# Relational Algebra

CS430/630
Lecture 2

Slides based on "Database Management Systems" 3rd ed, Ramakrishnan and Gehrke

## 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 intended to be used for complex calculations
  - QLs support easy, efficient access to large data sets



## Formal Relational Query Languages

- Two languages form the basis for SQL:
  - *Relational Algebra*:
    - operational
    - useful for representing execution plans
    - very relevant as it is used by query optimizers!
  - *Relational Calculus*:
    - Lets users describe the result, NOT how to compute it - declarative
    - We will focus on relational 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
  - The schema for the *result* of a given query is determined by input schemas and operator type
- Each operation returns a relation
  - operations can be *composed* !
  - Well formed expression: a relation, or the results of a relational algebra operation on one or two relations

## Relational Algebra

- Basic operations:
  - Selection $\sigma$ Selects a subset of rows from relation
  - Projection $\pi$ Deletes unwanted columns from relation
  - Cross-product X Allows us to combine several relations
  - Join $\bowtie$ Combines several relations using conditions
  - Division $\div$ A bit more complex, will cover later on
  - Set-difference — Union $\cup$ Intersection $\cap$
  - Renaming $\rho$ Helper operator, does not derive new result, just renames relations and fields

$$\rho(R(F), E)$$

  - F contains *oldname* → *newname* pairs

## Example Schema

***Sailors***

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

***Boats***

| bid | name | color |
|---|---|---|
| 101 | interlake | red |
| 103 | clipper | green |

***Reserves***

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

## Relation Instances Used

*Sailors*

*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 |

*Reserves*

*R1*

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

## Projection

- Unary operator
- Deletes (projects out) attributes that are not in *projection list*

$$\pi_{attr1,attr2,\ldots} relation$$

- *Result Schema* contains the attributes in the projection list
  - With the same names that they had in the input relation
- Projection operator has to eliminate *duplicates*!
  - Real systems typically do not do so by default
  - Duplicate elimination is expensive! (sorting)
  - User must explicitly asks for duplicate eliminations (DISTINCT)

## Projection Example

*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 |

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

$\pi_{sname,rating}(S2)$

## Selection

- Unary Operator
- Selects rows that satisfy *selection condition*

$$\sigma_{condition} relation$$

- Condition contains constants and attributes from relation
  - Evaluated for each **individual** tuple
  - May use logical connectors AND ($\wedge$), OR ($\vee$), NOT ($\neg$)
- No duplicates in result! Why?
- *Result Schema* is identical to schema of the input relation

## Selection Example

*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 |

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

$\sigma_{rating>8}(S2)$

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

Selection and Projection $\pi_{sname,rating}(\sigma_{rating>8}(S2))$

## Cross-Product

- Binary Operator

$$R \times S$$

- Each row of relation *R* is paired with each row of *S*
- *Result Schema* has one field per field of R and S
  - Field names `inherited' when possible

## Cross-Product Example

*S1*

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

*R1*

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

*C=S1 X R1*

| (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 |

*Conflict*: Both *R* and *S* have a field called *sid*

## Cross-Product + Renaming Example

*C*

| sid1 | sname | rating | age | sid2 | 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 |

*Renaming operator* $\rho(C(1\rightarrow sid1, 5\rightarrow sid2), S1\times R1)$

## Condition Join (Theta-join)

$$R \bowtie_\theta S = \sigma_\theta(R\times S)$$

- *Result Schema* same as that of cross-product



## Condition Join (Theta-join) Example

*S1 X R1*

| sid1 | sname | rating | age | sid2 | 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~~ |

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

| sid1 | sname | rating | age | sid2 | bid | day |
|---|---|---|---|---|---|---|
| ~~22~~ | ~~dustin~~ | ~~7~~ | ~~45.0~~ | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |

## Equi-Join

- A special case of condition join where the condition contains only ***equalities***

$$R \bowtie_{R.attr1=S.attr2} S$$

- *Result Schema* similar to cross-product, but only one copy of fields for which equality is specified.

## Equi-Join Example

*S1 X R1*

| sid1 | sname | rating | age | sid2 | 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 |

$$S1 \bowtie_{sid} R1$$

| sid | sname | rating | age | bid | day |
|---|---|---|---|---|---|
| 22 | dustin | 7 | 45.0 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 103 | 11/12/96 |

## Natural Join

- Equijoin on *all* common fields

$$R \bowtie S$$

- Common fields are NOT duplicated in the result

## 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 domain (type)
- What is the *schema* of result?

## Union Example

***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 |

| 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 |

$S1 \cup S2$

## Intersection Example

***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 |

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

$S1 \cap S2$

## Set-Difference Example

***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 |

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

$S1 - S2$