CSE544
Data Management

Lectures 7-8
Query Optimization
Announcements

• HW2 due on Friday

• HW3
  – Main option (mandatory for CSE): SimpleDB
  – Alternate: data cleaning (a bit open ended)
An Intermediate Format: PAX

• PAX = Partition Attributes Across

• Addresses memory access bottleneck (not the disk bottleneck)
Figure 2.1: Storage models for storing database records inside disk pages: NSM (row-store) and DSM (a predecessor to column-stores). Figure taken from [5].
Current Scheme: Slotted Pages

Formal name: NSM (N-ary Storage Model)

- Records are stored sequentially
- Offsets to start of each record at end of page

Ailamaki VLDB’01  http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Predicate Evaluation using NSM

```
select name from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Predicate Evaluation using NSM

select name from R where age > 50

NSM pushes non-referenced data to the cache

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
**Predicate Evaluation using NSM**

```
select name from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Predicate Evaluation using NSM

select name from R where age > 50

NSM pushes non-referenced data to the cache

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Predicate Evaluation using NSM

select name from R where age > 50

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Need New Data Page Layout

- Eliminates unnecessary memory accesses
- Improves inter-record locality
- Keeps a record’s fields together
- Does not affect I/O performance

and, most importantly, is…

**low-implementation-cost, high-impact**

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Partition data *within* the page for spatial locality

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Partition Attributes Across (PAX)

Partition data *within* the page for spatial locality

Ailamaki VLDB'01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Partition Attributes Across (PAX)

Partition data within the page for spatial locality

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Partition Attributes Across (PAX)

Partition data *within* the page for spatial locality

Ailamaki VLDB'01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Partition Attributes Across (PAX)

Partition data within the page for spatial locality

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt]
Partition Attributes Across (PAX)

Partition data *within* the page for spatial locality

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
Predicate Evaluation using PAX

select name from R where age > 50

Fewer cache misses, low reconstruction cost

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
Predicate Evaluation using PAX

Fewer cache misses, low reconstruction cost

select name from R where age > 50

Ailamaki VLDB’01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt
A Real NSM Record

NSM: All fields of record stored together + slots
**PAX: Detailed Design**

<table>
<thead>
<tr>
<th>pid</th>
<th># attributes</th>
<th># records</th>
<th>attribute sizes</th>
<th>free space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1237</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4322</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Page Header**

- **F - Minipage**
  - Presence bits
  - V - Minipage
  - F - Minipage

**PAX: Group fields + amortizes record headers**

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
PAX - Summary

• Improves processor cache locality
• Does not affect I/O behavior
  – Same disk accesses for NSM or PAX storage
  – No need to change the buffer manager

• Today:
  – Most engines use a PAX layout of the disk
  – Beyond disk: Snowflake partitions tables horizontally into files, then uses column-store inside each file (hence, PAX)
Query Optimization
Query Optimization Motivation

SQL query

Parse & Rewrite Query

Select Logical Plan

Select Physical Plan

Query Execution

Disk

Declarative query

Recall physical and logical data independence

Logical plan

Physical plan
Query Optimization

Three major components:

1. Search space
   - Access path selection
   - Rewrite rules

2. Cardinality and cost estimation

3. Plan enumeration algorithms
Access Path

Access path: implements a selection $\sigma_P(R)$,
Note: $P$ is called search argument SARG

• A file scan, or

• An index \textit{plus} a matching selection condition
Access Path Selection

SELECT * FROM Supplier
WHERE sid > 300 ∧ scity='Seattle'

Indices:
  B+-tree on sid; clustered
  B+-tree on scity; unclustered

Which access path should we use?
Access Path Selection

SELECT * FROM Supplier
WHERE sid > 300 ∧ scity='Seattle'

Indices:
B+-tree on sid; clustered
B+-tree on scity; unclustered

Which access path should we use?

1. Sequential scan: cost = 100
Access Path Selection

**SELECT** * FROM Supplier
WHERE sid > 300 \land scity='Seattle'

Indices:
- B+-tree on **sid**; clustered
- B+-tree on **scity**; unclustered

Which access path should we use?

1. Sequential scan: cost = 100
2. Index scan on **sid**: cost = 7/10 * 100 = 70
Access Path Selection

```sql
SELECT * FROM Supplier
WHERE sid > 300 ∧ scity='Seattle'
```

Indices:

- B+-tree on `sid`; clustered
- B+-tree on `scity`; unclustered

Which access path should we use?

