CSE 344

FEBRUARY 21ST – COST ESTIMATION
ADMINISTRIVIA

- HW5 Due Tonight (11:30)
- OQ5 Due Friday (11:00)
- HW6 Due next Wednesday (Feb 28)
- HW7 Out Friday
  - Entity Relations
  - Due TBD
- HW8 Out Monday
  - Due Mar 9th
BASIC INDEX SELECTION GUIDELINES

Consider queries in workload in order of importance

Consider relations accessed by query
  • No point indexing other relations

Look at WHERE clause for possible search key

Try to choose indexes that speed-up multiple queries
SELECT * 
FROM R 
WHERE R.K>? and R.K<?
COST PARAMETERS

Cost = I/O + CPU + Network BW

• We will focus on I/O in this class

Parameters (a.k.a. statistics):

• \( B(R) \) = # of blocks (i.e., pages) for relation R
• \( T(R) \) = # of tuples in relation R
• \( V(R, a) \) = # of distinct values of attribute a

DBMS collects statistics about base tables
must infer them for intermediate results

When a is a key, \( V(R,a) = T(R) \)
When a is not a key, \( V(R,a) \) can be anything \( \leq T(R) \)
SELECTIVITY FACTORS FOR CONDITIONS

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

\( A < c \)  
\[ \sigma_{A<c}(R) \]  
- Selectivity = \( \frac{c - \min(R,A)}{\max(R,A) - \min(R,A)} \)

\( c_1 < A < c_2 \)  
\[ \sigma_{c_1<A<c_2}(R) \]  
- Selectivity = \( \frac{c_2 - c_1}{\max(R,A) - \min(R,A)} \)
COST OF READING DATA FROM DISK

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

Index-based selection

- Estimate selectivity factor $f$ (see previous slide)
- Clustered index: $f*B(R)$
- Unclustered index $f*T(R)$

Note: we ignore I/O cost for index pages
INDEX BASED SELECTION

Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

cost of $\sigma_{a=v}(R)$ = ?

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

Index based selection:

- If index is clustered: $B(R) \times 1/V(R,a) = 100$ I/Os
- If index is unclustered: $T(R) \times 1/V(R,a) = 5,000$ I/Os

Lesson: Don’t build unclustered indexes when $V(R,a)$ is small!
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: \(B(R) + B(S)\)
Which relation to build the hash table on?
HASH JOIN

Hash join:  \( R \bowtie S \)
Scan R, build buckets 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 $\bowtie$ Insurance

Two tuples per page
**HASH JOIN EXAMPLE**

**Patient** \(\times\) **Insurance**

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

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

**Disk**

<table>
<thead>
<tr>
<th>Memory M = 21 pages</th>
</tr>
</thead>
</table>

Showing pid only

Some large-enough #

This is one page with two tuples
HASH JOIN EXAMPLE

Step 1: Scan Patient and build hash table in memory
Can be done in method open()

Memory M = 21 pages
Hash h: pid % 5

Input buffer
HASH JOIN EXAMPLE

Step 2: Scan Insurance and probe into hash table
Done during calls to next()

Memory M = 21 pages
Hash h: pid % 5

Write to disk or pass to next operator
HASH JOIN EXAMPLE

Step 2: Scan Insurance and probe into hash table
Done during calls to next()

Memory M = 21 pages
Hash h: pid % 5

Disk

Patient Insurance

<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>6 6</td>
</tr>
<tr>
<td>9 6</td>
<td>1 3</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Input buffer

Output buffer
HASH JOIN EXAMPLE

Step 2: Scan Insurance and probe into hash table
Done during calls to next()

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 $(t_1,t_2)$
```

What is the Cost?
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 since $S$ is read many times
for each page of tuples 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) + B(R)B(S)
PAGE-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

Input buffer for Patient
1  2
2  4

Input buffer for Insurance
2  2

Output buffer
PAGE-AT-A-TIME REFINEMENT

Disk

Patient  Insurance

Input buffer for Patient
1 2

Input buffer for Insurance
4 3

Output buffer

1 2

4 3
PAGE-AT-A-TIME REFINEMENT

Disk

Patient  Insurance

1 2
3 4
9 6
8 5

Input buffer for Patient

1 2

Input buffer for Insurance

2 8

Keep going until read all of Insurance

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

Output buffer

2 2

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) + B(R)B(S)/(M-1)
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  Insurance
1  2    2  4    6  6
3  4    4  3    1  3
9  6    2  8
8  5    8  9
**SORT-MERGE JOIN EXAMPLE**

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

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

Memory $M = 21$ pages

<table>
<thead>
<tr>
<th>1 2 3 4 5 6 8 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 2 3 3 4 4 6</td>
</tr>
<tr>
<td>6 8 8 9</td>
</tr>
</tbody>
</table>
**SORT-MERGE JOIN EXAMPLE**

Step 3: **Merge** Patient and Insurance

<table>
<thead>
<tr>
<th>Disk</th>
<th>Memory M = 21 pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>Insurance</td>
</tr>
<tr>
<td>1 2</td>
<td>2 4 6 6</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3 1 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: |

1 2 3 4 5 6 8 9

4 6

6 8 8 9

1 1
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:
  \[ B(R) + T(R) \times (B(S) \times \frac{1}{V(S,a)}) \]
- If index on S is unclustered:
  \[ B(R) + T(R) \times (T(S) \times \frac{1}{V(S,a)}) \]
LOGICAL QUERY PLAN 1

\[ \pi_{\text{sname}} \left( \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'}(\text{Supplier}) \right) \]

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

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

\[ M=11 \]

\[ T(\text{Supply}) = 10000 \\
B(\text{Supply}) = 100 \\
V(\text{Supply, pno}) = 2500 \]
LOGICAL QUERY PLAN 1

