Lecture 20: Query Execution: Relational Algebra
Friday, November 17, 2006

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:

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 $=, \leq, >, \geq, \neq$

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: $\text{Answer(SSN, Name)}$

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
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

Renaming

- Changes the schema, not the instance
- Notation: ρ_{B_1,…,B_n}(R)
- Example:
  - ρ_{LastName, SocSocNo}(Employee)
  - Output schema: Answer(LastName, SocSocNo)

Renaming Example

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

ρ_{LastName, SocSocNo}(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: R1 × R2
- Meaning: R1 × R2 = Π_C(σ_C(R1 × R2))
- Where:
  - The selection σ_C checks equality of all common attributes
  - The projection eliminates the duplicate common attributes

Natural Join Example

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

<table>
<thead>
<tr>
<th>Dependents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
</tr>
<tr>
<td>999999999</td>
</tr>
<tr>
<td>777777777</td>
</tr>
</tbody>
</table>

Employee × Dependents = Π_{Name, SSN, Dname}(σ_{SSN=SSN}(Employee × ρ_{SSN, Dname}(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

- R=

               R             | B | C |
            |---|---|---|
            A | X | Y |
            | Z | Z |

- S=

               S             | B | C |
            |---|---|---|
            X | Y | U |
            | Z | V |

- R × S=

               R × S         | B | C |
            |---|---|---|
            X | Z | U |
            | Y | Z |
            | Z | V |
Natural Join

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

Theta Join

- A join that involves a predicate
  - R1 ⋈ θ R2 = σθ (R1 × R2)
  - Here θ can be any condition

Eq-join

- A theta join where θ is an equality
  - R1 ⋈ A=B R2 = σA=B (R1 × R2)
- Example:
  - Employee ⋈ SSN=SSN Dependents
  - Most useful join in practice

Semijoin

- R ⋈ S = ΠA1,…,An (R ⋈ S)
- Where A1, … , An are the attributes in R
- Example:
  - Employee ⋈ Dependents

Semijoins in Distributed Databases

- Semijoins are used in distributed databases

Complex RA Expressions

- Person σname=fred
- Purchase σname=gizmo
- T = Π SSN σage>71 (Dependents)
- Answer = R ⋈ T Dependents

- Employee ⋈ SSN=SSN (Dependents)
- R = Employee ⋈ T
- T = Π SSN σage>71 (Dependents)
- Answer = R ⋈ T Dependents

- Person
- Purchase
- Product

- Person
- Product

- Person
- Product

- Person
- Purchase
- Product
Operations on Bags

A bag = a set with repeated elements
All operations need to be defined carefully on bags
- \{a,b,b\} ∪ \{a,b,b,b,c,e,f,f\} = \{a,a,b,b,b,b,b,c,e,f,f\}
- \{a,b,b,c\} – \{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!

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

Reading assignment: 5.3 – 5.4

From SQL to RA

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

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

Extended Logical Algebra Operators

(operate on Bags, not Sets)

- Union, intersection, difference
- Selection \(\sigma\)
- Projection \(\Pi\)
- Join \(\times\)
- Duplicate elimination \(\delta\)
- Grouping \(\gamma\)
- Sorting \(\tau\)

Non-monontone Queries (in class)

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

```
SELECT DISTINCT P.product
FROM Purchase P
WHERE P.city='Seattle' AND
not exists (select * 
from Purchase P2, Person Q 
where P2.product = P.product 
and P2.buyer = Q.name 
and Q.age > 20)
```
**Logical Query Plan**

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

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

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
```

**Physical Plans Can Be Subtle**

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

Where did the join come from?