1. Sequential scan: cost = 100
2. Index scan on `sid`: cost = 7/10 * 100 = 70
3. Index scan on `scity`: cost = 1000/20 = 50
Rewrite Rules

Search space is defined by the set of rewrite rules that the optimizer implements
Example Optimization

```
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

SELECT x.sid, y.pno, y.quantity
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and x.scity = 'Seattle'
```
Example Optimization

\[
\begin{align*}
\Pi_{x.sid, y.pno, y.quantity} \\
\sigma_{x.scity = 'Seattle'} \\
\bowtie_{x.sid = y.sid}
\end{align*}
\]

SELECT x.sid, y.pno, y.quantity
FROM Supplier x, Supply y
WHERE x.sid = y.sid
and x.scity = 'Seattle'
Push Selections Down

\[
\Pi_{x \cdot \text{sid}, y \cdot \text{pno}, y \cdot \text{quantity}} \\
\sigma_{x \cdot \text{scity} = 'Seattle'} \\
\bowtie_{x \cdot \text{sid} = y \cdot \text{sid}} \\
\text{Supplier} \; x \\
\text{Supply} \; y
\]
Push Selections Down

\[ \Pi_{x.sid,y.pno,y.quantity} \sigma_{x.scity='Seattle'} \Join x.sid = y.sid \]

\[ \Pi_{x.sid,y.pno,y.quantity} \sigma_{x.scity='Seattle'} \Join x.sid = y.sid \]
Push Selections Down

\[
\prod_{x} \text{sid, y.pno, y.quantity}
\]

\[
\sigma_{x.scity='Seattle'}
\]

\[
\bowtie_{x.\text{sid} = y.\text{sid}}
\]

\[
\text{Supplier } x
\]

\[
\text{Supply } y
\]

\[
\sigma_{C}(R \bowtie S) = \sigma_{C}(R) \bowtie S \text{ when } C \text{ refers only to } R
\]
\( \text{Push Selections Down} \)

\[
\Pi_{x\text{.sid}, y\text{.pno}, y\text{.quantity}} \left( \sigma_{x\text{.scity} = \text{’Seattle’} \text{ and } y\text{.pno} = 5} (x \bowtie y) \right)
\]
Push Selections Down

\[ \Pi_{x, y} \text{sid, y.pno, y.quantity} \]

\[ \sigma_{x \text{.scity} = 'Seattle' \text{ and } y \text{.pno} = 5} \]

\[ \bowtie \text{Supplier x} \bowtie \text{Supply y} \]

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

\[ \sigma_{y \text{.pno} = 5} \]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

\[ \text{Push Selections Down} \]

\[ \Pi_{x.sid,y.pno,y.quantity} \quad \sigma_{x.scity=\text{\textquoteleft}Seattle\text{\textquoteleft} and y.pno=5} \]

\[ \Join_{x.sid = y.sid} \quad \text{Supplier } x \quad \text{Supply } y \]

\[ \sigma_{y.pno=5} \quad \sigma_{x.scity=\text{\textquoteleft}Seattle\text{\textquoteleft}} \]

\[ \sigma_{C_1\text{ and } C_2}(R \bowtie S) = \sigma_{C_1}(\sigma_{C_2}(R \bowtie S)) = \sigma_{C_1}(R \bowtie \sigma_{C_2}(S)) = \sigma_{C_1}(R) \bowtie \sigma_{C_2}(S) \]
Join Reorder

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)

\( \bowtie \) x.sid = y.sid
\( \bowtie \) y.pno = z.pno
\( \bowtie \) x.pno = z.pno

Supplier x  Supply y  Part z
Join Reorder

\[ \text{Supplier}(\text{sid, sname, scity, sstate}) \]
\[ \text{Supply}(\text{sid, pno, quantity}) \]
\[ \text{Part}(\text{pno, pname, pprice}) \]

\[ \text{Supplier x} \bowtie \text{Supply y} \bowtie \text{Part z} \]
\[ \text{Supplier x} \bowtie \text{Supply y} \bowtie \text{Part z} \]
Supplier($sid, sname, scity, sstate$)
Supply($sid, pno, quantity$)
Part($pno, pname, pprice$)

$\text{Join Reorder}$

$\text{Supplier x}$ $\bowtie$ $\text{Supply y}$ $\bowtie$ $\text{Part z}$

$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

$R \bowtie S = S \bowtie R$
Join Reorder

When is one plan better than the other?

\[(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)\]

\[R \bowtie S = S \bowtie R\]
Join Reorder

When is one plan better than the other?

