CPS216: Data-Intensive Computing Systems

Query Processing (Overview)

Shivnath Babu
Query Processing

Declarative SQL Query $\rightarrow$ Query Plan

NOTE: You will not be tested on how well you know SQL. Understanding the SQL introduced in class will be sufficient (a primer follows). SQL is described in Chapter 6, GMUW.

Focus: Relational System (i.e., data is organized as tables, or relations)
We will focus on SPJ, or Select-Project-Join Queries

Example Filter Query over $R(A,B,C)$:

```
Select   B  
From     R  
Where    R.A = "c" $\land$ R.C > 10
```
We will focus on SPJ, or Select-Project-Join-Queries.

Example Join Query over R(A,B,C) and S(C,D,E):

```
SELECT B, D
FROM R, S
WHERE R.A = "c" ∧ S.E = 2 ∧ R.C = S.C
```
Select B,D
From R,S
Where R.A = “c”  ∧
S.E = 2  ∧  R.C=S.C

Answer

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>y</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>z</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>x</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>y</td>
</tr>
</tbody>
</table>
• How do we execute this query?

Select B,D
From R,S
Where R.A = “c” \( \land \) S.E = 2 \( \land \)
R.C=S.C

One idea

- Do Cartesian product
- Select tuples
- Do projection
<table>
<thead>
<tr>
<th>R X S</th>
<th>R.A</th>
<th>R.B</th>
<th>R.C</th>
<th>S.C</th>
<th>S.D</th>
<th>S.E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>Select B,D From R,S Where R.A = “c” ( \land S.E = 2 \land R.C = S.C )</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Bingo! Got one...
Relational Algebra - can be used to describe plans

Ex: Plan I

\[ \Pi_{B,D} \left( \sigma_{R.A = "c" \land S.E = 2 \land R.C = S.C} (R \times S) \right) \]
Relational Algebra Primer
(Chapter 5, GMUW)

Select: $\sigma_{R.A=\text{“c”}} \land R.C=10$

Project: $\Pi_{B,D}$

Cartesian Product: $R \times S$

Natural Join: $R \bowtie S$
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:

Select B,D
From R,S
Where R.A = “c”  ∧
S.E = 2  ∧  R.C=S.C
Select B,D
From R,S
Where R.A = “c”  ∧
S.E = 2  ∧  R.C=S.C
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
A = "c"

check = 2?

output: <2, x>

next tuple: <c, 7, 15>
Overview of Query Processing

1. SQL query
   - Parse
     - Parse tree
     - Query rewriting
     - Logical query plan
       - Statistics
         - Physical plan generation
           - Physical query plan
             - Execute
               - Result

Query Optimization

Query Execution
Example Query

Select B,D
From R,S
Where R.A = "c" \( \land \) R.C=S.C
Example: Parse Tree

Select B,D
From R,S
Where R.A = “c” \( \land \) R.C=S.C
Along with Parsing …

• Semantic checks
  – Do the projected attributes exist in the relations in the From clause?
  – Ambiguous attributes?
  – Type checking, ex: R.A > 17.5

• Expand views
Initial Logical Plan

Relational Algebra:

\[ \Pi_{B,D} [ \sigma_{R.A=“c” \land R.C = S.C} (R \times S)] \]
Apply Rewrite Rule (1)

\[ \pi_{B,D} \]

\[ \sigma_{R.A = "c" \land R.C = S.C} \]

\[ \sigma_{R.C = S.C} \]

\[ \sigma_{R.A = "c"}(R \times S) \]

\[ \pi_{B,D} \]

\[ \Pi_{B,D} \left[ \sigma_{R.C=S.C} \left[ \sigma_{R.A="c"}(R \times S) \right] \right] \]
Apply Rewrite Rule (2)

\[ \pi_{B,D} \sigma_{R.C = S.C} \sigma_{R.A = "c"} X S \]

\[ \pi_{B,D} \sigma_{R.C = S.C} \sigma_{R.A = "c"} X R S \]

\[ \Pi_{B,D} [ \sigma_{R.C=S.C} [\sigma_{R.A="c"}(R)] X S ] \]
Apply Rewrite Rule (3)

\[ \pi_{B,D} \]
\[ \sigma_{R.C = S.C} \]
\[ \pi_{B,D} \]

\[ \sigma_{R.A = "c"} \]
\[ \sigma_{R.A = "c"} \]

Natural join

\[ \Pi_{B,D} \left[ \left[ \sigma_{R.A = "c"}(R) \right] \bowtie S \right] \]
Query rewriting

parse

parse tree

Query rewriting

statistics

Initial logical plan

Rewrite rules

Logical plan

"Best" logical plan

Physical plan generation

physical query plan

execute

result