Introduction to Data Management
CSE 344
Lecture 8: Relational Algebra

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

Data Management with SQLite

- So far, we have been managing data with SQLite as follows:
  - One data file
  - One user
  - One DBMS application
- But only a limited number of scenarios work with such model

Client-Server Architecture

- There is a single server that stores the database (called DBMS or RDBMS):
  - Usually a beefy system, e.g., IISQLSRV1
  - But can be your own desktop...
  - ... or a huge cluster running a parallel dbms
- Many clients run apps and connect to DBMS
  - E.g., Microsoft’s Management Studio
  - Or psql (for postgres)
  - More realistically some Java or C++ program
- Clients “talk” to server using JDBC protocol

Using a DBMS Server

1. Client application establishes connection to server
2. Client must authenticate self
3. Client submits SQL commands to server
4. Server executes commands and returns results
Query Evaluation Steps Review

SQL query

Translate query string into internal representation

Parse & Check Query

Check syntax, access control, table names, etc.

Decide how best to answer query: query optimization

Query Execution

Return Results

Question: How does Query Evaluation Work?

Key Part of Answer: Relational Algebra

• Motivation and sets v.s. bags
• Relational Algebra

The WHAT and the HOW

• In SQL, we write WHAT we want to get from the data
• The database system needs to figure out HOW to get the data we want
• The passage from WHAT to HOW goes through the Relational Algebra

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

Relational Algebra = HOW

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

Final answer

Temporary tables T1, T2, ...
Relational Algebra = HOW

The order is now clearly specified:
• Iterate over PRODUCT…
• …join with PURCHASE…
• …join with CUSTOMER…
• …select tuples with Price>100 and City='Seattle'…
• …eliminate duplicates…
• …and that’s the final answer!

Relations

• A relation is a set of tuples
  – Sets: (a,b,c), (a,d,e,f), ( ) . . .

• But, commercial DBMS’s implement relations that are bags rather than sets
  – Bags: (a, a, b, c), (b, b, b, b, b) . . .

Sets v.s. Bags

Relational Algebra has two flavors:
• Over sets: theoretically elegant but limited
• Over bags: needed for SQL queries + more efficient
  – Example: Compute average price of all products

We discuss set semantics
• We mention bag semantics only where needed

Outline

• Motivation and sets v.s. bags
• Relational Algebra

Relational Algebra

• Query language associated with relational model

• Queries specified in an operational manner
  – A query gives a step-by-step procedure

• Relational operators
  – Take one or two relation instances as argument
  – Return one relation instance as result
  – Easy to compose into relational algebra expressions

Relational Algebra (1/3)

Five basic operators:
• Union (∪) and Set difference (−)
• Selection: σ_{condition}(S)
  – Condition is Boolean combination (∧, ∨) of terms
  – Term is: attribute op constant, attr. op attr.
  – Op is: <, <=, =, ≠, >=, or >
• Projection: \Pi_{list-of-attributes}(S)
• Cross-product or cartesian product (×)
Relational Algebra (2/3)

Derived or auxiliary operators:
- Intersection ($\cap$), Division ($R/S$)
- Join: $R \times S = \pi_{R(S)}$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B1,\ldots,Bn}(S)$

Relational Algebra (3/3)

Extensions for bags
- Duplicate elimination: $\delta$
- Group by: $\gamma$ [Same symbol as aggregation]
  - Partitions tuples of a relation into "groups"
- Sorting: $\tau$

Other extensions
- Aggregation: $\gamma$ (min, max, sum, average, count)

Union and Difference

- $R1 \cup R2$
- Example:
  - ActiveEmployees $\cup$ RetiredEmployees

- $R1 - R2$
- Example:
  - AllEmployees $-$ RetiredEmployees

Be careful when applying to bags!

What about Intersection?

- It is a derived operator
- $R1 \cap R2 = R1 - (R1 - R2)$
- Also expressed as a join (will see later)
- Example
  - UnionizedEmployees $\cap$ RetiredEmployees

Selection

- Returns all tuples that satisfy a condition
- Notation: $\sigma_c(R)$
- Examples
  - $\sigma_{Salary > 40000}(Employee)$
  - $\sigma_{Name = 'Smith'}(Employee)$
- The condition $c$ can be
  - Boolean combination ($\land, \lor$) of terms
  - Term is: attribute $\text{op}$ constant, attr. $\text{op}$ attr.
  - Op is: $<$, $<=$, $=$, $\neq$, $>$, $>$=
- Projection: $\pi_{\text{list-of-attributes}}(S)$
- Cross-product or cartesian product ($\times$)
Projection

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

Semantics differs over set or over bags

Set semantics: duplicate elimination automatic

Bag semantics: no duplicate elimination; need explicit \( \delta \)

Selection & Projection Examples

Relational Algebra (1/3)

Five basic operators:
- Union (\( \cup \)) and Set difference (\( \setminus \))
- Selection: \( \sigma_{\text{condition}}(S) \)
  - Condition is Boolean combination (\( \land, \lor \)) of terms
  - Term is: attribute op constant, attr. op attr.
  - Op is: \( <, \leq, =, \neq, \geq, > \)
- Projection: \( \pi_{\text{list-of-attributes}}(S) \)
- Cross-product or cartesian product (\( \times \))
Cartesian Product

