first relation*
Index Joins

R
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>98</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

S
c  d  e
3  43  7
...  ...  ...
9  24  9

...
Index Nested Loop Join

R ↘ S

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

```python
for r in R
    // use index to lookup
    for s' in S that should be joined with r
        s = fetch S tuple pointed to by s' from disk
    output (r, s)
```
Index Nested Loop Join

\[ R \bowtie S \]

```python
for r in R
    # use index to lookup
    for s' in S that should be joined with r
        s = fetch S tuple pointed to by s' from disk
    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:
Logical vs Physical Plans
A physical query plan is a logical query plan annotated with physical implementation details.

**SELECT** `sname`

**FROM** `Supplier x, Supply y`

**WHERE** `x.sid = y.sid` and `y.pno = 2` and `x.scity = 'Seattle'` and `x.sstate = 'WA'`
Review: Physical Query Plan 2

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

Same logical query plan
Different physical plan

\[ \sigma_{scity='Seattle' \text{ and } sstate='WA' \text{ and } pno=2} \]
\[ \pi_{sname} \]

(On the fly) [File scan]
(On the fly) [File scan]
(Hash join) [On the fly]

Supplier
(File scan)

Supply
(File scan)
Supplier($sid, sname, scity, sstate$)
Supply($sid, pno, quantity$)

Review: Physical Query Plan 3

(a) $\sigma_{scity='Seattle' \text{ and } sstate='WA'}$
(b) $\sigma_{pno=2}$ (Scan & write to T2)

(Scan & write to T1)

(Sort-merge join)

(On the fly)

\[ \Pi_{sname} \]

Different but equivalent logical query plan; different physical

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'

Scan & write to T1

Supplier (File scan)

Supply (File scan)
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 1

\[ \pi_{\text{sname}} \]
\[ \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \]
\[ \text{sid} = \text{sid} \]

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

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

\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]

\[ M=11 \]

Supplier \((\text{sid}, \text{sname}, \text{scity}, \text{sstate})\)
Supply \((\text{sid}, \text{pno}, \text{quantity})\)
Logical Query Plan 1

\[ \pi_{\text{sname}} \sigma_{pno=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \]

\[ T = 10000 \]

\[ M = 11 \]

\[
\begin{align*}
T(\text{Supply}) &= 10000 \\
B(\text{Supply}) &= 100 \\
V(\text{Supply}, pno) &= 2500
\end{align*}
\]

\[
\begin{align*}
T(\text{Supplier}) &= 1000 \\
B(\text{Supplier}) &= 100 \\
V(\text{Supplier}, \text{scity}) &= 20 \\
V(\text{Supplier}, \text{sstate}) &= 10
\end{align*}
\]

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
  and y.pno = 2
  and x.scity = 'Seattle'
  and x.sstate = 'WA'
Logical Query Plan 1

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

\[ T(Supply) = 10000 \]
\[ B(Supply) = 100 \]
\[ V(Supply, \text{pno}) = 2500 \]

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

\[ M = 11 \]
Logical Query Plan 2

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

T(Supplier) = 10000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11
Logical Query Plan 2

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

\( T(\text{Supply}) = 10000 \)
\( B(\text{Supply}) = 100 \)
\( V(\text{Supply, pno}) = 2500 \)

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

\( M = 11 \)
Logical Query Plan 2

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
  and y.pno = 2
  and x.scity = 'Seattle'
  and x.sstate = 'WA'

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

M=11

Why?
Logical Query Plan 2

\[
\begin{align*}
\pi_{\text{sname}} & \quad T = 4 \\
\sigma_{\text{pno}=2} & \quad T = 4 \\
\text{Supply} & \\
\sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA}} & \quad T = 5 \\
\text{Supplier} & \\
\end{align*}
\]

\[
\begin{align*}
T(\text{Supplier}) &= 10000 \\
B(\text{Supplier}) &= 100 \\
V(\text{Supplier}, \text{scity}) &= 20 \\
V(\text{Supplier}, \text{sstate}) &= 10 \\
M &= 11 \\
\end{align*}
\]
Logical Query Plan 2

\[
\begin{align*}
\text{SELECT} & \quad \text{sname from} \\
\text{FROM} & \quad \text{Supplier x, Supply y} \\
\text{WHERE} & \quad x.\text{sid} = y.\text{sid} \quad \text{and} \quad y.\text{pno} = 2 \\
& \quad \text{and} \quad x.\text{scity} = \text{Seattle} \quad \text{and} \quad x.\text{sstate} = \text{WA}.
\end{align*}
\]

**Wrong! Why?**

**Different estimate 😞**
Computing Plan Costs
Supplier\((\text{sid}, \text{sname}, \text{scity}, \text{sstate})\)
Supply\((\text{sid}, \text{pno}, \text{quantity})\)

Physical Plan 1

Block nested loop join

\[ \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \]

\( \pi_{\text{sname}} \)

\( T < 1 \)

\( T = 10000 \)

Supply

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

scan

Supplier

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

M=11

Total cost:

\[ 100/10 \times 100 = 1000 \]
Physical Plan 1

\[ \Pi_{\text{sname}} \]

\[ \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \]

\[ \text{T} = 10000 \]

Block nested loop join

Supply

Scan

\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]

Supplier

Scan

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

\[ \text{M=11} \]

Total cost: 100 + 100 * 100/10 = 1100

Scan

\[ \text{T} < 1 \]

\[ \text{M=11} \]
Physical Plan 2

Cost of Supply(pno) = 4
Cost of Supplier(scity) = 50
Total cost: 54

T(Supplier) = 10000
B(Supplier) = 100
V(Supplier, pno) = 2500

T(Supply) = 1000
B(Supply) = 100
V(Supply, pno) = 2500

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, sstate) = 10

M=11
Physical Plan 2

\[ \pi_{\text{sn}} = \pi_{\text{sid}} \]
\[ \sigma_{\text{sstate} = 'WA'} \]
\[ \pi_{\text{sname}} \]

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

\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]

**Total cost:**

**Main memory join**

\[ \text{Cost of Supply(pno)} = 4 \]
\[ \text{Cost of Supplier(scity)} = 50 \]

**Unclustered index lookup**

\[ \text{Unclustered index lookup \ Supplier(scity)} \]

\[ \text{Unclustered index lookup \ Supply(pno)} \]
Physical Plan 2

\[ \text{Physical Plan 2} \]

\[ \pi_{\text{name}} \]

\[ T = 4 \]

\[ \text{sid = sid} \]

\[ T = 4 \]

Main memory join

\[ \sigma_{\text{pno}=2} \]

\[ \text{Supply} \]

\[ T(\text{Supply}) = 10000 \]
\[ B(\text{Supply}) = 100 \]
\[ V(\text{Supply, pno}) = 2500 \]

\[ \sigma_{\text{sstate}='WA'} \]

\[ \text{Supplier} \]

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

Cost of Supply(pno) = 4
Cost of Supplier(scity) = 50
Total cost: 54

Unclustered index lookup
Supply(pno)

Unclustered index lookup
Supplier(scity)

M = 11
**Physical Plan 3**