Depends on:
|Supplier \Join Supply| \lessorequal\|Supply \Join Part|

(R \Join S) \Join T = R \Join (S \Join T)

R \Join S = S \Join R
Join Reorder

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)

\( \Join \) y.pno = z.pno

\( \Join \) x.sid = y.sid

\( \sigma \) x.scity = 'Seattle'

\( \sigma \) z.pprice > 99

Supplier x  Supply y  Part z
Join Reorder

When is one plan better than the other?

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)

\( \Join \) x.sid = y.sid
\( \sigma \) x.scity = 'Seattle'
\( \sigma \) z.pprice > 99

\( \Join \) y.pno = z.pno
\( \Join \) x.sid = y.sid
\( \sigma \) x.scity = 'Seattle'
\( \sigma \) z.pprice > 99

Supplier x
Supply y
Part z
Supplier x
Supply y
Part z
Join Reorder

When is one plan better than the other?

- Supplier $x$  
  - Supply $y$  
    - Part $z$  
      - $\sigma_{x.scity='Seattle'}$  
        - $\sigma_{z.pprice > 99}$  
          - $\sigma_{x.scity='Seattle'}$  
            - $\sigma_{z.pprice > 99}$

Lesson: need sizes of $\sigma_{x.scity='Seattle'}$ (Supplier), $\sigma_{z.pprice > 99}$ (Part)
Aggregate Push-down

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Part(pno, pname, pprice)

SELECT x.sstate, sum(y.quantity*z.price)
FROM Supplier x, Supply y, Part z
WHERE x.sid = y.sid and y.pno = z.pno
GROUP BY x.sstate
Aggregate Push-down

\[ \gamma_{x.sstate, \sum(y.quantity \times z.price)} \]

\[ \Join_{x.sid = y.sid} \]

\[ \Join_{y.pno = z.pno} \]

Supplier(x)  Supply(y)  Part(z)

SELECT x.sstate, \sum(y.quantity \times z.price) FROM Supplier x, Supply y, Part z WHERE x.sid = y.sid and y.pno = z.pno GROUP BY x.sstate
```
SELECT x.sstate, sum(y.quantity*z.price)
FROM Supplier x, Supply y, Part z
WHERE x.sid = y.sid and y.pno = z.pno
GROUP BY x.sstate
```
Aggregate Push-down

\[ \gamma_x.sstate, \sum(y.quantity \cdot z.price) \]

\[ \bowtie x.sid = y.sid \]
\[ \bowtie y.pno = z.pno \]

Supplier x \hspace{2cm} Supply y \hspace{2cm} Part z

Generalized distributivity law (next...)

\[ \gamma_y.sid, \sum(y.quantity \cdot z.price) \rightarrow s \]

SELECT x.sstate, \( \sum(y.quantity \cdot z.price) \)
FROM Supplier x, Supply y, Part z
WHERE x.sid = y.sid and y.pno = z.pno
GROUP BY x.sstate
Aggregate Push-Down

• Motivation: try this in postgres

```sql
select count(*) from author;
```

Answer: 2652053
Time: 0.058 s
Aggregate Push-Down

• Motivation: try this in postgres

```
select count(*) from author;
```

Answer: 2652053
Time: 0.058 s

```
select count(*) from publication;
```

Answer: 5120896
Time: 0.062 s
Aggregate Push-Down

• Motivation: try this in postgres

```sql
select count(*) from author;
```
Answer: 2652053
Time: 0.058 s

```sql
select count(*) from publication;
```
Answer: 5120896
Time: 0.062 s

```sql
select count(*) from author, publication;
```
Timeout!!!
Generalized Distributivity Law

```
SELECT count(*) from R, S where R.x = S.x
```
Generalized Distributivity Law

\[
\text{SELECT count(*) from R, S where R.x=S.x}
\]

\[
R:
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>d</td>
</tr>
<tr>
<td>h</td>
<td>g</td>
</tr>
</tbody>
</table>

\[
S:
<table>
<thead>
<tr>
<th>x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>k</td>
</tr>
<tr>
<td>h</td>
<td>m</td>
</tr>
</tbody>
</table>
\]

