Relational Algebra 1

Week 4
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
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

• 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.

SQL is closely based on relational algebra.
Select Operation – Example

• Relation \( r \)

\[
\begin{array}{cccc}
A & B & C & D \\
\alpha & \alpha & 1 & 7 \\
\alpha & \beta & 5 & 7 \\
\beta & \beta & 12 & 3 \\
\beta & \beta & 23 & 10 \\
\end{array}
\]

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.

\[
\begin{array}{cccc}
A & B & C & D \\
\alpha & \alpha & 1 & 7 \\
\beta & \beta & 23 & 10 \\
\end{array}
\]
Select Operation

- Notation: \( \sigma_p(r) \)
- \( p \) is called the **selection** predicate
- Defined as:

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

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

Each term is one of:

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

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

- Example of selection:

\[
\sigma_{\text{name}= 'Lee'}(\text{professor})
\]
Project Operation – Example I

• 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>7</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 allows us to retrieve some columns of a relation (by “some” I mean anywhere from none of them to all of them)

• $\pi_{A,C}(r)$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>7</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>

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

Greek lower-case pi
Project Operation – Example II

• Relation $r$:

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

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

• $\pi_{A,C}(r)$

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

= 

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

$\begin{array}{|c|c|} 
\hline
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\hline
\end{array}$

$\begin{array}{|c|c|} 
\hline
A & B \\
\hline
\alpha & 2 \\
\beta & 3 \\
\hline
\end{array}$

$\begin{array}{|c|c|} 
\hline
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\beta & 3 \\
\hline
\end{array}$

$\begin{array}{|c|c|} 
\hline
A & B \\
\hline
\alpha & 2 \\
\beta & 3 \\
\hline
\end{array}$

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:

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

“Union-compatible”

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\(^{nd}\) column of \( r \) deals with the same type of values as does the 2\(^{nd}\) 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:

\[
\text{professor}(\text{PID} : \text{string}, \text{name} : \text{string})
\]
\[
\text{lecturer}(\text{LID} : \text{string}, \text{first}_\text{name} : \text{string})
\]
Related Operation: Intersection

Relations $r, s$:

\[
\begin{array}{cc}
A & B \\
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\end{array}
\quad
\begin{array}{cc}
A & B \\
\alpha & 2 \\
\beta & 3 \\
\end{array}
\]

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

- 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 $r, s$:

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\hline
\end{array}
\quad
\begin{array}{|c|c|}
\hline
A & B \\
\hline
\alpha & 2 \\
\beta & 3 \\
\hline
\end{array}
\]

$r - s$:

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
\alpha & 1 \\
\beta & 1 \\
\hline
\end{array}
\]

Intuition: The set difference operation returns all the rows that are in $r$ but not in $s$. 
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.
  - $r$ and $s$ must have the *same arity*
  - attribute domains of $r$ and $s$ must be compatible

- Note that in general $r - s \neq s - r$
### Cross-Product 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>$\beta$</td>
<td>2</td>
</tr>
</tbody>
</table>

$\mathbf{r}$

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>

$\mathbf{s}$

#### $r \times s$:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\alpha$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>10</td>
<td>a</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\beta$</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>$\beta$</td>
<td>2</td>
<td>$\gamma$</td>
<td>10</td>
<td>b</td>
</tr>
</tbody>
</table>

**Intuition:** The **cross product** operation returns all possible combinations of rows in $\mathbf{r}$ with rows in $\mathbf{s}$.

In other words the result is every possible pairing of the rows of $\mathbf{r}$ and $\mathbf{s}$.
Cross-Product Operation

• Notation $r \times s$
• Defined as:

$$r \times s = \{ t, q \mid t \in r \text{ and } q \in s \}$$

• Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).
• If attributes names of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.
Composition of Operations

- We can build expressions using multiple operations.
- Example: \( \sigma_{A=C}(r \times s) \)

```
\begin{array}{|c|c|}
\hline
A & B \\
\hline
\alpha & 1 \\
\beta & 2 \\
\hline
\end{array}
```

```
\begin{array}{|c|c|c|}
\hline
C & D & E \\
\hline
\alpha & 10 & a \\
\beta & 10 & a \\
\beta & 20 & b \\
\gamma & 10 & b \\
\hline
\end{array}
```

```
\begin{array}{|c|c|c|c|c|}
\hline
A & B & C & D & E \\
\hline
\alpha & 1 & \alpha & 10 & a \\
\alpha & 1 & \beta & 10 & a \\
\alpha & 1 & \beta & 20 & b \\
\alpha & 1 & \gamma & 10 & b \\
\beta & 1 & \alpha & 10 & a \\
\beta & 1 & \beta & 10 & a \\
\beta & 1 & \beta & 20 & b \\
\beta & 1 & \gamma & 10 & b \\
\beta & 1 & \gamma & 10 & b \\
\hline
\end{array}
```

“take the cross product of \( r \) and \( s \), then return only the rows where \( A \) equals \( B \)”
Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.