```
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, sstate) = 10

M=11

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

Suppliers(sid, sname, scity, sstate)
Supplies(sid, pno, quantity)
LOGICAL QUERY PLAN 1

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

\[
\begin{align*}
T & = 10000 \\
B & = 100 \\
V & = 2500
\end{align*}
\]

\[
\begin{align*}
T & < 1 \\
M & = 11
\end{align*}
\]
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
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(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500
```

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

`M=11`
```
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(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500
```

```
T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, state) = 10
```
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(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500

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

M=11

Very wrong! Why?
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?

Different estimate 😞

Very wrong!

Why?
**PHYSICAL PLAN 1**

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

\[ T = 10000 \]

\[ T < 1 \]

\[ \text{Block nested loop join} \]

\[ \text{Scan} \]

\[ \text{Scan} \]

\[ \text{Total cost:} \]

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

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

\[ M = 11 \]
PHYSICAL PLAN 1

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

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

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

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

Total cost: \( 100 + 100 \times 100 / 10 = 1100 \)
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

**PHYSICAL PLAN 2**

\[
\text{T}_{\text{sname}} = 4
\]

\[
\text{T} = 4
\]

\[
\text{id} = \text{id}
\]

\[
\text{T} = 4
\]

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

\[
\text{Supply}
\]

\[
\text{Main memory join}
\]

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

\[
\text{T} = 5
\]

\[
\sigma_{\text{scity}='Seattle'}
\]

\[
\text{T} = 50
\]

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

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

\[
\text{Total cost: 54}
\]

\[
\text{T(Supply)} = 10000
\]

\[
\text{B(Supply)} = 100
\]

\[
\text{V(Supply, pno)} = 2500
\]

\[
\text{T(Supplier)} = 1000
\]

\[
\text{B(Supplier)} = 100
\]

\[
\text{V(Supplier, scity)} = 20
\]

\[
\text{V(Supplier, state)} = 10
\]

\[
\text{M} = 11
\]
\[ \text{Supplier}(\text{sid, sname, scity, sstate}) \]
\[ \text{Supply}(\text{sid, pno, quantity}) \]

**PHYSICAL PLAN 2**

\[ \pi_{\text{sname}} \]
\[ \text{T} = 4 \]
\[ \text{sid} = \text{sid} \]
\[ \text{T} = 4 \]

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

Main memory join
\[ \sigma_{\text{pno}=2} \]

\[ \text{T} = 4 \]

\[ \sigma_{\text{sstate} = 'WA'} \]
\[ \text{T} = 5 \]

\[ \sigma_{\text{scity} = 'Seattle'} \]
\[ \text{T} = 50 \]

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

Cost of \( \text{Supply}(\text{pno}) \) = 4
Cost of \( \text{Supplier}(\text{scity}) \) = 50
Total cost: 54

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

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

\[ M = 11 \]
PHYSICAL PLAN 2

\[\pi_{\text{sname}} (\sigma_{\text{sstate} = \text{WA}} (\text{Supplier} = 1000)) = 100\]
\[B(\text{Supplier}) = 100\]
\[V(\text{Supplier, scity}) = 20\]
\[V(\text{Supplier, state}) = 10\]

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

Cost of \text{Supply(pno)} = 4
Cost of \text{Supplier(scity)} = 50
Total cost: 54

\[\text{Main memory join}\]
\[\sigma_{\text{pno} = 2}\]
\[\sigma_{\text{state} = \text{WA}}\]
\[\sigma_{\text{scity} = \text{Seattle}}\]

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

M=11
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

**PHYSICAL PLAN 3**

\[ \pi_{sname}(\text{Supplier}) = 1000 \]
\[ \text{B(Supplier)} = 100 \]
\[ \text{V(Supplier, scity)} = 20 \]
\[ \text{V(Supplier, state)} = 10 \]
\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]

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

\[ \sigma_{\text{pno}=2} \]
\[ \text{sid} = \text{sid} \]
\[ \text{Clustered Index join} \]

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

\[ \text{Unclustered index lookup} \]

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

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

\[ \text{M=11} \]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

**PHYSICAL PLAN 3**

\[ \text{T}_{\text{sname}} \]
\[ \text{σ}_{\text{scity}=\text{'Seattle'} \land \text{sstate}=\text{'WA'}} \]
\[ \text{T} = 4 \]
\[ \text{Clusted Index join} \]
\[ \text{σ}_{\text{pno}=2} \]
\[ \text{Supply} \]

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

Clustered Index join

\[ \text{σ}_{\text{scity}=\text{'Seattle'} \land \text{sstate}=\text{'WA'}} \]
\[ \text{σ}_{\text{pno}=2} \]
\[ \text{sid} = \text{sid} \]
\[ \text{T} = 4 \]

Unclustered index lookup
Supply(pno)

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

\[ \text{M} = 11 \]

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

PHYSICAL PLAN 3

\[ \pi_{\text{sname}}(\text{Supplier}) = 1000 \]
\[ \text{B(Supplier)} = 100 \]
\[ \text{V(Supplier, scity)} = 20 \]
\[ \text{V(Supplier, state)} = 10 \]
\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]

\[ \sigma_{\text{scity}=\text{\textquote{Seattle}}} \land \text{sstate}=\text{\textquote{WA}}(\text{Supplier}) \]
\[ \text{Cost of Supply(pno)} = 4 \]
\[ \text{Cost of Index join} = 4 \]
\[ \text{Total cost: 8} \]
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