```
∗
\[ \pi_{\text{sname}} \]
\[ \sigma_{\text{scity}='Seattle' \land \text{sstate}='WA'} \]
\[ \sigma_{\text{pno}=2} \]
\[ \text{sid} = \text{sid} \]
```

**Cost of Index join = 4**

**Unclustered index lookup**

**Supply**

```
T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500
```

**Clustered Index join**

```
\[ \sigma_{\text{scity}='Seattle' \land \text{sstate}='WA'} \]
```

**Total cost:** 8

**M=11**
Physical Plan 3

\[ \pi_{\text{sname}} \]

\[ \sigma_{\text{scity}='Seattle' \land \text{sstate}='WA'} \]

\[ \text{sid} = \text{sid} \]

\[ \sigma_{\text{pno}=2} \]

Unclustered index lookup
Supply\( (\text{pno}) \)

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

Clustered Index join

Cost of Supply\( (\text{pno})\) = 4
Cost of Index join = 4
Total cost: 8

\[ T(\text{Supplier}) = 1000 \]
\[ B(\text{Supplier}) = 100 \]
\[ V(\text{Supplier, scity}) = 20 \]
\[ V(\text{Supplier, state}) = 10 \]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Plan 3

Cost of Supply(pno) = 4
Cost of Index join = 4
Total cost: 8

M = 11

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

Unclustered index lookup
Supply(pno)

σ_{\text{pno}=2} \rightarrow \text{Clustered Index join}

σ_{\text{scity}='Seattle' \land \text{sstate}='WA'} \rightarrow \text{sid = sid}

Π_{\text{sname}}

T = 4

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