Answer = 5

Runtime = O(N^2)
Generalized Distributivity Law

SELECT count(*) from R, S where R.x=S.x

Answer = 5

Runtime = O(N^2)
Generalized Distributivity Law

SELECT count(*) from R, S where R.x = S.x

Answer = 5

Runtime = O(N^2)
Generalized Distributivity Law

SELECT count(*) from R, S where R.x=S.x

Answer = 5

Runtime = O(N²)

Runtime = O(N)

A:  
<table>
<thead>
<tr>
<th>x</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
</tr>
</tbody>
</table>

B:  
<table>
<thead>
<tr>
<th>x</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>h</td>
<td>1</td>
</tr>
</tbody>
</table>

A \bowtie B:  

\[ \bigotimes x \]

R(x,y) \bigotimes S(x,z)

A \bowtie B:  

\[ \bigotimes x \]

R(x,y) \bigotimes S(x,z)

\[ \bigotimes x \]

R(x,y) \bigotimes S(x,z)

R:  
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>d</td>
</tr>
<tr>
<td>h</td>
<td>g</td>
</tr>
</tbody>
</table>

S:  
<table>
<thead>
<tr>
<th>x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>k</td>
</tr>
<tr>
<td>h</td>
<td>m</td>
</tr>
</tbody>
</table>

Answer = 5

\[ \gamma \text{count}(*) \]

\[ x, y, z \]

\[ \gamma \text{sum}(c*d) \]

\[ x, c, d \]

Generalized Distributivity Law

$\forall Y, Z, \text{sum}(A \ast B \ast C \ast \cdots)$

\[ \bigotimes_X \cdots \cdots \]
Generalized Distributivity Law

\[ \forall Y, Z, \text{sum}(A \ast B \ast C \ast \ldots) \]

\[ \Join_X \]

\[ \ldots \]

\[ \ldots \]

\[ \forall X, Y, \text{sum}(A \ast C \ast E \ast \ldots) \rightarrow S_1 \]

\[ \forall X, Z, \text{sum}(B \ast D \ast F \ast \ldots) \rightarrow S_2 \]

\[ \ldots \]

\[ \ldots \]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Key / Foreign-Key

Select x.pno, x.quantity
From Supply x, Supplier y
Where x.sid = y.sid
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Key / Foreign-Key

Select x.pno, x.quantity
From Supply x, Supplier y
Where x.sid = y.sid

Select x.pno, x.quantity
From Supply x
Key / Foreign-Key

Select x.pno, x.quantity
From Supply x, Supplier y
Where x.sid = y.sid

Select x.pno, x.quantity
From Supply x

Only if these constraints hold:
1. Supplier.sid = key
2. Supply.sid = foreign key
3. Supply.sid NOT NULL
Practice

• Database optimizers typically have a database of rewrite rules
• E.g. SQL Server is rumored to have about 500 rules
• Rules become complex as they need to serve specialized types of queries
Search Space Challenges

• Search space is huge

• Typical compromises:
  – Left-deep plans
  – Plans without cartesian products
Left-Deep Plans and Bushy Plans

Left-deep plan

Bushy plan
Announcements

• HW2 was due last Friday

• Paper review due on Wednesday

• Project proposals due next Friday

• HW3 is posted
Query Optimization

Three major components:

1. Search space
2. Cardinality and cost estimation
3. Plan enumeration algorithms
Cardinality Estimation

**Problem**: given statistics on base tables and a query, estimate size of the answer

Very difficult, because:

- Need to do it very fast
- Need to use very little memory
Statistics on Base Data

- Number of tuples (cardinality) \( T(R) \)
- Number of physical pages \( B(R) \)
- Indexes, number of keys in the index \( V(R,a) \)
- Histogram on single attribute (1d)
- Histogram on two attributes (2d)

Computed periodically, often using sampling
Assumptions

• Uniformity

• Independence

• Containment of values

• Preservation of values
Size Estimation

Selection: size decreases by \textit{selectivity factor} $\theta$

\[ T(\sigma_{\text{pred}}(R)) = \theta_{\text{pred}} \times T(R) \]

\[ T(R \bowtie_{A=B} S) = \theta_{A=B} \times T(R) \times T(S) \]
Selectivity Factors

