Query Optimization

- Relational algebra level
- Detailed query plan level

Relational algebra optimization

- Transformation rules (preserve equivalence)
- What are good transformations?
  - Heuristic application of transformations

Query Equivalence

- Two queries q and q’ are equivalent:
  - If for every database instance I
    - Contents of all the tables
  - Both queries have the same result

\( q \equiv q' \iff \forall I: q(I) = q'(I) \)
Rules: Natural joins & cross products & union

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

Note:

- Carry attribute names in results, so order is not important
- Can also write as trees, e.g.:

```
  \( T \)  
 / \   
R   S
```

Rules: Natural joins & cross products & union

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

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

Rules: Selects

\[ \sigma_{p1 \land p2}(R) = \]
\[ \sigma_{p1}(\sigma_{p2}(R)) \]

\[ \sigma_{p1 \lor p2}(R) = \]
\[ \sigma_{p1}(R) \cup \sigma_{p2}(R) \]

Rules: Selects

\[ \sigma_{p1 \land p2}(R) = \]
\[ \sigma_{p1}(R) \land \sigma_{p2}(R) \]

\[ \sigma_{p1 \lor p2}(R) = \]
\[ \sigma_{p1}(R) \lor \sigma_{p2}(R) \]

 Bags vs. Sets

\[ R = \{a,a,b,b,b,c\} \]
\[ S = \{b,b,c,c,d\} \]
\[ R \cup S = ? \]
Bags vs. Sets

R = {a,a,b,b,b,c}
S = {b,b,c,c,d}
RUS = ?

- Option 1  SUM
  RUS = {a,a,b,b,b,b,c,c,c,d}
- Option 2  MAX
  RUS = {a,a,b,b,b,c,c,d}

Option 2 (MAX) makes this rule work:

σ_{p_1 v p_2}(R) = σ_{p_1}(R) U σ_{p_2}(R)

Example: R={a,a,b,b,b,c}
P1 satisfied by a,b; P2 satisfied by b,c

σ_{p_1}(R) = {a,a,b,b,b}
σ_{p_2}(R) = {b,b,b,c}
σ_{p_1}(R) U σ_{p_2}(R) = {a,a,b,b,b,c}

Option 2 (MAX) makes this rule work:

σ_{p_1 v p_2}(R) = σ_{p_1}(R) U σ_{p_2}(R)

Example: R={a,a,b,b,b,c}
P1 satisfied by a,b; P2 satisfied by b,c

σ_{p_1}(R) = {a,a,b,b,b,c}

“Sum” option makes more sense:

Senators (……)
Rep (……)
T1 = \pi_{yr,state} Senators; T2 = \pi_{yr,state} Reps

<table>
<thead>
<tr>
<th>Yr</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>CA</td>
</tr>
<tr>
<td>99</td>
<td>CA</td>
</tr>
<tr>
<td>98</td>
<td>AZ</td>
</tr>
</tbody>
</table>

Union?

Executive Decision

-> Use “SUM” option for bag unions
-> Some rules cannot be used for bags

Rules: Project

Let: X = set of attributes
Y = set of attributes
XY = X U Y

\pi_{xy}(R) =
**Rules: Project**

Let: $X$ = set of attributes
    $Y$ = set of attributes
    $XY = X \cup Y$

$$\pi_{xy}(R) = \pi_x [\pi_y (R)]$$

**Rules: Project**

Let: $X$ = set of attributes
    $Y$ = set of attributes
    $XY = X \cup Y$

$$\pi_{xy}(R) = \pi_x [\pi_y (R)]$$

---

**Rules: $\sigma + \bowtie$ combined**

Let $p$ = predicate with only $R$ attribs
    $q$ = predicate with only $S$ attribs
    $m$ = predicate with only $R,S$ attribs

$$\sigma_p (R \bowtie S) =$$
$$\sigma_q (R \bowtie S) =$$

**Rules: $\sigma + \bowtie$ combined**

Let $p$ = predicate with only $R$ attribs
    $q$ = predicate with only $S$ attribs
    $m$ = predicate with only $R,S$ attribs

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$
$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

---

**Rules: $\sigma + \bowtie$ combined (continued)**

Some Rules can be Derived:

$$\sigma_{paq} (R \bowtie S) =$$
$$\sigma_{paqam} (R \bowtie S) =$$
$$\sigma_{pavq} (R \bowtie S) =$$

