Introduction to Database Systems
CSE 414

Lecture 17:
Basics of Query Optimization and Query Cost Estimation
Announcements

• Midterm will be released by end of day today
• Need to start one HW6 step NOW:
  – Need to make an AWS account, can use existing Amazon account
  – Click on application button under Students and fill out form with your @uw.edu email
  – Will then be sent email for verification, must click to verify your email address
Two typical kinds of queries

- Point queries
  - What data structure should be used for index?

- Range queries
  - What data structure should be used for index?

SELECT * FROM Movie WHERE year = ?

SELECT * FROM Movie WHERE year >= ? AND year <= ?
Choosing Index is Not Enough

• To estimate the cost of a query plan, we still need to consider other factors:
  – How each operator is implemented
  – The cost of each operator
  – Let’s start with the basics
Cost of Reading Data From Disk
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
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

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

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

• DBMS collects statistics about base tables
  must infer them for intermediate results
Selectivity Factors for Conditions

How many tuples would this select:

SELECT *
FROM One_year
WHERE did = 32

1 tuple (out of 365)
Selectivity Factors for Conditions

How many tuples would this select:

SELECT *
FROM One_year
WHERE month = Jan

31 tuples (out of 365)
This is roughly 1/12 of the tuples, because 12 distinct values equally distributed.
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 \( a \)

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

• DBMS collects statistics about base tables
  must infer them for intermediate results
Selectivity Factors for Conditions

- **A = c**  
  \[
  \sigma_A = c \quad (R) 
  \]  
  - Selectivity \( f = \frac{1}{V(R,A)} \)

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

- **c1 < A < c2**  
  \[
  \sigma_{c1 < A < c2} = c \quad (R) 
  \]  
  - Selectivity \( f = \frac{c2 - c1}{\max(R, A) - \min(R, A)} \)

- **Cond1 \land Cond2 \land Cond3 \land \ldots**  
  - Selectivity = \( f1 \times f2 \times f3 \times \ldots \) (assumes independence)
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

- Table scan:
- Index based selection:

\[
\text{cost of } \sigma_{a=v}(R) = ?
\]
Index Based Selection

• Example:
  \[ B(R) = 2000 \]
  \[ T(R) = 100,000 \]
  \[ V(R, a) = 20 \]

  \[ \text{cost of } \sigma_{a=v}(R) = {?} \]

• Table scan: \( B(R) = 2,000 \) I/Os

• Index based selection:
Index Based Selection

Example:

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

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

Index based selection:

- If index is clustered:

- If index is unclustered:

\[
\text{cost of } \sigma_{a=v}(R) = ?
\]
Index Based Selection

- Example:

<table>
<thead>
<tr>
<th>B(R) = 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(R) = 100,000</td>
</tr>
<tr>
<td>V(R, a) = 20</td>
</tr>
</tbody>
</table>

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

- Index based selection:
  - If index is clustered: \( B(R) \times \frac{1}{V(R,a)} = 100 \) I/Os
    
    Why: we know we can scan a full block to get the desired range
  - If index is unclustered:

Cost of \( \sigma_{a=v}(R) = ? \)
Index Based Selection

Example:

- Table scan: $B(R) = 2000$ I/Os
- Index based selection:
  - If index is clustered: $B(R) \times \frac{1}{V(R,a)} = 100$ I/Os
  
  **Why:** we know we can scan a full block to get the desired range
  
  - If index is unclustered: $T(R) \times \frac{1}{V(R,a)} = 5,000$ I/Os

<table>
<thead>
<tr>
<th>B(R) = 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(R) = 100,000</td>
</tr>
<tr>
<td>V(R, a) = 20</td>
</tr>
</tbody>
</table>

**cost of $\sigma_{a=v}(R)$ = ?**
Index Based Selection

• Example:
  \[
  B(R) = 2000 \\
  T(R) = 100,000 \\
  V(R, a) = 20
  \]

  \[
  \text{cost of } \sigma_{a=v}(R) = ?
  \]

• 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
    Why: we know we can scan a full block to get the desired range
  – 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!
**SELECT** *, *
**FROM** R
**WHERE** R.K>?? and R.K<??
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
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 $B(R) \leq M$
  - $M = \text{number of memory pages available}$
Hash Join Example