**Uniformity assumption**

Equality:
- \( \theta_{A=c} = \frac{1}{V(R,A)} \)
**Uniformity assumption**

**Equality:**

- $\theta_{A=c} = 1/V(R,A)$

**Range:**

- $\theta_{c_1<A<c_2} = (c_2 - c_1)/(\max(R,A) - \min(R,A))$
Selectivity Factors

**Uniformity assumption**

Equality:
- \( \theta_{A=c} = 1/V(R,A) \)

Range:
- \( \theta_{c1<A<c2} = (c2 - c1)/(\max(R,A) - \min(R,A)) \)

**Independence assumption**

- \( \theta_{\text{pred1 and pred2}} = \theta_{\text{pred1}} \times \theta_{\text{pred2}} = 1/V(R,A) \times 1/V(R,B) \)
Selectivity Factors

Join

• $\theta_{R.A=S.B} = \frac{1}{\max(V(R,A), V(S,B))}$

Why? Will explain next...
Selectivity Factors

_Containment of values_: if \( V(R,A) \leq V(S,B) \), then the set of A values of R is included in the set of B values of S

- Note: this indeed holds when A is a foreign key in R, and B is a key in S
Selectivity Factors

Assume $V(R,A) \leq V(S,B)$

- Tuple $t$ in $R$ joins with $T(S)/V(S,B)$ tuples in $S$
- Hence $T(R \bowtie_{A=B} S) = T(R) T(S) / V(S,B)$
Selectivity Factors

Assume $V(R,A) \leq V(S,B)$

• Tuple $t$ in $R$ joins with $T(S)/V(S,B)$ tuples in $S$
• Hence $T(R \bowtie_{A=B} S) = T(R) \cdot T(S) / V(S,B)$

In general:

• $T(R \bowtie_{A=B} S) = T(R) \cdot T(S) / \max(V(R,A),V(S,B))$
• $\theta_{R.A=S.B} = 1/ ( \max( V(R,A), V(S,B))$
Final Assumption

Preservation of values:

For any other attribute C:

- \( V(R \bowtie_{A=B} S, C) = V(R, C) \) or
- \( V(R \bowtie_{A=B} S, C) = V(S, C) \)

- This is needed higher up in the plan
Computing the Cost of a Plan

• Estimate **cardinalities** bottom-up

• Estimate **cost** by using estimated cardinalities

• Extensive example next...
Logical Query Plan 1

\[
\begin{align*}
\Pi_{sname} & \\
\sigma_{pno=2 \land scity='Seattle' \land sstate='WA'} & \\
\end{align*}
\]

\[
\begin{align*}
sid = sid & \\
\end{align*}
\]

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

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

M=11
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'

Estimated (why?)

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

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

T = 10000

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

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

M = 11
Logical Query Plan 1

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

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

\[ T < 1 \]

Supplier \((\text{sid, sname, scity, sstate})\)

Supply \((\text{sid, pno, quantity})\)

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

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

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

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

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

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

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

Estimated (why?)