**Rules: $\sigma + \bowtie$ combined (continued)**

Do one:

$$\sigma_{paq} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$
$$\sigma_{paqam} (R \bowtie S) =$$
    $$\omega_m [(\sigma_p R) \bowtie (\sigma_q S)]$$
$$\sigma_{pavq} (R \bowtie S) =$$
    $$[(\sigma_p R) \bowtie S] \cup [R \bowtie (\sigma_q S)]$$
Derivation for first one:

$$\sigma_{p \land q} (R \bowtie S) =$$
$$\sigma_p [\sigma_q (R \bowtie S)] =$$
$$\sigma_p [R \bowtie \sigma_q (S)] =$$
$$[\sigma_p (R)] \bowtie [\sigma_q (S)]$$

Rules: \(\pi, \sigma\) combined

Let \(x\) = subset of \(R\) attributes
\(z\) = attributes in predicate \(P\)
(subset of \(R\) attributes)

$$\pi_x [\sigma_p (R)] = \{\sigma_p [\pi_x (R)]\}$$

Rules: \(\pi, \bowtie\) combined

Let \(x\) = subset of \(R\) attributes
\(y\) = subset of \(S\) attributes
\(z\) = intersection of \(R, S\) attributes

$$\pi_{xy} (R \bowtie S) =$$
$$\pi_{xy} [\pi_{xz} (R)] \bowtie [\pi_{yz} (S)]$$
\[ \pi_{xy} \{ \sigma_p (R \bowtie S) \} = \pi_{xy} \{ \sigma_p (R \bowtie S) \} = \pi_{xy} \{ \sigma_p [\pi_{xz'} (R) \bowtie \pi_{yz'} (S)] \} \]

\[ z' = z \cup \{ \text{attributes used in } P \} \]

**Rules** for \( \sigma, \pi \) combined with \( X \)

similar...

e.g., \( \sigma_p (R \times S) = ? \)

**Rules** \( \sigma, U \) combined:

\[ \sigma_p (R \cup S) = \sigma_p (R) \cup \sigma_p (S) \]
\[ \sigma_p (R - S) = \sigma_p (R) - S = \sigma_p (R) - \sigma_p (S) \]

Which are “good” transformations?

- \( \sigma_{p1 \bowtie p2} (R) \rightarrow \sigma_{p1} [\sigma_{p2} (R)] \)
- \( \sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S \)
- \( R \bowtie S \rightarrow S \bowtie R \)
- \( \pi_x [\sigma_p (R)] \rightarrow \pi_x \{ \sigma_p [\pi_{xz} (R)] \} \)

Conventional wisdom:
do projects early

**Example:** \( R(A,B,C,D,E) \)
\( x=\{E\} \)
\( P: \{A=3\} \wedge \{B=\text{"cat"}\} \)

\[ \pi_x \{ \sigma_p (R) \} \text{ vs. } \pi_x \{ \sigma_p [\pi_{ABE} (R)] \} \]
**But** What if we have A, B indexes?

B = “cat”  
A = 3

Intersect pointers to get pointers to matching tuples  
e.g., using bitmaps

---

**Bottom line:**

- No transformation is always good
- Usually good: early selections  
  - Exception: expensive selection conditions  
  - E.g., UDFs

---

**More transformations**

- Eliminate common sub-expressions
- Detect constant expressions
- Other operations: duplicate elimination

---

**Pushing Selections**

- Idea:
  - Join conditions equate attributes  
  - For parts of algebra tree (scope) store which attributes have to be the same  
  - Called Equivalence classes
- Example: R(a,b), S(c,d)

\[
\sigma_{b=3}(R \bowtie_{b=c} S) = \sigma_{b=3}(R) \bowtie_{b=c} \sigma_{b=3}(S)
\]

---

**Outer-Joins**

- Not commutative  
  - R × S ≠ S × R
- p – condition over attributes in A
- A list of attributes from R  
  \[ \alpha_p (R \bowtie_{A=B} S) \equiv \sigma_p (R) \bowtie_{A=B} S \]
  
  Not \[ \alpha_p (R \bowtie_{A=B} S) \equiv R \bowtie_{A=B} \sigma_p (S) \]