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

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

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

Two tuples per page
Hash Join Example

Patient $\bowtie$ Insurance

Memory M = 21 pages

Showing pid only

Disk

Patient Insurance

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

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

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

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 $h: \text{pid} \% 5$

<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>

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

Disk

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

Input buffer

Output buffer

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

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

<table>
<thead>
<tr>
<th>Input buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 4</td>
</tr>
</tbody>
</table>

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

```
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

\begin{verbatim}
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 ($t_1,t_2$)
\end{verbatim}

• Cost: $B(R) + T(R) \cdot B(S)$
• Multiple-pass since $S$ is read many times

What is the Cost?
Page-at-a-time Refinement

\[
\text{for each page of tuples } r \text{ in } R \text{ do} \\
\text{for each page of tuples } s \text{ in } S \text{ do} \\
\text{for all pairs of tuples } t_1 \text{ in } r, t_2 \text{ in } s \\
\text{if } t_1 \text{ and } t_2 \text{ join then output } (t_1, t_2)
\]

- Cost: $B(R) + B(R)B(S)$
Page-at-a-time Refinement

Disk

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

Input buffer for Patient
1 2
2 4

Input buffer for Insurance

Output buffer
2 2
Page-at-a-time Refinement

- Disk
  - Patient: 12, 34, 96, 85
  - Insurance: 24, 66, 43, 13, 28, 89

- Input buffer for Patient: 12
- Input buffer for Insurance: 43
- 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>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

Cost: \( B(R) + B(R)B(S) \)

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
INDEX JOINS
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:
    \[ B(R) + T(R) \times (B(S) \times 1/V(S,a)) \]
  - If index on S is unclustered:
    \[ B(R) + T(R) \times (T(S) \times 1/V(S,a)) \]
Index Nested Loop Join

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

Still have to scan in R

Why is the multiplier term T(R)?

What does 1/V(S,a) represent?

T(R) must be used because we cannot assume that a whole block of R (B(R)) will have the same attribute to join on, and thus use the same index access on S for.

1/V(S,a) represents the nature of the B+ Tree index. We are only scanning as much as we need. Note that the performance of the index join will decrease as V decreases.
Index Nested Loop Join

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

Why did this change from \( B(R) \) to \( T(R) \)?

Remember that tuples are stored on contiguous blocks. In a clustered index from before we know we can scan a single chunk of the disk to get the entire desired range. In an unclustered index we no longer can assume contiguous access. Thus we estimate that every tuple needs its own I/O operation.
SELECT sname 
FROM Supplier x, Supply y 
WHERE x.sid = y.sid 
    and y.pno = 2 
    and x.scity = 'Seattle' 
    and x.sstate = 'WA'

GENERATING QUERY PLANS
(REVIEW)
Review: Logical vs Physical Plans

• Logical plans:
  – Created by the parser from the input SQL text
  – Expressed as a relational algebra tree
  – Each SQL query has many possible logical plans

• Physical plans:
  – Goal is to choose an efficient implementation for each operator in the RA tree
  – Each logical plan has many possible physical plans
Relational algebra expression is also called the “logical query plan”
Review: Physical Query Plan 1

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

Same logical query plan
Different physical plan

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

• Compute cost of each operator
  – This depends on:
    • Table statistics (# of tuples etc)
    • Algorithm used

• Cost of a physical plan = sum(each operator cost)

• Cost each plan and choose the one with lowest cost
Cost of Query Plans
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Logical Query Plan 1

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

\[ \Pi_{sname} \]

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'
Logical Query Plan 1

### Query Plan

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

### Tables

- **Supplier**
  - `sid`, `sname`, `scity`, `sstate`
  - `T(Supplier) = 1000`
  - `B(Supplier) = 100`
  - `V(Supplier, scity) = 20`
  - `V(Supplier, sstate) = 10`

- **Supply**
  - `sid`, `pno`, `quantity`
  - `T(Supply) = 10000`
  - `B(Supply) = 100`
  - `V(Supply, pno) = 2500`
Logical Query Plan 1

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

\[
\begin{align*}
\text{T(Supplier)} &= 10000 \\
\text{B(Supplier)} &= 100 \\
\text{V(Supplier, scity)} &= 20 \\
\text{V(Supplier, state)} &= 10
\end{align*}
\]

\[
\begin{align*}
\text{T(Supply)} &= 10000 \\
\text{B(Supply)} &= 100 \\
\text{V(Supply, pno)} &= 2500
\end{align*}
\]
Logical Query Plan 2

