Introduction to Data Management
CSE 344

Lectures 9-10: Relational Algebra
Announcements

• Webquiz 3 due tomorrow night, 11 pm

• HW3 is out, due a week from today, BUT:
  – Log on to your Azure account by now

• HW4 will be out this Thursday

• Today’s lecture: secs. 2.4 and 5.1
Where We Are

• Motivation for using a DBMS for managing data
• SQL, SQL, SQL
  – Declaring the schema for our data (CREATE TABLE)
  – Inserting data one row at a time or in bulk (INSERT/.import)
  – Modifying the schema and updating the data (ALTER/UPDATE)
  – Querying the data (SELECT)
  – Tuning queries (CREATE INDEX)

• Next step: More knowledge of how DBMSs work
  – Client-server architecture
  – Relational algebra and query execution
Query Evaluation Steps

1. **Parse & Check Query**
   - Translate query string into internal representation
   - Check syntax, access control, table names, etc.

2. **Logical plan → physical plan**

3. **Query Execution**

4. **Return Results**

---

CSE 344 - Winter 2015
The WHAT and the HOW

• SQL = **WHAT** we want to get form the data

• Relational Algebra = **HOW** to get the data we want

• The passage from **WHAT** to **HOW** is called query optimization
Overview: SQL = WHAT

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
    x.price > 100 and z.city = 'Seattle'

It's clear WHAT we want, unclear HOW to get it
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
    x.price > 100 and
    z.city = 'Seattle'

Execution order is now clearly specified

But a lot of physical details are still left open!
Relational Algebra
Edgar Frank “Ted” Codd

"A Relational Model of Data for Large Shared Data Banks"
Sets v.s. Bags

- Sets: \{a,b,c\}, \{a,d,e,f\}, \{\}\ldots
- Bags: \{a, a, b, c\}, \{b, b, b, b, b\}, \ldots

Relational Algebra has two semantics:
- Set semantics = standard Relational Algebra
- Bag semantics = extended Relational Algebra

DB systems implement bag semantics (Why?)
Relational Algebra Operators

- Union $\cup$, intersection $\cap$, difference $-$
- Selection $\sigma$
- Projection $\Pi$
- Cartesian product $\times$, join $\Join$
- Rename $\rho$
- Duplicate elimination $\delta$
- Grouping and aggregation $\gamma$
- Sorting $\tau$

Extended RA

CSE 344 - Winter 2015
Union and Difference

$R1 \cup R2$

$R1 - R2$

What do they mean over bags?
What about Intersection?

- Derived operator using minus
  \[ R_1 \cap R_2 = R_1 - (R_1 - R_2) \]

- Derived using join (will explain later)
  \[ R_1 \cap R_2 = R_1 \bowtie R_2 \]
Selection

• Returns all tuples which satisfy a condition

\[ \sigma_c(R) \]

• Examples

  – \( \sigma_{\text{Salary} > 40000} \) (Employee)
  
  – \( \sigma_{\text{name} = “Smith”} \) (Employee)

• The condition c can be =, <, ≤, >, ≥, <>
### Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234545</td>
<td>John</td>
<td>20000</td>
</tr>
<tr>
<td>5423341</td>
<td>Smith</td>
<td>60000</td>
</tr>
<tr>
<td>4352342</td>
<td>Fred</td>
<td>50000</td>
</tr>
</tbody>
</table>

\[ \sigma_{\text{Salary} > 40000} (\text{Employee}) \]

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>5423341</td>
<td>Smith</td>
<td>60000</td>
</tr>
<tr>
<td>4352342</td>
<td>Fred</td>
<td>50000</td>
</tr>
</tbody>
</table>
Projection

• Eliminates columns

\[ \Pi_{A_1, \ldots, A_n}(R) \]

• Example: project social-security number and names:
  – \[ \Pi_{SSN, Name}(Employee) \]
  – Answer(SSN, Name)

Different semantics over sets or bags! Why?
Which is more efficient?
Composing RA Operators

\[ \sigma_{\text{disease} = \text{'heart'}}(\text{Patient}) \]

\[
\begin{array}{cccc}
\text{no} & \text{name} & \text{zip} & \text{disease} \\
1 & p1 & 98125 & \text{flu} \\
2 & p2 & 98125 & \text{heart} \\
3 & p3 & 98120 & \text{lung} \\
4 & p4 & 98120 & \text{heart} \\
\end{array}
\]

\[ \pi_{\text{zip}, \text{disease}}(\text{Patient}) \]

\[
\begin{array}{cc}
\text{zip} & \text{disease} \\
98125 & \text{flu} \\
98125 & \text{heart} \\
98120 & \text{lung} \\
98120 & \text{heart} \\
\end{array}
\]

\[ \pi_{\text{zip}}(\sigma_{\text{disease} = \text{'heart'}}(\text{Patient})) \]

\[
\begin{array}{c}
\text{zip} \\
98120 \\
98125 \\
\end{array}
\]
Cartesian Product

- Each tuple in R1 with each tuple in R2

\[ R1 \times R2 \]

- Rare in practice; mainly used to express joins
## Cross-Product Example

### Employee

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

### Dependent

<table>
<thead>
<tr>
<th>EmpSSN</th>
<th>DepName</th>
</tr>
</thead>
<tbody>
<tr>
<td>999999999</td>
<td>Emily</td>
</tr>
<tr>
<td>777777777</td>
<td>Joe</td>
</tr>
</tbody>
</table>

### Employee × Dependent

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

• Changes the schema, not the instance

\[ \rho_{B_1, \ldots, B_n}(R) \]

• Example:
  - \( \rho_{N, S}(\text{Employee}) \rightarrow \text{Answer}(N, S) \)

Not really used by systems, but needed on paper
Natural Join

\[ R_1 \Join R_2 \]

- **Meaning:** \( R_1 \Join R_2 = \Pi_A(\sigma(R_1 \times R_2)) \)

- **Where:**
  - Selection \( \sigma \) checks equality of all common attributes
  - Projection eliminates duplicate all common attributes
Natural Join Example

R × S = Π_{ABC}(σ_{R.B=S.B}(R × S))
### Natural Join Example 2

#### AnonPatient $P$

<table>
<thead>
<tr>
<th>age</th>
<th>zip</th>
<th>disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flu</td>
</tr>
</tbody>
</table>

#### Voters $V$

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>p2</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

$P \bowtie V$

<table>
<thead>
<tr>
<th>age</th>
<th>zip</th>
<th>disease</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
<td>p1</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flu</td>
<td>p2</td>
</tr>
</tbody>
</table>
Natural Join

• Given 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 $R \bowtie S$?