Chapters 15 and 16: Query Processing

(Slides by Hector Garcia-Molina, http://www-db.stanford.edu/~hector/cs245/notes.htm)

Query Processing

Q → Query Plan

Focus: Relational System

Example

Select B,D
From R,S
Where R.A = “c”
AND S.E = 2
AND R.C=S.C
How do we execute query?

- Do Cartesian product
- Select tuples
- Do projection

Answer: \( B \mid D \)
\[
\begin{array}{cccc}
  a & 1 & 10 & 10 \times 2 \\
  b & 1 & 20 & 20 \times 2 \\
  c & 2 & 10 & 30 \times 2 \\
  d & 2 & 35 & 40 \times 1 \\
  e & 3 & 45 & 50 \times 3 \\
\end{array}
\]
Relational Algebra - can be used to describe plans...

Ex: Plan I

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

OR: \[ \Pi_{B,D} [ \sigma_{R.A = "c" \land S.E = 2 \land R.C = S.C} (RXS)] \]

Another idea:

Plan II

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

natural join

<table>
<thead>
<tr>
<th>R</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>35</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td>x</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
<td>y</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>10</td>
<td>y</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>35</td>
<td>z</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>45</td>
<td>z</td>
</tr>
</tbody>
</table>
Plan III
Use R.A and S.C Indexes
(1) Use R.A index to select R tuples with R.A = "c"
(2) For each R.C value found, use S.C index to find matching tuples
(3) Eliminate S tuples S.E ≠ 2
(4) Join matching R,S tuples, project B,D attributes and place in result

Overview of Query Optimization
Example: SQL query

```sql
SELECT title
FROM StarsIn
WHERE starName IN (
    SELECT name
    FROM MovieStar
    WHERE birthdate LIKE '%1960'
);
```

(Find the movies with stars born in 1960)

Example: Parse Tree
Example: Generating Relational Algebra

```
Πtitle

σ

StarsIn

<condition>

IN

Πname

<tuple>

<attribute>

σbirthdate LIKE '%1960'

starName

MovieStar
```

An expression using a two-argument σ midway between a parse tree and relational algebra.

Example: Logical Query Plan

```
Πtitle

σ

σ

starsIn

= name

×

StarsIn

Πname

σ

birthdate LIKE '%1960'

MovieStar
```

Applying the rule for IN conditions.

Example: Improved Logical Query Plan

```
Πtitle

σ

σ

starName = name

 StarsIn

Πname

σ

birthdate LIKE '%1960'

MovieStar
```

Fig. 7.20: An improvement on fig. 7.18.

Question: Push project to StarsIn?
**Example: Estimate Result Sizes**

- Need expected size

**Example: One Physical Plan**

- Parameters: join order, memory size, ...
- Hash join
- Parameters: Select Condition, ...
- SEQ scan
- Index scan

**Example: Estimate costs**

- \( L \cdot Q \cdot P \)
- \( P_1, P_2, ..., P_n \)
- \( C_1, C_2, ..., C_n \)
- Pick best!
Textbook outline

Chapter 5: Algebra for queries (bags vs sets)
- Select, project, join, ... [project list]
  a, a+b->x, ...]
- Duplicate elimination, grouping, sorting

Chapter 15:
- Physical operators
  - Scan, sort, ...
- Implementing operators and estimating their cost

Chapter 16:
- Parsing
- Algebraic laws
- Parse tree -> logical query plan
- Estimating result sizes
- Cost based optimization

Query Optimization

- Relational algebra level
- Detailed query plan level
  - Estimate Costs
    - without indexes
    - with indexes
  - Generate and compare plans
Relational algebra optimization

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

Rules: Natural joins & cross products & union

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

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

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

Note:

- Can also write as trees, e.g.:
**Rules: Selects**

\[
\sigma_{p_1 \land p_2}(R) = \sigma_{p_1} [ \sigma_{p_2}(R)] \\
\sigma_{p_1 \lor p_2}(R) = [ \sigma_{p_1}(R)] \cup [ \sigma_{p_2}(R)]
\]

**Rules: \(\sigma \oplus \bigodot\) combined**

Let \(p\) = predicate with only R attribs

\(q\) = predicate with only S attribs

\[
\sigma_p (R \bigodot S) = [\sigma_p (R)] \bigodot S \\
\sigma_q (R \bigodot S) = R \bigodot [\sigma_q (S)]
\]

Which are “good” transformations?

- \(\sigma_{p_1 \lor p_2}(R) \rightarrow \sigma_{p_1} [\sigma_{p_2}(R)]\)
- \(\sigma_p (R \bigodot S) \rightarrow [\sigma_p (R)] \bigodot S\)
- \(R \bigodot S \rightarrow S \bigodot R\)
Conventional wisdom: do projects early

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

$$\pi_x \{ \sigma_P (R) \} \quad \text{vs.} \quad \pi_E \{ \sigma_P (\pi_{ABE}(R)) \}$$

But What if we have $A$, $B$ indexes?

B = "cat" $\rightarrow$ 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: duplicate elimination

Outline - Query Processing

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

• Estimating cost of query plan

  (1) Estimating size of results
  (2) Estimating # of IOs
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></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>

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

T(R) = 5  S(R) = 37
V(R,A) = 3  V(R,C) = 5
V(R,B) = 1  V(R,D) = 4

Size estimates for W = R1 x R2

T(W) = T(R1) x T(R2)
S(W) = S(R1) + S(R2)
Selection cardinality

\[ SC(R,A) = \frac{T(R)}{V(R,A)} \]
\[ SC(R,A) = \begin{cases} 
\frac{T(R)}{DOM(R,A)} 
\end{cases} \]

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

\[ T(W) = ? \]

- Solution # 1:
  \[ T(W) = \frac{T(R)}{2} \]
- Solution # 2:
  \[ T(W) = \frac{T(R)}{3} \]

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

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

**Case 1**
\[ X \cap Y = \emptyset \]
Same as \( R_1 \times R_2 \)
Case 2 \[ W = R_1 \bowtie R_2 \quad X \cap Y = A \]

<table>
<thead>
<tr>
<th>R1</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>A</td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

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

“containment of value sets”

**In general** \( W = R_1 \bowtie R_2 \)

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

• \( V(R_1,A) \leq V(R_2,A) \)
• \( V(R_2,A) \leq V(R_1,A) \)

[A is common attribute]
Using similar ideas, we can estimate sizes of:

\[ \Pi_{AB} (R) \] …

\[ \sigma_{A=a\land B=b} (R) \] …

\( R \bowtie S \) with common attrs. A,B,C

Union, intersection, diff, …

Example:

\( Z = R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D) \)

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

Partial Result: \( U = R_1 \bowtie R_2 \)

\[
\frac{T(U) = 1000 \times 2000}{200} \quad \begin{array}{c}
\text{V(U,A)} = 50 \\
\text{V(U,B)} = 100 \\
\text{V(U,C)} = 300
\end{array}
\]
\[ Z = U \Join R3 \]
\[
T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300} \\
V(Z,A) = 50 \\
V(Z,B) = 100 \\
V(Z,C) = 90 \\
V(Z,D) = 500
\]

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
- Generate and compare plans