Query in SQL

→

Query Plan in Algebra (logical)

→

Other Query Plan in Algebra (logical)
Query plan 1 (in relational algebra)

\[ \Pi_{B,D} \sigma_{R.A= "c" \land S.E=2 \land R.C=S.C} \times RS \]
Query plan 2 (in relational algebra)

\[ \Pi_{B,D}(\sigma_{R.A = "c"}(R) \bowtie \sigma_{S.E = 2}(S)) \]

natural join
Relational algebra optimization

- What are transformation rules?
  - preserve equivalence
- What are good transformations?
  - reduce query execution costs
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.:
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

\[ \sigma_{p_1 \land p_2}(R) = \sigma_{p_1} \left[ \sigma_{p_2}(R) \right] \]

\[ \sigma_{p_1 \lor p_2}(R) = \left[ \sigma_{p_1}(R) \right] \cup \left[ \sigma_{p_2}(R) \right] \]
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,b,c,c,c,d\}

• **Option 2** MAX
  RUS = \{a,a,b,b,b,c,c,d\}
Option 2 (MAX) makes this rule work:

$$\sigma_{p_1 \lor p_2} (R) = \sigma_{p_1}(R) \cup \sigma_{p_2}(R)$$

**Example:** $R=\{a,a,b,b,b,c\}$

- $P_1$ satisfied by $a,b$; $P_2$ satisfied by $b,c$

  $$\sigma_{p_1 \lor p_2} (R) = \{a,a,b,b,b,c\}$$

  $$\sigma_{p_1}(R) = \{a,a,b,b,b\}$$

  $$\sigma_{p_2}(R) = \{b,b,b,c\}$$

  $$\sigma_{p_1}(R) \cup \sigma_{p_2} (R) = \{a,a,b,b,b,c\}$$
"Sum" option makes more sense:

Senators (…….)  Rep (…….)

$T_1 = \pi_{yr,\text{state}} \text{Senators}$;  $T_2 = \pi_{yr,\text{state}} \text{Reps}$

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

Union?
Executive Decision

- Use "SUM" option for bag unions
- Some rules cannot be used for bags
Rules: Project

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

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$
Rules: $\sigma + \bowtie$ combined

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

\[
\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S
\]
\[
\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]
\]
Rules: $\sigma + \Join$ combined (continued)

Some Rules can be Derived:

$\sigma_{p \land q} (R \Join S) =$

$\sigma_{p \land q \land m} (R \Join S) =$

$\sigma_{p \lor q} (R \Join S) =$
Do one, others for homework:

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

\[ \sigma_{p \land q \land m} (R \bowtie S) = \sigma_m \left[ (\sigma_p R) \bowtie (\sigma_q S) \right] \]

\[ \sigma_{p \lor q} (R \bowtie S) = \left[ (\sigma_p R) \bowtie S \right] \cup \left[ R \bowtie (\sigma_q S) \right] \]
--> 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)

\[
\pi_x[\sigma_p(R)] = \pi_x \{ \sigma_p[\pi_x(R)] \}
\]
Rules: $\pi$, $\bowtie$ combined

Let $x = \text{subset of } R \text{ attributes}$
$y = \text{subset of } S \text{ attributes}$
$z = \text{intersection of } R, S \text{ attributes}$

$\pi_{xy}(R \bowtie S) =$

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

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

- $\sigma_{p_1 \land p_2} (R) \rightarrow \sigma_{p_1} [\sigma_{p_2} (R)]$
- $\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S$
- $R \bowtie S \rightarrow S \bowtie R$
- $\pi_x [\sigma_p (R)] \rightarrow \pi_x \{\sigma_p [\pi_{xz} (R)]\}$
Conventional wisdom: do projects early

Example: \( R(A,B,C,D,E) \times \{E\} \)
\[ P: (A=3) \land (B="\text{cat"}) \]

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

B = “cat” \[\rightarrow\] \[\rightarrow\] A=3

Intersect pointers to get pointers to matching tuples
Bottom line:

- No transformation is always good
- Usually good: early selections
In textbook: more transformations

- Eliminate common sub-expressions
- Other operations, such as, duplicate elimination and others.