Introduction to Database Systems
CSE 444

Lecture 20:
Query Execution: Relational Algebra

November 16, 2007

DBMS Architecture

How does a SQL engine work?

• SQL query → relational algebra plan
• Relational algebra plan → Optimized plan
• Execute each operator of the plan

Relational Algebra

• Formalism for creating new relations from existing ones
• Its place in the big picture:

<table>
<thead>
<tr>
<th>Declarative query language</th>
<th>Algebra</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>Relational algebra</td>
<td>Relational bag algebra</td>
</tr>
</tbody>
</table>

Relational Algebra

• Five operators:
  – Union: ∪
  – Difference: -
  – Selection: σ
  – Projection: Π
  – Cartesian Product: ×
• Derived or auxiliary operators:
  – Intersection, complement
  – Joins (natural,equi-join, theta join, semi-join)
  – Renaming: ρ

1. Union and 2. Difference

• R1 ∪ R2
• Example:
  – ActiveEmployees ∪ RetiredEmployees

• R1 – R2
• Example:
  – AllEmployees -- RetiredEmployees

What about Intersection?

• It is a derived operator
• R1 ∩ R2 = R1 – (R1 – R2)
• Also expressed as a join (will see later)
• Example
  – UnionizedEmployees ∩ RetiredEmployees
3. Selection

- Returns all tuples which satisfy a condition
- Notation: $\sigma_c(R)$
- Examples:
  - $\sigma_{\text{Salary} > 40000}(\text{Employee})$
  - $\sigma_{\text{name} = \text{"Smith"}}(\text{Employee})$
- The condition $c$ can be $=, <, \leq, >, \geq, \neq$

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234545</td>
<td>John</td>
<td>200000</td>
</tr>
<tr>
<td>5423341</td>
<td>Smith</td>
<td>600000</td>
</tr>
<tr>
<td>4352342</td>
<td>Fred</td>
<td>500000</td>
</tr>
</tbody>
</table>

4. Projection

- Eliminates columns, then removes duplicates
- Notation: $\Pi_{A_1, \ldots, A_n}(R)$
- Example: project social-security number and names:
  - $\Pi_{\text{SSN, Name}}(\text{Employee})$
  - Output schema: $\text{Answer(SSN, Name)}$

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234545</td>
<td>John</td>
<td>200000</td>
</tr>
<tr>
<td>5423341</td>
<td>Smith</td>
<td>600000</td>
</tr>
<tr>
<td>4352342</td>
<td>Fred</td>
<td>500000</td>
</tr>
</tbody>
</table>

5. Cartesian Product

- Each tuple in $R_1$ with each tuple in $R_2$
- Notation: $R_1 \times R_2$
- Example:
  - $\text{Employee} \times \text{Dependents}$
  - Very rare in practice; mainly used to express joins

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>20000</td>
</tr>
<tr>
<td>John</td>
<td>60000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>999999999</td>
<td>Emily</td>
<td></td>
</tr>
<tr>
<td>777777777</td>
<td>Joe</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>999999999</td>
<td>Emily</td>
</tr>
<tr>
<td>John</td>
<td>999999999</td>
<td>Joe</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
<td>Emily</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
<td>Joe</td>
</tr>
</tbody>
</table>
Relational Algebra

• Five operators:
  – Union: $\cup$
  – Difference: $-$
  – Selection: $\sigma$
  – Projection: $\Pi$
  – Cartesian Product: $\times$

• Derived or auxiliary operators:
  – Intersection, complement
  – Joins (natural, equivalent-join, theta join, semi-join)
  – Renaming: $\rho$

Renaming

• Changes the schema, not the instance
• Notation: $\rho_{B_1, \ldots, B_n}(R)$
• Example:
  – $\rho_{\text{LastName}, \text{SocSocNo}}(\text{Employee})$
  – Output schema:
    Answer(\text{LastName}, \text{SocSocNo})

Renaming Example

<table>
<thead>
<tr>
<th>Employee</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>John</td>
</tr>
<tr>
<td>SSN</td>
<td>999999999</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
</tr>
</tbody>
</table>

$\rho_{\text{LastName}, \text{SocSocNo}}(\text{Employee})$

<table>
<thead>
<tr>
<th>LastName</th>
<th>SocSocNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>999999999</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
</tr>
</tbody>
</table>

Natural Join

• Notation: $R \bowtie S$
• Meaning: $R \bowtie S = \Pi_A(\sigma_C(R \times S))$
• Where:
  – The selection $\sigma_C$ checks equality of all common attributes
  – The projection eliminates the duplicate common attributes

Natural Join Example

<table>
<thead>
<tr>
<th>Employee</th>
<th>SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>John</td>
</tr>
<tr>
<td>SSN</td>
<td>999999999</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dependents</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>999999999</td>
<td>Emily</td>
</tr>
<tr>
<td>777777777</td>
<td>Joe</td>
</tr>
</tbody>
</table>

Employee $\bowtie$ Dependents $=$

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>999999999</td>
<td>Emily</td>
</tr>
<tr>
<td>Tony</td>
<td>777777777</td>
<td>Joe</td>
</tr>
</tbody>
</table>

Natural Join

<table>
<thead>
<tr>
<th>$R=$</th>
<th>$S=$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Z</td>
<td>V</td>
</tr>
</tbody>
</table>

$R \bowtie S =$

<table>
<thead>
<tr>
<th>$R \bowtie S=$</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>V</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
Natural Join

- Given the schemas \( R(A, B, C, D), S(A, C, E) \), what is the schema of \( R \bowtie S \) ?
- Given \( R(A, B, C), S(D, E) \), what is \( R \bowtie S \) ?
- Given \( R(A, B), S(A, B) \), what is \( R \bowtie S \) ?

