SQL and Relational Algebra

Zaki Malik

September 02, 2008
Basics of Relational Algebra

• Four types of operators:
  – Select/Show parts of a single relation: projection and selection.
  – Usual set operations (union, intersection, difference).
  – Combine the tuples of two relations, such as cartesian product and joins.
  – Renaming.
Projection

• The projection operator produces from a relation R a new relation containing only some of R’s columns.

• “Delete” (i.e. not show) attributes not in projection list.
• Duplicates eliminated

• To obtain a relation containing only the columns $A_1,A_2, \ldots A_n$ of R

\[ RA: \quad \pi A_1,A_2, \ldots A_n (R) \]

\[ \text{SQL: } \quad \text{SELECT } A_1,A_2, \ldots A_n \text{ FROM } R; \]
Selection

• The selection operator applied to a relation R produces a new relation with a subset of R’s tuples.
• The tuples in the resulting relation satisfy some condition C that involves the attributes of R.
  – with duplicate removal

\[ RA: \sigma_C(R) \]

SQL: \texttt{SELECT * FROM R WHERE C;}

• The WHERE clause of a SQL command corresponds to \( \sigma( ) \).
Union

• The union of two relations R and S is the set of tuples that are in R or in S or in both.
  – R and S must have identical sets of attributes and the types of the attributes must be the same.
  – The attributes of R and S must occur in the same order.

• What is the schema of the result?

  RA: \( R \cup S \)

  SQL: \( (\text{SELECT } * \text{ FROM } R) \)
  \( \text{UNION} \)
  \( (\text{SELECT } * \text{ FROM } S); \)
# Union

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>pupper</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>pupper</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

\[
S_1 \cup S_2
\]

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>44</td>
<td>pupper</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>28</td>
<td>pupper</td>
<td>9</td>
<td>35.0</td>
</tr>
</tbody>
</table>
Intersection

• The intersection of two relations R and S is the set of tuples that are in both R and S.

• Same conditions hold on R and S as for the union operator.
  – R and S must have identical sets of attributes and the types of the attributes must be the same.
  – The attributes of R and S must occur in the same order.

RA: \( R \cap S \)

SQL: 
\[
(\text{SELECT} \ * \ \text{FROM} \ R) \\
\text{INTERSECT} \\
(\text{SELECT} \ * \ \text{FROM} \ S);
\]
Intersection

### $S_1$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### $S_2$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>yuppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### $S_1 \cap S_2$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>
Difference

• The difference of two relations $R$ and $S$ is the set of tuples that are \textit{in $R$ but not in $S$}.

• Same conditions hold on $R$ and $S$ as for the union operator.
  – $R$ and $S$ must have \textit{identical sets of attributes} and the types of the attributes must be the same.
  – The attributes of $R$ and $S$ must occur in the \textit{same order}.

\textbf{RA:} \quad R - S

\textbf{SQL:} \quad (\text{SELECT} \ast \text{ FROM } R) \text{ EXCEPT} (\text{SELECT} \ast \text{ FROM } S);

• $R - (R - S) = R \cap S$
## Difference

### $S_1$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### $S_2$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>uppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### $S_1 - S_2$

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
</tbody>
</table>
Cartesian Product

• The Cartesian product (or cross-product or product) of two relations R and S is the set of pairs that can be formed by pairing each tuple of R with each tuple of S.
  – The result is a relation whose schema is the schema for R followed by the schema for S.