---

**Summary Equivalences**

- Associativity:  
  \[(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)\]
- Commutativity:  
  \[R \bowtie S \equiv S \bowtie R\]
- Distributivity:  
  \[(R \bowtie S) \bowtie T \equiv (R \bowtie T) \bowtie (S \bowtie T)\]
- Difference between Set and Bag Equivalences  
  - Only some equivalence are useful
Outline - Query Processing

• Relational algebra level
  – transformations
  – good transformations
• Detailed query plan level
  – estimate costs
  – generate and compare plans

Estimating result size

• Keep statistics for relation R
  – \( T(R) \) : # tuples in R
  – \( S(R) \) : # of bytes in each R tuple
  – \( B(R) \) : # of blocks to hold all R tuples
  – \( V(R, A) \) : # distinct values in R
    for attribute A

Example

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
</tr>
</tbody>
</table>

\( A \): 20 byte string
\( B \): 4 byte integer
\( C \): 8 byte date
\( D \): 5 byte string

Size estimates for \( W = R_1 \times R_2 \)

\[
T(W) = T(R_1) \times T(R_2) + S(R_1) + S(R_2)
\]

\[
S(W) = S(R_1) + S(R_2)
\]

\( T(R) = 5 \quad S(R) = 37 \)

\[
V(R,A) = 3 \quad V(R,C) = 5
\]

\[
V(R,B) = 1 \quad V(R,D) = 4
\]
Size estimates for $W = R_1 \times R_2$

$T(W) = T(R_1) \times T(R_2)$

$S(W) = S(R_1) + S(R_2)$

Size estimate for $W = \sigma_{A=a}(R)$

$S(W) = S(R)$

$T(W) = ?$

### Example

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

$V(R,A)=3$

$V(R,B)=1$

$V(R,C)=5$

$V(R,D)=4$

$W = \sigma_{Z=val}(R)$  \quad T(W) = \frac{T(R)}{V(R,Z)}$

### Example

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

$V(R,A)=3$

$V(R,B)=1$

$V(R,C)=5$

$V(R,D)=4$

### Assumption:

Values in select expression $Z = val$ are uniformly distributed over possible $V(R,Z)$ values.

### Alternate Assumption:

Values in select expression $Z = val$ are uniformly distributed over domain with $DOM(R,Z)$ values.
Example

\begin{tabular}{|c|c|c|c|}
\hline
R & A & B & C & D \\
\hline
cat & 1 & 10 & a & \\
cat & 1 & 20 & b & \\
dog & 1 & 30 & a & \\
dog & 1 & 40 & c & \\
bat & 1 & 50 & d & \\
\hline
\end{tabular}

Alternate assumption

\begin{align*}
V(R,A) &= 3 & \text{DOM}(R,A) &= 10 \\
V(R,B) &= 1 & \text{DOM}(R,B) &= 10 \\
V(R,C) &= 5 & \text{DOM}(R,C) &= 10 \\
V(R,D) &= 4 & \text{DOM}(R,D) &= 10 \\
\end{align*}

Example

\begin{tabular}{|c|c|c|c|}
\hline
R & A & B & C & D \\
\hline
cat & 1 & 10 & a & \\
cat & 1 & 20 & b & \\
dog & 1 & 30 & a & \\
dog & 1 & 40 & c & \\
bat & 1 & 50 & d & \\
\hline
\end{tabular}

Alternate assumption

\begin{align*}
V(R,A) &= 3 & \text{DOM}(R,A) &= 10 \\
V(R,B) &= 1 & \text{DOM}(R,B) &= 10 \\
V(R,C) &= 5 & \text{DOM}(R,C) &= 10 \\
V(R,D) &= 4 & \text{DOM}(R,D) &= 10 \\
\end{align*}

\[ W = \sigma_{z=val}(R) \quad T(W) = ? \]

\[ C=val \Rightarrow T(W) = (1/10)1 + (1/10)1 + ... = (5/10) = 0.5 \]

\[ B=val \Rightarrow T(W) = (1/10)5 + 0 + 0 = 0.5 \]

\[ A=val \Rightarrow T(W) = (1/10)2 + (1/10)2 + (1/10)1 = 0.5 \]

Selection cardinality

\[ SC(R,A) = \text{average} \# \text{ records that satisfy equality condition on } R.A \]

\[ SC(R,A) = \frac{T(R)}{V(R,A)} \]