Example:

\[ \rho (myRelation, (r - s)) \]

Renaming columns (rename A to A2):

\[ \rho (myRelation(A->A2), (r - s)) \]

Take the set difference of \( r \) and \( s \), and call the result \( myRelation \)

Renaming in relational algebra is essentially the same as assignment in a programming language.
Rename Operation

If a relational-algebra expression $Y$ has arity $n$, then

$$\rho(X(A\rightarrow A1, B\rightarrow A2, \ldots), Y)$$

returns the result of expression $Y$ under the name $X$, and with the attributes renamed to $A1, A2, \ldots, An$.

For example,

$$\rho(\text{myRelation}(A\rightarrow E, B\rightarrow K), (r - s))$$

Take the set difference of $r$ and $s$, and call the result myRelation, while renaming the first field to $E$, and the second field to $K$. 
Sailors Example

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

- “Sailors” and “Reserves” relations for our examples.

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>surname</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>surname</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>
Algebra Operations

• Look what we want to get from the following table:

<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>

S2
Selection

- Selects rows that satisfy \textit{selection condition}.
- No duplicates in result! (Why?)
- \textit{Schema} of result identical to schema of (only) input relation.

\[
\sigma_{\text{rating} > 8}(S2) = \\
\begin{array}{|c|c|c|c|}
\hline
\text{sid} & \text{sname} & \text{rating} & \text{age} \\
\hline
28 & yuppy & 9 & 35.0 \\
31 & lubber & 8 & 55.5 \\
44 & guppy & 5 & 35.0 \\
58 & rusty & 10 & 35.0 \\
\hline
\end{array}
\]
Projection

• Deletes attributes that are not in projection list.
• *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
• Projection operator has to eliminate *duplicates*! (Why??)
  – Note: real systems typically don’t do duplicate elimination unless the user explicitly asks for it.

\[
\pi_{\text{sname, rating}}(S2)
\]

<table>
<thead>
<tr>
<th>sname</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>yuppy lubber</td>
<td>9</td>
</tr>
<tr>
<td>guppy rusty</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

\[
\pi_{\text{age}}(S2)
\]

<table>
<thead>
<tr>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.0</td>
</tr>
<tr>
<td>55.5</td>
</tr>
</tbody>
</table>
Composition of Operations

- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition*)

<table>
<thead>
<tr>
<th>sid</th>
<th>surname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>yummy</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>

\[
\pi_{\text{name}, \text{rating}}(\sigma_{\text{rating} > 8}(S2)) = \\
\begin{array}{|c|c|}
\hline
\text{name} & \text{rating} \\
\hline
\text{yummy} & 9 \\
\text{rusty} & 10 \\
\hline
\end{array}
\]
What do we want to get from two relations?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>bid</td>
<td>day</td>
</tr>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

What about: **Who reserved boat 101?**

Or: **Find the name of the sailor who reserved boat 101.**
Cross-Product

- Each row of S1 is paired with each row of R1.
- Result schema has one field per field of S1 and R1, with field names inherited.

<table>
<thead>
<tr>
<th>sid1</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
<th>sid2</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

Renaming operator (because of naming conflict):
\[ \rho(sid \rightarrow sid1, S1) \times \rho(sid \rightarrow sid2, R1) \]
Why does this cross product help

Query: Find the name of the sailor who reserved boat 101.
Another example

- Find the name of the sailor having the highest rating.

\[ \text{AllR} = \pi_{\text{ratingA}} \rho(\text{rating} \rightarrow \text{ratingA}, S2) \]

\[ \text{Result?} = \pi_{\text{Sname}} (\sigma_{\text{rating} < \text{ratingA}} (S2 \times \text{AllR})) \]

What’s in “Result?”?

Does it answer our query?

<table>
<thead>
<tr>
<th>sid</th>
<th>surname</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>
S2

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>puppy</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>

AllR

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

AllR = \pi_{\text{ratingA}} \rho(\text{rating} \rightarrow \text{ratingA}, S2)

Result? = \pi_{\text{Sname}} (\sigma_{\text{rating} < \text{ratingA}} (S2 \times \text{AllR}))
### Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - Same number of fields.
  - ‘Corresponding’ fields have the same type.

- What is the *schema* of result?

<table>
<thead>
<tr>
<th>sid</th>
<th>surname</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>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>28</td>
<td>yuppy</td>
<td>9</td>
<td>35.0</td>
</tr>
</tbody>
</table>

\[ S_1 \cup S_2 \]

\[ S_1 \cap S_2 \]
Back to our query

• Find the name of the sailor having the highest rating.
Relational Algebra (Summary)

• Basic operations:
  – **Selection** (σ) Selects a subset of rows from relation.
  – **Projection** (π) Deletes unwanted columns from relation.
  – **Cross-product** (×) Allows us to combine two relations.
  – **Set-difference** (-) Tuples in reln. 1, but not in reln. 2.
  – **Union** (∪) Tuples in reln. 1 and in reln. 2.

Also,
  – **Rename** (ρ) Changes names of the attributes

• Since each operation returns a relation, operations can be composed! (Algebra is “closed”.)

• Use of temporary relations recommended.