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

`M=11`

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

T(Supplier) = 1000
B(Supplier) = 100
V(Supplier, scity) = 20
V(Supplier, sstate) = 10
```
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'

M=11
Logical Query Plan 2

\[
\begin{align*}
\sigma_{\text{scity} = \text{Seattle}} \land \sigma_{\text{sstate} = \text{WA}}(\text{Supplier}) &= 1000 \quad \text{B(Supplier)} = 100 \quad \text{V(Supplier, scity)} = 20 \quad \text{V(Supplier, state)} = 10 \\
\sigma_{\text{pno} = 2}(\text{Supply}) &= 10000 \quad \text{B(Supply)} = 100 \quad \text{V(Supply, pno)} = 2500
\end{align*}
\]

\[M=11\]
Logical Query Plan 2

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

Very wrong! Why?

M=11
Logical Query Plan 2

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

**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

**M=11**
Physical Plan 1

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

Total cost: \( 100/10 \times 100 = 1000 \)

Scan

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

Supplier

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

M=11
Physical Plan 1

\[ \Pi_{\text{sname}} T \sigma_{\text{pno}=2 \land \text{scity}='Seattle' \land \text{sstate}='WA'} \]

\[ T = 10000 \]

\[ T < 1 \]

\[ \text{Total cost: } 100 + 100 \times 100/10 = 1100 \]

\[ \text{Scan} \]

\[ \text{Scan} \]

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

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

\[ \text{Supply} \]

\[ \text{Supplier} \]

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

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

\[ M=11 \]
Physical Plan 2

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

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

\[ \text{Supply} \]

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

\[ \text{Supplier} \]

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

Unclustered index lookup

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

Main memory join

Unclustered index lookup

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

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

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

Total cost: 54

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

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

\[ M=11 \]
Physical Plan 2

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

Total cost: \( 54 \)

\text{Unclustered index lookup Supply}(\text{pno})
\text{Unclustered index lookup Supplier}(\text{scity})
Physical Plan 2

\[ \pi_{\text{sname}}(\sigma_{\text{pno}=2}(\text{Supply})) \]

\[ \sigma_{\text{sstate}=\text{WA}}(\text{Supplier}) \]

Cost of Supply(\text{pno}) = 4
Cost of 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 3

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

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

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

\[ \text{Supply} \]

\[ \text{Supplier} \]

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

Unclustered index lookup
Supply(pno)

Clustered Index join

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

M=11

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

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Physical Plan 3

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

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

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

Unclustered index lookup

Supply(pno)

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

Clustered Index join

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

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

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

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

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

\[ \text{M=11} \]

\[ \text{T(Supplier) = 10000} \]
\[ \text{B(Supplier) = 100} \]
\[ \text{V(Supplier, scity) = 20} \]
\[ \text{V(Supplier, sstate) = 10} \]

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

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
Physical Plan 3

\[ \pi_{sname} \]

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

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

\[ \text{Supply} \]

\[ \text{Supplier} \]

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

\[ M = 11 \]

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

• We considered only IO cost; in general we need IO+CPU

• We assumed that all index pages were in memory: sometimes we need to add the cost of fetching index pages from disk
Histograms

• $T(R)$, $V(R,A)$ too coarse
• Histogram: separate stats per bucket

• In each bucket store:
  – $T($bucket$)$
  – $V($bucket,A$)$ – optional
Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Employee, age) = 50
\sigma_{age=48}(Employee) = ?
Histograms

Employee(ssn, name, age)

\[ T(\text{Employee}) = 25000, \quad V(\text{Employee, age}) = 50 \]

\[ \sigma_{\text{age}=48}(\text{Employee}) = ? \]

Estimate: \[ T(\text{Employee}) / V(\text{Employee}, \text{age}) = 500 \]
Histograms