What about \( W = \sigma_{z \geq \text{val}}(R) \) ?

\[ T(W) = ? \]

What about \( W = \sigma_{z \leq \text{val}}(R) \) ?

\[ T(W) = ? \]

• Solution # 1:

\[ T(W) = T(R)/2 \]
What about \( W = \sigma_{z \geq \text{val}}(R) \) ?

\[ T(W) = ? \]

- **Solution # 1:**
  \[ T(W) = T(R)/2 \]

- **Solution # 2:**
  \[ T(W) = T(R)/3 \]

- **Solution # 3:** Estimate values in range

**Example**

- **Table:**
  
<table>
<thead>
<tr>
<th>R</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1</td>
</tr>
<tr>
<td>V(R,Z)</td>
<td>10</td>
</tr>
<tr>
<td>Max</td>
<td>20</td>
</tr>
</tbody>
</table>

\[ W = \sigma_{z \geq 15}(R) \]

\[ f = \frac{20-15+1}{20-1+1} = \frac{6}{20} \]

\[ T(W) = f \times T(R) \]

**Equivalent:**

\[ f \times V(R,Z) = \text{fraction of distinct values} \]

\[ T(W) = [f \times V(Z,R)] \times T(R) = f \times T(R) \]

\[ V(Z,R) \]

**Size estimate for** \( W = R_1 \bowtie R_2 \)

Let \( x \) = attributes of \( R_1 \)
Let \( y \) = attributes of \( R_2 \)

**Case 1**

\[ X \cap Y = \emptyset \]

Same as \( R_1 \times R_2 \)
**Case 2** \[ W = R_1 \bowtie R_2 \quad X \cap Y = A \]

- **R1** | A | B | C
- **R2** | A | D

**Assumption:**
- \( V(R_1, A) \leq V(R_2, A) \Rightarrow \) Every A value in R1 is in R2
- \( V(R_2, A) \leq V(R_1, A) \Rightarrow \) Every A value in R2 is in R1

### Computing \( T(W) \) when \( V(R_1, A) \leq V(R_2, A) \)

- Take 1 tuple
- Match

1 tuple matches with \( \frac{T(R_2)}{V(R_2, A)} \) tuples...

So

\[
T(W) = \frac{T(R_2) \times T(R_1)}{V(R_2, A)}
\]

### In general

- \( W = R_1 \bowtie R_2 \)

\[
T(W) = \frac{T(R_2) T(R_1)}{\max\{ V(R_1, A), V(R_2, A) \}}
\]

[A is common attribute]
Case 2 with alternate assumption

Values uniformly distributed over domain

\[
\begin{align*}
R1 & \begin{array}{ccc}
A & B & C \\
\end{array} \\
R2 & \begin{array}{ccc}
A & D \\
\end{array}
\end{align*}
\]

This tuple matches \( T(R2)/\text{DOM}(R2,A) \) so

\[
T(W) = \frac{T(R2) T(R1)}{\text{DOM}(R2, A)} = \frac{T(R2) T(R1)}{\text{DOM}(R1, A)}
\]

In all cases:

\[
S(W) = S(R1) + S(R2) - S(\text{size of attribute } A)
\]

Using similar ideas, we can estimate sizes of:

- \( \Pi_{AB}(R) \)
- \( \sigma_{A=a \land B=b}(R) \)
- Union, intersection, diff,

Note: for complex expressions, need intermediate \( T, S, V \) results.

E.g. \( W = [\sigma_{A=a}(R1)] \Join R2 \)

Treat as relation \( U \)

\[
T(U) = \frac{T(R1)}{V(R1,A)}
\]

\[
S(U) = S(R1)
\]

Also need \( V(U, *) \)!!