- Each tuple in R1 with each tuple in R2
- Notation: $R_1 \times R_2$
- Example:
  - Employee \times Dependents
- Rare in practice; mainly used to express joins

Cross-Product Example

<table>
<thead>
<tr>
<th>AnonPatient $P$</th>
<th>Voters $V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>zip</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

$P \times V$

<table>
<thead>
<tr>
<th>P.age</th>
<th>P.zip</th>
<th>P.disease</th>
<th>V.name</th>
<th>V.age</th>
<th>V.zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
<td>p1</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
<td>p2</td>
<td>20</td>
<td>98120</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flue</td>
<td>p1</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flue</td>
<td>p2</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

Relational Algebra (2/3)

Derived or auxiliary operators:
- Intersection ($\cap$), Division ($R/S$)
- Join: $R_{\theta S} = \sigma_{\theta}(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B_1, \ldots, B_n}(S)$

Renaming

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

Renaming Example

<table>
<thead>
<tr>
<th>Employee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SSN</td>
</tr>
<tr>
<td>John</td>
<td>999999999</td>
</tr>
<tr>
<td>Tony</td>
<td>????77777</td>
</tr>
</tbody>
</table>

$\rho_{Lastname, SocSocNo}(Employee)$

<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>????77777</td>
</tr>
</tbody>
</table>

Relational Algebra (2/3)

Derived or auxiliary operators:
- Intersection ($\cap$), Division ($R/S$)
- Join: $R_{\theta S} = \sigma_{\theta}(R \times S)$
- Variations of joins
  - Natural, equijoin, theta-join
  - Outer join and semi-join
- Rename $\rho_{B_1, \ldots, B_n}(S)$
Different Types of Join

- **Theta-join**: $R \theta S = \sigma_\theta (R \times S)$
  - Join of $R$ and $S$ with a join condition $\theta$
  - Cross-product followed by selection $\sigma_\theta$

- **Equijoin**: $R \rho S = \pi_A (\sigma_\theta (R \times S))$
  - Join condition $\theta$ consists only of equalities
  - Projection $\pi_A$ drops all redundant attributes

- **Natural join**: $R \Join S = \pi_A (\sigma_\theta (R \times S))$
  - Equijoin
  - Equality on all fields with same name in $R$ and in $S$

Theta-Join Example

<table>
<thead>
<tr>
<th>AnonPatient $P$</th>
<th>AnnonJob $J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>zip</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

$P \Join J$

<table>
<thead>
<tr>
<th>job</th>
<th>age</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>cashier</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

Equijoin Example

<table>
<thead>
<tr>
<th>AnonPatient $P$</th>
<th>AnnonJob $J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>zip</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

$P \rho J$

<table>
<thead>
<tr>
<th>job</th>
<th>age</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>cashier</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

Natural Join Example

<table>
<thead>
<tr>
<th>AnonPatient $P$</th>
<th>AnnonJob $J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>zip</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

$P \Join J$

<table>
<thead>
<tr>
<th>job</th>
<th>age</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>cashier</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

So Which Join Is It?

- When we write $R \rho S$ we usually mean an equijoin, but we often omit the equality predicate when it is clear from the context

More Joins

- **Outer join**
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes

- **Variants**
  - Left outer join
  - Right outer join
  - Full outer join
Outer Join Example

<table>
<thead>
<tr>
<th>AnonPatient P</th>
<th>AnonJob J</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>zip</td>
</tr>
<tr>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
</tr>
<tr>
<td>33</td>
<td>98120</td>
</tr>
</tbody>
</table>

Example of Algebra Queries

Q1: Jobs of patients who have heart disease
\[ \pi_{\text{job}}(\text{AnnonJob} \sigma_{\text{disease} = \text{heart}}(\text{AnonPatient})) \]

Extended Operators of Relational Algebra

- Duplicate elimination (\(\delta\))
  - Since commercial DBMSs operate on multisets not sets
- Aggregate operators (\(\gamma\))
  - Min, max, sum, average, count
- Grouping operators (\(\gamma\))
  - Partitions tuples of a relation into “groups”
  - Aggregates can then be applied to groups
- Sort operator (\(\tau\))

RA Expressions v.s. Programs

- An Algebra Expression is like a program
  - Several operations
  - Strictly specified order
- But Algebra expressions have limitations

More Examples

Q2: Name of supplier of parts with size greater than 10
\[ \pi_{\text{sname}}(\text{Supplier} \sigma_{\text{psize} > 10}(\text{Part})) \]

Q3: Name of supplier of red parts or parts with size greater than 10
\[ \pi_{\text{sname}}(\text{Supplier} \sigma_{\text{psize} > 10}(\text{Part}) \cup \sigma_{\text{pcolor} = \text{red}}(\text{Part})) \]
RA and Transitive Closure

- Cannot compute “transitive closure”
- Find all direct and indirect relatives of Fred
- Cannot express in RA!!! Need to write Java program

<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>

Outline

- Motivation and sets v.s. bags
- Relational Algebra