Theta Join

- A join that involves a predicate
- \( R_1 \bowtie \theta R_2 = \sigma_{\theta} (R_1 \times R_2) \)
- Here \( \theta \) can be any condition

Eq-join

- A theta join where \( \theta \) is an equality
- \( R \bowtie_{A=B} R_2 = \sigma_{A=B} (R_1 \times R_2) \)
- Example:
  - Employee \( \bowtie_{SSN=SSN} \) Dependents
  - Most useful join in practice

Semijoin

- \( R \bowtie S = \Pi_{A_1, \ldots, A_n} (R \bowtie S) \)
- Where \( A_1, \ldots, A_n \) are the attributes in \( R \)
- Example:
  - Employee \( \bowtie \times \) Dependents

Semijoins in Distributed Databases

- Semijoins are used in distributed databases

Complex RA Expressions

\[ \begin{align*}
\Pi_{\text{name}} & \Pi_{\text{pid}} \\
\text{Person} & \text{Purchase} \\
\text{Person} & \text{Product} \\
\sigma_{\text{name}=\text{fred}} & \sigma_{\text{name}=\text{gizmo}} \\
\sigma_{\text{pid}=\text{pid}}
\end{align*} \]
Operations on Bags

A bag is a set with repeated elements
All operations need to be defined carefully on bags
• \( \{a,b,c\}\cup\{a,b,b,b,c,f,f\} = \{a,a,b,b,b,b,c,e,f,f\}\)
• \(\{a,b,b,c,e\} -\{b,c,c,d\} = \{a,b,d\}\)
• \(\sigma_C(R)\): preserve the number of occurrences
• \(\Pi_A(R)\): no duplicate elimination
• Cartesian product, join: no duplicate elimination

Important! Relational Engines work on bags, not sets!

Reading assignment: 5.3 – 5.4

Note: RA has Limitations!

• Cannot compute "transitive closure"

<table>
<thead>
<tr>
<th>Name1</th>
<th>Name2</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>Mary</td>
<td>Father</td>
</tr>
<tr>
<td>Mary</td>
<td>Joe</td>
<td>Cousin</td>
</tr>
<tr>
<td>Mary</td>
<td>Bill</td>
<td>Spouse</td>
</tr>
<tr>
<td>Nancy</td>
<td>Lou</td>
<td>Sister</td>
</tr>
</tbody>
</table>

• Find all direct and indirect relatives of Fred
• Cannot express in RA!!! Need to write C program

From SQL to RA

Purchase(buyer, product, city)
Person(name, age)

\[
\text{SELECT DISTINCT } P.buyer \\
\text{FROM Purchase } P, \text{ Person } Q \\
\text{WHERE } P.buyer=Q.name \text{ AND } P.city='Seattle' \text{ AND } Q.age > 20
\]

Also...

Purchase(buyer, product, city)
Person(name, age)

\[
\text{SELECT DISTINCT } P.buyer \\
\text{FROM Purchase } P, \text{ Person } Q \\
\text{WHERE } P.buyer=Q.name \text{ AND } P.city='Seattle' \text{ AND } Q.age > 20
\]

Non-monontone Queries (in class)

Purchase(buyer, product, city)
Person(name, age)

\[
\text{SELECT DISTINCT } P.product \\
\text{FROM Purchase } P \\
\text{WHERE } P.city='Seattle' \text{ AND } \text{not exists (select * from Purchase } P2, \text{ Person } Q \\
\text{where } P2.product = P.product \text{ and } P2.buyer = Q.name \text{ and } Q.age > 20)\]

Extended Logical Algebra Operators

(operate on Bags, not Sets)

• Union, intersection, difference
• Selection \(\sigma\)
• Projection \(\Pi\)
• Join \(\bowtie\)
• Duplicate elimination \(\delta\)
• Grouping \(\gamma\)
• Sorting \(\tau\)
**Logical Query Plan**

```
SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100
```

```
Π city, c
T2(city, p, c)
σ p > 100
T1(city, p, c)
γ city, sum(price) → p, count(*) → c
σ city, sum(price) > 100
T3(city, c)
```

T1, T2, T3 = temporary tables

**Logical v.s. Physical Algebra**

- We have seen the logical algebra so far:
  - Five basic operators, plus group-by, plus sort
- The Physical algebra refines each operator into a concrete algorithm

**Physical Plan**

```
SELECT DISTINCT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
  P.city='Seattle' AND
  Q.age > 20
```

```
δ buyer
σ buyer
σ age > 20
σ city='Seattle'
```

**Physical Plans Can Be Subtle**

```
SELECT *
FROM Purchase P
WHERE P.city='Seattle'
```

```
π buyer
σ buyer
δ buyer
σ buyer
σ city='Seattle'
```

Where did the join come from?