# Database Querying Through Relational Algebra

ACS 575: Database Systems

Instructor: Dr. Jin Soung Yoo, Professor
Department of Computer Science
Purdue University Fort Wayne

# References

- Elmasri et al., Fundamentals of Database Systems, Ch4

# 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 support easy, efficient access to large data sets.
  - QLs not expected to be "Turing complete".
  - QLs not intended to be used for complex calculations.

# Relational Query Languages

- Two mathematical Query Languages form the basis for "real" languages (e.g. SQL), and for implementation:
  - **Relational Algebra**
    - A procedure query language that can be used to tell the DBMS how to build a new relation from one or more relations in databases
  - **Relational Calculus**
    - A non-procedural language that can be used to formulate the definition of a relation in terms of one or more relations
    - Provides only the description for the query but does not provide the methods to solve it. E.g., $\{t \mid t \in loan \wedge t[amount] >= 10000\}$
- Relational Algebra and Relational Calculous mainly provides theoretical foundation for relational databases and SQL

# Relational Algebra

- A theoretical language with operations that work on one or more relations to define another relation without changing the original relation(s).
- Relational Algebra collects instances of relations as input and gives occurrences of relations as output.
  - Each operation takes one or more relations as its operand(s) and generates another relation as its result, e.g., $R1 \cup R2 = R3$
- Relational algebra query operations are performed recursively on a relation.
  - The output from one operation can become the input to another operation.

# Basic relational Algebra Operations

- Unary Relational Operations
  - *Selection* (symbol: $\sigma$)
  - *Projection* (symbol: $\pi$)
  - *Rename* (symbol: $\rho$)
- Relational Algebra Operations from Set Theory
  - *Union* ($\cup$)
  - *Intersection*($\cap$)
  - *Difference* ($-$)
  - *Cartesian Product* ($\times$)
- Binary Relational Operations
  - *Join* ($\bowtie$)
  - *Division*

# Example Relations

- Relation schema

  Sailors (<u>sid</u>, sname, rating, age)

  Boats (<u>bid</u>, bname, color)

  Reserves (<u>sid</u>, <u>bid</u>, <u>day</u>)

- Example instances

Sailors (*S*)

| <u>sid</u> | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

Reserves (*R*)

| <u>sid</u> | <u>bid</u> | <u>day</u> |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

Boats (*B*)

| <u>bid</u> | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |

# Project ($\pi$)

- The PROJECT operation eliminates all attributes of the input relation but those mentioned in the projection list.
- The project method defines a relation that contains a vertical subset of relation
  - Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- This operators is useful for deleting unwanted columns from relation
- E.g.,

**Sailors (*S*)**

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$\boldsymbol{\pi_{sname,rating}(S)}$

| sname | rating |
|---|---|
| yuppy | 9 |
| lubber | 8 |
| guppy | 5 |
| rusty | 10 |

$\boldsymbol{\pi_{age}(S)}$

| age |
|---|
| 35.0 |
| 55.5 |
| 35.0 |
| 35.0 |

# Select ($\sigma$)

- The SELECT operation is used for selecting a subset of the tuples according to a given selection condition

  $\boldsymbol{\sigma_p(r)}$ , where $\boldsymbol{r}$ stands for relation which is the (alias) name of the table, and $\boldsymbol{p}$ is a prepositional logic for row selection.

  - e.g., $\boldsymbol{\sigma_{rating=8 \ and \ age>50}(S)}$
- The schema of result is identical to schema of input relation
- The result relation can be the *input* for another relational algebra operation

**Sailors ($S$)**

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$\boldsymbol{\sigma_{rating>8}(S)}$

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$\boldsymbol{\pi_{sname,rating}(\sigma_{rating>8}(S))}$

| sname | rating |
|---|---|
| yuppy | 9 |
| rusty | 10 |

# Union, Intersection, Set-Difference

- UNION(∪)

  $$R_1 \cup R_2 = \{r \mid r \in R_1 \text{ } or \text{ } r \in R_2\}$$

  - It performs binary union between two given relations
- INTERESECT(∩)

  $$R_1 \cap R_2 = \{r \mid r \in R_1 \text{ } and \text{ } r \in R_2\}$$

  - It finds all the tuples that are present in both $R_1$ and $R_2$.
- Set Difference ($-$)

  $$R_1 - R_2$$

  - It finds all the tuples that are present in $R_1$ but not in $R_2$.

# Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - The same number of attributes
  - Attribute domains need to be compatible
  - Duplicate tuples should be automatically removed

*S1*

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

*S2*

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

$S1 \cup S2$

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |
| 44 | guppy | 5 | 35.0 |
| 28 | yuppy | 9 | 35.0 |

$S1 \cap S2$

| sid | sname | rating | age |
|---|---|---|---|
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

$S1 - S2$

| sid | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |

# Cartesian Product ($\times$)

- Cartesian Product is an operation used to merge columns from two relations information of two relations into one.
  $S \times R = \{r\ s \mid r \in R \text{ or } s \in S\}$
  - Each row of *S* is paired with each row of *R*.
  - *Result schema* has one attribute per attribute of *S* and *R*
  - *Conflict*: Both *S* and *R* have a field called *sid*.
- Generally, Cartesian product is never a meaningful operation when it performs alone.