RA: \( R \times S \)

SQL: `SELECT * FROM R, S;`
# Cartesian Product

- **$S1$**
  - **sid**
  - **sname**
  - **rating**
  - **age**
  - | sid | sname | rating | age |
  - |-----|-------|--------|-----|
  - | 22  | dustin | 7      | 45.0 |
  - | 31  | lubber | 8      | 55.5 |
  - | 58  | rusty  | 10     | 35.0 |

- **$R1$**
  - **sid**
  - **bid**
  - **day**
  - | sid | bid  | day     |
  - |-----|------|--------|
  - | 22  | 101  | 10/10/96 |
  - | 58  | 103  | 11/12/96 |

- **$S1 \times R1$**
  - **(sid)**
  - **sname**
  - **rating**
  - **age**
  - **(sid)**
  - **bid**
  - **day**
  - | (sid) | sname | rating | age | (sid) | bid  | day     |
  - |-------|-------|--------|-----|-------|------|--------|
  - | 22    | dustin | 7      | 45.0 | 22    | 101  | 10/10/96 |
  - | 22    | dustin | 7      | 45.0 | 58    | 103  | 11/12/96 |
  - | 31    | lubber | 8      | 55.5 | 31    | lubber | 8     | 55.5 |
  - | 31    | lubber | 8      | 55.5 | 58    | 103  | 11/12/96 |
  - | 58    | rusty  | 10     | 35.0 | 58    | 103  | 11/12/96 |
  - | 58    | rusty  | 10     | 35.0 | 58    | 103  | 11/12/96 |

We **rename** attributes to avoid ambiguity or we **prefix** attribute with the name of the relation it belongs to.
Theta-Join

• The theta-join of two relations R and S is the set of tuples in the Cartesian product of R and S that satisfy some condition C.

\[ \theta_{C} R \bowtie S = \sigma_{C} (R \times S) \]

RA:

\[ R \bowtie_{C} S \]

SQL:

\[ \text{SELECT } * \]
\[ \text{FROM } R , S \]
\[ \text{WHERE } C ; \]
 Theta-Join

\[ S1 \bowtie R1 \quad S1.sid < R1.sid \]

\[ R \bowtie_c S = \sigma_c (R \times S) \]
Natural Join

• The natural join of two relations R and S is a set of pairs of tuples, one from R and one from S, that agree on whatever attributes are common to the schemas of R and S.

• The schema for the result contains the union of the attributes of R and S.

• Assume the schemas $R(A,B,C)$ and $S(B,C,D)$

RA: $R \bowtie S$

SQL: SELECT *
    FROM $R$, $S$
    WHERE $R.B = S.B$ AND $R.C = S.C$;
Operators Covered So far

- **Remove parts of a single relation:**
  - projection: \( \pi_{A,B}(R) \) and \( \text{SELECT A, B FROM R} \).
  - selection: \( \sigma_C(R) \) and \( \text{SELECT * FROM R WHERE C} \).
  - combining projection and selection:
    - \( \pi_{A,B}(\sigma_C(R)) \)
    - \( \text{SELECT A, B FROM R WHERE C} \).

- **Set operations** (\( R \) and \( S \) must have the same attributes, same attribute types, and same order of attributes):
  - union: \( R \cup S \) and \( (R) \text{ UNION } (S) \).
  - intersection: \( R \cap S \) and \( (R) \text{ INTERSECT } (S) \).
  - difference: \( R - S \) and \( (R) \text{ EXCEPT } (S) \).

- **Combine the tuples of two relations:**
  - Cartesian product: \( R \times S \) and \( \ldots \text{ FROM R, S } \ldots \).
  - Theta-join: \( R \bowtie_C S \) and \( \ldots \text{ FROM R, S WHERE C} \).
  - Natural join: \( R \bowtie S \); in SQL, list the conditions that the common attributes be equal in the WHERE clause.
Renaming

• If two relations have the same attribute, disambiguate the attributes by prefixing the attribute with the name of the relation it belongs to.

• How do we answer the query “Name pairs of students who live at the same address”? Students(Name, Address)
  – We need to take the cross-product of Students with itself?
  – How do we refer to the two “copies” of Students?
  – Use the rename operator.

RA: \( \rho_S (A_1, A_2, \ldots, A_n)(R) \) : give R the name S; R has n attributes, which are called A1, A2, \ldots, An in S

SQL: Use the AS keyword in the FROM clause: Students AS Students1 renames Students to Students1.