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

\[
\begin{align*}
\text{T(Supply)} &= 10000 \\
\text{B(Supply)} &= 100 \\
\text{V(Supply, pno)} &= 2500 \\
\text{T(Supplier)} &= 1000 \\
\text{B(Supplier)} &= 100 \\
\text{V(Supplier, scity)} &= 20 \\
\text{V(Supplier, state)} &= 10
\end{align*}
\]
Logical Query Plan 2

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

\[
\begin{align*}
T(\text{Supplier}) &= 1000 \\
B(\text{Supplier}) &= 100 \\
V(\text{Supplier, scity}) &= 20 \\
V(\text{Supplier, sstate}) &= 10
\end{align*}
\]
Logical Query Plan 2

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

**Very wrong! Why?**

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

\[
\begin{align*}
T(\text{Supply}) & = 10000 \\
B(\text{Supply}) & = 100 \\
V(\text{Supply, pno}) & = 2500
\end{align*}
\]
Logical Query Plan 2

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

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

\[
\begin{align*}
T(\text{Supply}) &= 10000 \\
B(\text{Supply}) &= 100 \\
V(\text{Supply, pno}) &= 2500
\end{align*}
\]
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, sstate) = 10

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

Different estimate 😞

Very wrong! Why?
Physical Plan 1

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

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

Block nested loop join

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

Total cost:

\[ \frac{100}{10} \times 100 = 1000 \]

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Physical Plan 1

\[ \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \left( T \right) \]

\[ \Pi_{\text{sname}} \left( \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \left( T \right) \times \text{Supplier} \right) \]

Block nested loop join

\[ \text{Total cost: } 100 + 100 \times 100 = 10100 \]

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

\[ \text{T(Supply)} = 10000 \]
\[ \text{B(Supply)} = 100 \]
\[ \text{V(Supply, pno)} = 2500 \]
Physical Plan 2

\[ \pi_{\text{snname}}(\text{Supplier}(\text{sid}, \text{sname}, \text{scity}, \text{sstate})) \]

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

\[ \text{Supply} \]

\[ \text{T} = 4 \]

\[ \text{Main memory join} \]

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

\[ \text{T} = 5 \]

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

\[ \text{Supplier} \]

\[ \text{T} = 50 \]

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

\[ \text{Unclustered index lookup} \]

\[ \text{Supplier(scity)} \]

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

\[ \text{T} = 4 \]

\[ \text{Unclustered index lookup} \]

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

\[ \text{T} = 5 \]

\[ \text{Unclustered index lookup} \]

\[ \text{Supplier(scity)} \]

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

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

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

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

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

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

\[ \text{V(Supply, pno) = 2500} \]}
Physical Plan 2

\[ \Pi_{\text{sname}} \]
\[ \sigma_{\text{sstate} = \text{WA}} \]
\[ \pi_{\text{sname}} \]
\[ \sigma_{\text{pno} = 2} \]
\[ \text{Main memory join} \]

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

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

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

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

\[ \text{Unclustered index lookup} \]
\[ \text{Supplier(scity)} \]
Physical Plan 2

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

Unclustered index lookup
Supply(pno)

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

Main memory join

σ_{pno=2}

T = 4

σ_{sstate='WA'}

T = 5

σ_{scity='Seattle'}

T = 50

Unclustered index lookup
Supplier(scity)

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

Unclustered index lookup
Supply(pno)

T = 4
Physical Plan 3

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

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

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

\[ \text{Supply} \]

\[ \text{Supplier} \]

T(Supply) = 10000
B(Supply) = 100
V(Supply, scity) = 20
V(Supply, pno) = 2500

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

Cost of Supply(pno) = 4
Cost of Index join = 4
Total cost: 8
Physical Plan 3

```
π sname (σ scity='Seattle' ∧ sstate='WA'
    (σ pno=2
        sid = sid
        Supply
    )
    Supplier
)
```

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

Unclustered index lookup
Supply(pno)

Clustered Index join

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

T(Supply) = 10000
B(Supply) = 100
V(Supply, pno) = 2500
Physical Plan 3

\[
\begin{align*}
\text{Supplier}(\text{sid}, \text{sname}, \text{scity}, \text{sstate}) \\
\text{Supply}(\text{sid}, \text{pno}, \text{quantity})
\end{align*}
\]

\[
\begin{align*}
\text{T(Supplier)} &= 10000 \\
\text{B(Supplier)} &= 100 \\
\text{V(Supplier, scity)} &= 20 \\
\text{V(Supplier, state)} &= 10
\end{align*}
\]

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

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