Relational Algebra 1

Week 3 - 2
Relational Query Languages

• *Query languages*: Allow manipulation and retrieval of data from a database.

• Relational model supports simple, powerful QLs:
  – Strong formal foundation based on logic.
  – Allows for much optimization.

• Query Languages != programming languages!
  – QLs not expected to be “Turing complete”.
  – QLs not intended to be used for complex calculations.
  – QLs support easy, efficient access to large data sets.
Formal Relational Query Languages

Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:

1. **Relational Algebra**: More operational, very useful for representing execution plans.

2. **Relational Calculus**: Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)

❗️ Understanding Algebra is key to understanding SQL, and query processing!
The Role of Relational Algebra in a DBMS

SQL Query

Parser

Relational Algebra Expression

Query Optimizer

Query Execution Plan

Code Generator

Executable Code
Algebra Preliminaries

• A query is applied to *relation instances*, and the result of a query is also a relation instance.
  – *Schemas* of input relations for a query are *fixed* (but query will run regardless of instance!)
  – The *schema for the result* of a given query is also *fixed*! Determined by definition of query language constructs.
Relational Algebra

- Procedural language
- Five basic operators
  - selection
  - projection
  - union
  - set difference
  - Cross product
- SQL is closely based on relational algebra.

- The are some other operators which are composed of the above operators. These show up so often that we give them special names.
- The operators take one or two relations as inputs and give a new relation as a result.
Select Operation – Example

• Relation $r$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\beta$</td>
<td>23</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Intuition: The select operation allows us to retrieve some rows of a relation (by “some” I mean anywhere from none of them to all of them)

Here I have retrieved all the rows of the relation $r$ where the value in field $A$ equals the value in field $B$, and the value in field $D$ is greater than 5.
Select Operation

• Notation: \( \sigma_p(r) \)

• \( p \) is called the selection predicate

• Defined as:

\[
\sigma_p(r) = \{ t \mid t \in r \text{ and } p(t) \}
\]

Where \( p \) is a formula in propositional calculus consisting of terms connected by: \( \land \) (and), \( \lor \) (or), \( \neg \) (not)

Each term is one of:

\(<\text{attribute}> \ op \ <\text{attribute}> \text{ or } <\text{constant}>\)

where \( op \) is one of: \( =, \neq, >, \geq, <, \leq \)

• Example of selection:

\( \sigma_{\text{name}=\text{‘Lee’}}(\text{professor}) \)
Project Operation – Example I

- Relation $r$:

- $\pi_{A,C}(r)$

Intuition: The **project** operation allows us to retrieve some columns of a relation (by “some” I mean anywhere from none of them to all of them)

Here I have retrieved columns $A$ and $C$. 

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>10</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>20</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>30</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>40</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Greek lower-case pi
Project Operation – Example II

• Relation $r$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>10</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>30</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>40</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Intuition: The project operation removes duplicate rows, since relations are sets.

• $\pi_{A,C}(r)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

= 

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Here there are two rows with $A = \alpha$ and $C = 1$. So one was discarded.
Project Operation

• Notation:

$$\pi_{A_1, A_2, \ldots, A_k}(r)$$

where $A_1, A_2$ are attribute names and $r$ is a relation name.

• The result is defined as the relation of $k$ columns obtained by erasing the columns that are not listed.

• Duplicate rows removed from result, since relations are sets.
Union Operation – Example

Relations $r$, $s$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>3</td>
</tr>
</tbody>
</table>

Intuition: The union operation concatenates two relations, and removes duplicate rows (since relations are sets).

Here there are two rows with $A = \alpha$ and $B = 2$. So one was discarded.
Union Operation

- Notation: \( r \cup s \)
- Defined as:

\[
r \cup s = \{ t | t \in r \text{ or } t \in s \}\]

For \( r \cup s \) to be valid:

1. \( r, s \) must have the same arity (same number of attributes)
2. The attribute domains must be compatible (e.g., 2\textsuperscript{nd} column of \( r \) deals with the same type of values as does the 2\textsuperscript{nd} column of \( s \)).

Although the field types must be the same, the names can be different. For example I can union professor and lecturer where:

\[
\begin{align*}
\text{professor}(\text{PID} & : \text{string}, \text{name} : \text{string}) \\
\text{lecturer}(\text{LID} & : \text{string}, \text{first\_name} : \text{string})
\end{align*}
\]
Related Operation: Intersection

Relations \( r, s:\)

\[
\begin{array}{cc}
A & B \\
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\end{array}
\]

\( r \)

\[
\begin{array}{cc}
A & B \\
\alpha & 2 \\
\beta & 3 \\
\end{array}
\]

\( s \)

\[ r \cap s: \]

\[
\begin{array}{cc}
A & B \\
\alpha & 2 \\
\end{array}
\]

- Similar to Union operation.
- But Intersection is NOT one of the five basic operations.
- **Intuition**: The *intersection* operation computes the common rows between two relations.
Set Difference Operation – Example

Relations $r, s$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>

Intuition: The set difference operation returns all the rows that are in $r$ but not in $s$.

$r - s$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>$\beta$</td>
<td>1</td>
</tr>
</tbody>
</table>
Set Difference Operation

• Notation $r - s$
• Defined as:
  $$r - s = \{ t \mid t \in r \text{ and } t \notin s \}$$
• Set differences must be taken between compatible relations.
  “Union-compatible”
  – $r$ and $s$ must have the same arity
  – attribute domains of $r$ and $s$ must be compatible
• Note that in general $r - s \neq s - r$