**Employee**(ssn, name, age)

\(T(\text{Employee}) = 25000,\ V(\text{Employee}, \text{age}) = 50\)

\(\sigma_{\text{age}=48}(\text{Empolyee}) = ?\)

**Estimate:** \(T(\text{Employee}) / V(\text{Employee}, \text{age}) = 500\)

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>T =</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
</tbody>
</table>
Histograms

**Employee**(ssn, name, age)

\[ T(\text{Employee}) = 25000, \ V(\text{Employee, age}) = 50 \]

\[ \sigma_{\text{age}=48}(\text{Employee}) = ? \]

Estimate: \[ T(\text{Employee}) / V(\text{Employee, age}) = 500 \]

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T = )</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
</tbody>
</table>

Assume \( V = 10 \)
Histograms

**Employee**(ssn, name, age)

\[ T(\text{Employee}) = 25000, \ V(\text{Employee, age}) = 50 \]

\[ \sigma_{\text{age}=48}(\text{Employee}) = ? \]

*Estimate: \( T(\text{Employee}) / V(\text{Employee, age}) = 500 \)*

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>T =</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
</tbody>
</table>

*Estimate: 12000/10 = 1200*

Assume \( V = 10 \)
Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Employee, age) = 50
\( \sigma_{age=48}(Employee) = ? \)

Estimate: \( \frac{T(Employee)}{V(Employee, age)} = 500 \)

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>T =</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
<tr>
<td>V =</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Estimate: \( \frac{12000}{10} = 1200 \)
Histograms

Employee(ssn, name, age)

T(Employee) = 25000, V(Employee, age) = 50
σ_{age=48}(Employee) = ?

Estimate: \( \frac{T(Employee)}{V(Employee, age)} = 500 \)

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
<tr>
<td>V</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Estimate: \( \frac{12000}{10} = 1200 \) \( \frac{12000}{6} = 2000 \)
Types of Histograms

• Eq-Width

• Eq-Depth

• Compressed: store outliers separately

• V-Optimal histograms
**Employee(ssn, name, age)**

### Histograms

**Eq-width:**

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..20</th>
<th>20..29</th>
<th>30-39</th>
<th>40-49</th>
<th>50-59</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuples</td>
<td>200</td>
<td>800</td>
<td>5000</td>
<td>12000</td>
<td>6500</td>
<td>500</td>
</tr>
</tbody>
</table>

**Eq-depth:**

<table>
<thead>
<tr>
<th>Age:</th>
<th>0..32</th>
<th>33..41</th>
<th>42-46</th>
<th>47-52</th>
<th>53-58</th>
<th>&gt; 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuples</td>
<td>1800</td>
<td>2000</td>
<td>2100</td>
<td>2200</td>
<td>1900</td>
<td>1800</td>
</tr>
</tbody>
</table>

**Compressed:** store separately highly frequent values: (48,1900)
V-Optimal Histograms

• Error:

\[
\sum_{v \in \text{Domain}(A)} \left( |\sigma_{A=v}(R)| - \text{est}_{\text{Hist}}(\sigma_{A=v}(R)) \right)^2
\]

• Bucket boundaries = argmin_{\text{Hist}}(Error)

• Dynamic programming

• Modern databases systems use V-optimal histograms or some variations
Discussion

• Cardinality estimation = open problem

• Histograms:
  – Small number of buckets (why?)
  – Updated only periodically (why?)
  – No 2d histograms (except db2) why?

• Samples:
  – Fail for low selectivity estimates, joins

• Cross-join correlation – open problem
Query Optimization

Three major components:

1. Search space
2. Cardinality and cost estimation
3. Plan enumeration algorithms
Two Types of Optimizers

- Heuristic-based optimizers
- Cost-based optimizers (next)
Two Types of Plan Enumeration Algorithms

- Dynamic programming *(in class)*
  - Based on System R [Selinger 1979]
  - *Join reordering algorithm*

- Rule-based algorithm *(will not discuss)*
  - Database of rules (=algebraic laws)
  - Usually: dynamic programming
System R Optimizer

For each subquery $Q \subseteq \{R_1, \ldots, R_n\}$, compute best plan:

- **Step 1:** $Q = \{R_1\}, \{R_2\}, \ldots, \{R_n\}$
- **Step 2:** $Q = \{R_1, R_2\}, \{R_1, R_3\}, \ldots, \{R_{n-1}, R_n\}$
- ...
- **Step n:** $Q = \{R_1, \ldots, R_n\}$
Details

For each subquery \( Q \subseteq \{R_1, \ldots, R_n\} \) store:

- Estimated Size(Q)
- A best plan for Q: Plan(Q)
- The cost of that plan: Cost(Q)
Step 1: single relations \{R_1\}, \{R_2\}, \ldots, \{R_n\}

- Consider all possible access paths:
  - Sequential scan, or
  - Index 1, or
  - Index 2, or
  - ...

- Keep optimal plan for each “interesting order”
Details

Step $k = 2 \ldots n$:
For each $Q = \{R_{i_1}, \ldots, R_{i_k}\}$

- For each $j=1,\ldots,k$:
  - Let: $Q' = Q - \{R_{i_j}\}$
  - Let: $Plan(Q') \bowtie R_{i_j} \quad Cost(Q') + CostOf(\bowtie)$

- $Plan(Q), Cost(Q) = \text{cheapest of the above}$
  - Keep separate optimal for “interesting orders”
Discussion

- All database systems implement Selinger’s algorithm for join reorder

- For other operators (group-by, aggregates, difference): rule-based

- Many search strategies beyond dynamic programming
Final Discussion

• Query optimizer = critical part of DBMS
• Search space + Size est + Algorithm
• Ideal: find “optimal” plan
• In practice: avoid “very bad plans”
• Successful because:
  – RA is a set-at-a-time language
  – RA is order-independent
• Next time:
  How good are they?; New approaches