SQL: Use the AS keyword in the SELECT clause to rename attributes.
Renaming

Name pairs of students who live at the same address.

RA $\pi_{S1.Name,S2.Name}(\sigma_{S1.Address = S2.Address}(\rho_{S1}(Students) \times \rho_{S2}(Students)))$.

SQL

```
SELECT S1.name, S2.name
FROM Students AS S1, Students AS S2
WHERE S1.address = S2.address;
```

- Are these correct?
- **No!!!** the result includes tuples where a student is paired with himself/herself
- **Solution:** Add the condition $S1.name <> S2.name$. 
Practicing Relational Algebra
Q1: Find names of sailors who have reserved boat #103

Reserves(sid, bid, day)
Sailors(sid, sname, rating, age)

• Solution 1:
  \[ \pi_{sname}(\sigma_{bid = 103} (\text{Reserves} \bowtie \text{Sailors})) \]

• Solution 2 (more efficient)
  \[ \pi_{sname}((\sigma_{bid = 103} \text{Reserves}) \bowtie \text{Sailors}) \]

• Solution 3 (using rename operator)
  \[ P(\text{Temp1 } (\sigma_{bid = 103} \text{Reserves})) \]
  \[ P(\text{Temp2 } (\text{Temp1} \bowtie \text{Sailors})) \]
  \[ \pi_{sname}(\text{Temp2}) \]
Q2: Find names of sailors who have reserved a red boat

Reserves(sid, bid, day)  Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

• Solution 1:
\[ \pi_{sname}(\sigma_{\text{color} = \text{'red'}} \ Boats \bowtie Reserves \bowtie Sailors ) \]

• Solution 2 (more efficient)
\[ \pi_{sname}(\pi_{sid}(\pi_{bid}(\sigma_{\text{color} = \text{'red'}} \ Boats)\bowtie Reserves )\bowtie Sailors ) \]
Q3: Find the colors of boats reserved by Lubber

Reserves(sid, bid, day)  Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

• Solution:
  \[ \pi_{\text{color}}\left((\sigma_{\text{sname} = \text{Lubber}} \text{Sailor}) \bowtie \text{Reserves} \bowtie \text{Boats}\right) \]
Q4: Find the names of sailors who have reserved at least one boat

\[ \text{Reserves}(sid, bid, day) \quad \text{Sailors}(sid, sname, rating, age) \]
\[ \text{Boats}(bid, bname, color) \]

- Solution:
  \[ \pi_{sname}(\text{Sailor}^\infty \text{ Reserves}) \]
Q5: Find the names of sailors who have reserved a red or a green boat

Reserves(sid, bid, day)  Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

• Solution:
  \[ \pi_{sname}(\sigma_{\text{color}=\text{red} \text{ or color}=\text{green}} \ Boats \bowtie\bowtie \ Reserves \bowtie\bowtie \ Sailors) \]
Q6: Find the names of sailors who have reserved a red and a green boat

Reserves(sid, bid, day) Sailors(sid, sname, rating, age)
Boats(bid, bname, color)

Solution:

\[ \pi_{\text{sname}}(\sigma_{\text{color} = 'red' \text{ and } \text{color} = 'green'} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \]

A ship cannot have TWO colors at the same time

\[ \pi_{\text{sname}}(\sigma_{\text{color} = 'red'} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \cap \pi_{\text{sname}}(\sigma_{\text{color} = 'green'} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \]
Q7: Find the sids of sailors with age over 20 who have not reserved a red boat

\[ \pi_{\text{sid}} (\sigma_{\text{age}>20} \text{Sailors}) - \pi_{\text{sid}} ((\sigma_{\text{color}=\text{red}} \text{Boats}) \bowtie \text{Reserves}) \]

Strategy:
- Find all sailors (sids) with age over 20
- Find all sailors (sids) who have reserved a red boat
- Take their set difference