Example

\[
\begin{array}{cccc}
R1 & A & B & C & D \\
\hline
\text{cat} & 1 & 10 & 10 & \text{V(R1,A)=3} \\
\text{cat} & 1 & 20 & 20 & \text{V(R1,B)=1} \\
\text{dog} & 1 & 30 & 10 & \text{V(R1,C)=5} \\
\text{dog} & 1 & 40 & 30 & \text{V(R1,D)=3} \\
\text{bat} & 1 & 50 & 10 & \text{U = } \sigma_{A=a}(R1)
\end{array}
\]

To estimate \( V_s \)

E.g., \( U = \sigma_{A=a}(R1) \)

Say \( R1 \) has attribs \( A, B, C, D \)

\[
\begin{align*}
V(U, A) &= \\
V(U, B) &= \\
V(U, C) &= \\
V(U, D) &=
\end{align*}
\]
Example:

\[\begin{array}{c|cccc}
   & A & B & C & D \\
R1 & cat & 1 & 10 & 10 \\
 & cat & 1 & 20 & 20 \\
 & dog & 1 & 30 & 10 \\
 & dog & 1 & 40 & 30 \\
 & bat & 1 & 50 & 10 \\
\end{array}\]

\[\begin{array}{c}
V(R1,A)=3 \\
V(R1,B)=1 \\
V(R1,C)=5 \\
V(R1,D)=3 \\
\end{array}\]

Possible Guess:

\[U = \sigma_{A=a}(R1)\]

\[\begin{array}{c}
V(U,A) = 1 \\
V(U,B) = V(R,B) \\
\end{array}\]

\[V(D,U) \text{... somewhere in between}\]

For Joins:

\[U = R1(A,B) \bowtie R2(A,C)\]

\[\begin{array}{c}
V(U,A) = \min\{V(R1,A), V(R2,A)\} \\
V(U,B) = V(R1,B) \\
V(U,C) = V(R2,C) \\
\end{array}\]

Example:

\[Z = R1(A,B) \bowtie R2(B,C) \bowtie R3(C,D)\]

\[\begin{array}{c}
R1 & T(R1) = 1000 & V(R1,A)=50 & V(R1,B)=100 \\
R2 & T(R2) = 2000 & V(R2,B)=200 & V(R2,C)=300 \\
R3 & T(R3) = 3000 & V(R3,C)=90 & V(R3,D)=500 \\
\end{array}\]

Partial Result:

\[U = R1 \bowtie R2\]

\[\begin{array}{c}
T(U) = \frac{1000 \times 2000}{200} & V(U,A) = 50 \\
 & V(U,B) = 100 \\
 & V(U,C) = 300 \\
\end{array}\]

\[Z = U \bowtie R3\]

\[\begin{array}{c}
T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300} & V(Z,A) = 50 \\
 & V(Z,B) = 100 \\
 & V(Z,C) = 90 \\
 & V(Z,D) = 500 \\
\end{array}\]
Approximating Distributions

- Summarize the distribution
  - Used to better estimate result sizes
  - Without the need to look at all the data

- Concerns
  - Error metric: How to measure preciseness
  - Memory consumption
  - Computational Complexity

Parameterized distribution
- E.g., gauss distribution
- Adapt parameters to fit data

Histograms
- Divide domain into ranges (buckets)
- Store the number of tuples per bucket
- Both need to be maintained

Maintaining Statistics

- Use separate command that triggers statistics collection
  - Postgres: ANALYZE
- During query processing
  - Overhead for queries
- Use Sampling?

Estimating Result Size using Histograms

\[ \sigma_{A=val}(R) = ? \]

\[ \frac{\#B}{|B|} \]
Join Size using Histograms

- $R \bowtie S$
- Use
  \[
  T(W) = \frac{T(R2) T(R1)}{\max\{V(R1,A), V(R2,A)\}}
  \]
- Apply for each bucket

Equi-width vs. Equi-depth

- Equi-width
  - All buckets contain the same number of values
  - Easy, but inaccurate
- Equi-depth (used by most DBMS)
  - All buckets contain the same number of tuples
  - Better accuracy, need to sort data to compute

Construct Equi-depth Histograms

- Sort input
- Determine size of buckets
  - #bucket / #tuples
- Example 3 buckets
  1, 5, 44, 6, 10, 12, 3, 6, 7
  1, 3, 5, 6, 6, 7, 10, 12, 44
  [1–5] [6–8] [9–44]

Advanced Techniques

- Wavelets
- Approximate Histograms
- Sampling Techniques
- Compressed Histograms
Summary

- Estimating size of results is an “art”

- Don’t forget: Statistics must be kept up to date... (cost?)

Outline

- Estimating cost of query plan
  - Estimating size of results — done!
  - Estimating # of IOs — next...
  - Operator Implementations

- Generate and compare plans