Query Processing

Query in SQL

→

Query Plan in Algebra
Example

Data:
relation R (A, B, C)
relation S (C, D, E)

Query:
SELECT  B, D
FROM  R, S
WHERE  R.A = “c” and  S.E = 2 and  R.C=S.C
<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td>10</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
<td>20</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>10</td>
<td>30</td>
<td></td>
<td>30</td>
<td>z</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>35</td>
<td></td>
<td></td>
<td>40</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>45</td>
<td></td>
<td></td>
<td>50</td>
<td>y</td>
<td>3</td>
</tr>
</tbody>
</table>

Answer

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>x</td>
</tr>
</tbody>
</table>
• How do we execute query?

One idea

- Form **Cartesian product** of all tables in FROM-clause
- **Select** tuples that match WHERE-clause
- **Project** columns that occur in SELECT-clause
Relational Algebra - to describe plan

Ex: Plan I

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

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

Plan II

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

natural join
\[ \sigma(R) \quad \sigma(S) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</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>2</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>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>z</td>
<td>2</td>
</tr>
<tr>
<td>40</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>y</td>
<td>3</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
CS 4432 query processing - lecture 12

I_1 \rightarrow A = "c" \rightarrow I_2

\langle c, 2, 10 \rangle \rightarrow \langle 10, x, 2 \rangle

check=2?

output: \langle 2, x \rangle

next tuple: \langle c, 7, 15 \rangle
Overview of Query Optimization
SQL query

parse

parse tree

convert

logical query plan

apply laws

"improved" l.q.p

estimate result sizes

l.q.p. + sizes

consider physical plans

{P1,P2,.....}

answer

execute

pick best

statistics

estimate costs

{(P1,C1),(P2,C2)....}
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

```
<Query>
  
  <SFW>
    SELECT <SelList>
    FROM <FromList>
    WHERE <Condition>
    
    <Attribute>
    title
    <RelName>
    StarsIn
    <Tuple>
    IN
    <Query>
    
    <Attribute>
    starName

  SELECT <SelList>
  FROM <FromList>
  WHERE <Condition>
  
  <Attribute>
  name
  <RelName>
  MovieStar
  <Attribute>
  LIKE
  <Pattern>
  birthDate
  '1960'
```
Example: Generating Relational Algebra

\[ \Pi_{\text{title}} \]
\[ \sigma_{\text{StarsIn}} \]
\[ \langle \text{condition} \rangle \]
\[ \langle \text{tuple} \rangle \]
\[ \langle \text{attribute} \rangle \]
\[ \sigma_{\text{birthdate LIKE '1960'}} \]
\[ \Pi_{\text{name}} \]
\[ \text{starName} \]
\[ \text{MovieStar} \]

Fig. 7.15: An expression using a two-argument \( \sigma \), midway between a parse tree and relational algebra
Example: Logical Query Plan

\[ \Pi_{\text{title}} \]

\[ \sigma_{\text{starName} = \text{name}} \]

\[ \times \]

\[ \Pi_{\text{name}} \]

\[ \sigma_{\text{birthdate LIKE '1960'}} \]

\[ \text{MovieStar} \]

Fig. 7.18: Applying the rule for IN conditions
Example: Improved Logical Query Plan

\[ \Pi_{\text{title}} \]
\[ \sigma_{\text{birthdate LIKE '1960'}} \]
\[ \Pi_{\text{name}} \]
\[ \text{StarName = name} \]
\[ \text{StarsIn} \]
\[ \text{MovieStar} \]

Question: Push project to StarsIn?

Fig. 7.20: An improvement on fig. 7.18.
Example: Estimate Result Sizes

Need expected size

StarsIn

\[ \pi \]

\[ \sigma \]

MovieStar
Example: One Physical Plan

Hash join

SEQ scan: StarsIn

index scan: MovieStar

Parameters: join order, memory size, project attributes,...

Parameters: Select Condition,...
Example: Estimate costs

L.Q.P

P1 P2 .... Pn

C1 C2 .... Cn

Pick best!
Query Optimization

• Relational algebra level (NEXT) ...
• Detailed query plan level
  – Estimate Costs
    • without indexes
    • with indexes
  – Generate and compare plans