*S*

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

*R*

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

*S* x *R*

| (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 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

# Cross Product

- Cartesian Product becomes meaningful when it is followed by other operations. E.g., $\sigma_{\,sid=22}\,(S \times R)$
- It is called Cross Product or Cross Join

| <u>sid</u> | sname | rating | age |
|---|---|---|---|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

$S$

| <u>sid</u> | <u>bid</u> | <u>day</u> |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

$R$

| (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 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

$S$ x $R$

$\sigma_{S.sid < R.sid}(S \times R)$

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |

# Join ($\bowtie$)

- Join operation is essentially a Cartesian product followed by a selection criterion. $R \bowtie_c S = \sigma_c(R \times S)$
  - The result schema is same as that of cross-product.
  - Fewer tuples than cross-product, might be able to compute more efficiently

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

$S$

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

$R$

| (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 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

$S \times R$

$\sigma_{S.sid < R.sid}(S \times R)$

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |

$S \bowtie_{S.sid < R.sid} R$

# Type of Join (Inner Join)

- ***Theta join*:** the general case of join is called Theta join
- ***Equi join*:** A special case of condition join where the condition *c* contains only ***equalities*.**

$S \bowtie_{sid=sid} R$ $\quad$ $S \bowtie_{sid} R$ $\quad$ $S \bowtie R$

$S$

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

$R$

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

$S \bowtie_{sid=sid} R$

| (sid) | sname | rating | age | (sid) | bid | day |
|---|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |

- ***Natural join*:** Equijoin on *all* common fields.

# Example 1 of Algebra Query

- Find names of sailors who've reserved boat #103

*Sailors*

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

*Reserves*

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

- Answer

$$\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie_{sid=sid} Sailors)$$

Internal procedure:

$\rho\ (Temp1,\ \sigma_{bid=103} Reserves)$

$\rho\ (Temp2,\ Temp1 \bowtie Sailors)$

$\pi_{sname}(Temp2)$

* ρ: renaming operator

- several alternative query expressions

$$\pi_{sname}((\sigma_{bid=103}(Reserves \bowtie_{sid=sid} Sailors))$$

# Example 2

- Find names of sailors who've reserved a red boat

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

*Sailors*

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 22 | 102 | 10/10/98 |
| 58 | 103 | 11/12/96 |

*Reserves*

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |

*Boats*

- Answer

$$\pi_{sname}(((\sigma_{color=`red`}\ Boats) \bowtie_{bid} Reserve) \bowtie_{sid} Sailors)$$

- several alternative query expressions

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color=`red`}\ Boats) \bowtie_{bid} Reserves) \bowtie_{sid} Sailors)$$

# Example 3

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

*Sailors*

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 22 | 102 | 10/10/98 |
| 58 | 103 | 11/12/96 |

*Reserves*

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |

*Boats*

- Find sailors who've reserved a red <u>or</u> a green boat
  - Can identify all red or green boats, then find sailors who've reserved one of these boats:

❖ Answer

$$\pi_{sname}(((\sigma_{color=`red` \ or \ color=`green`} \ Boats) \bowtie_{bid} Reserves) \bowtie_{sid} Sailors)$$

Internal procedure

$$\rho(Tempboats, (\sigma_{color='red' \vee color='green'} Boats))$$

$$\rho(Tempboats2, (Tempboats \bowtie Reserves))$$

$$\pi_{sname}(Tempboats2 \bowtie Sailors)$$

| sname |
|---|
| dustin |
| rusty |

# Example 4

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

*Sailors*

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/96 |
| 22 | 102 | 10/10/98 |
| 58 | 103 | 11/12/96 |

*Reserves*

| bid | bname | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |

*Boats*

- Find sailors who've reserved a red **and** a blue boat
  - Previous approach won't work! Must identify sailors who've reserved red boats, sailors who've reserved blue boats, then find the intersection

$$\cancel{\pi_{sname}(((\sigma_{color=`red` \text{ and } color=`blue`} Boats) \bowtie_{bid} Reserves) \bowtie_{sid} Sailors)}$$

- Answer

$$\boldsymbol{\pi_{sname}(((\pi_{sid}((\sigma_{color=`red`} Boats) \bowtie_{bid} Reserves)) \cap (\pi_{sid}((\sigma_{color=`blue`} Boats) \bowtie_{bid} Reserves))) \bowtie_{sid} Sailors)}$$

that is,

$$\rho\ (Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie Reserves))$$

$$\rho\ (Tempblue, \pi_{sid}((\sigma_{color='blue'} Boats) \bowtie Reserves))$$

$$\pi_{sname}((Tempred \cap Tempblue) \bowtie Sailors)$$

| sname |
|---|
| dustin |