

# CSCI3170 Introduction to Database Systems

---

## TUTORIAL 2 – RELATIONAL ALGEBRA

Chen Xiong

# Outline

---

□ Overview

□ Examples

□ Practice

# Relational Algebra

---

- *Query languages*: Allow manipulation and retrieval of data from a database.
- *Relational algebra*: Instruct system operations to produce the desired results.
  - Input & Output: relation instance.

# General Operators

---

## BASIC OPERATORS

- Selection ( $\sigma$ )
- Projection ( $\Pi$ )
- Union ( $\cup$ )
- Set difference ( $-$ )
- Cartesian Product ( $\times$ )
- Rename ( $\rho$ )

## ADDITIONAL OPERATORS

- Intersection ( $\cap$ )
- Join ( $\bowtie$ )
- Division ( $/$ )

# Selection ( $\sigma$ )

---

$R2 := \sigma_C(R1)$

$C$ : selection condition

- Logical connectives: and ( $\wedge$ ), or ( $\vee$ )
- Comparison operators:  $<$ ,  $\leq$ ,  $=$ ,  $\neq$ ,  $>$ ,  $\geq$

Example:  $\sigma_{\text{year} > 2}(S1)$

sid	name	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21

**S1**



sid	name	year	age
1	Peter	3	22
3	Mary	4	21

# Projection ( $\Pi$ )


---

$R2 := \Pi_L(R1)$

$L$ : a list of attributes from the schema of  $R1$ .

Example:  $\Pi_{\text{name,age}}(S1)$

sid	name	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21



name	age
Peter	22
John	20
Mary	21

**S1**

## Union ( $\cup$ ), Intersection ( $\cap$ ), Set difference ( $-$ ),

---

- S1 and S2 must be union-compatible
- $S1 \cup S2$ : returns a relation instance containing all tuples that occur in either relation S1 or relation S2 (or both).
- $S1 \cap S2$ : returns a relation instance containing all tuples that occur in both S1 and S2.
- $S1 - S2$ : returns a relation instance containing all the tuples that occur in S1 but not in S2.

## Union ( $\cup$ ), Set difference ( $-$ ), Intersection ( $\cap$ )

sid	name	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21

**S1**

sid	name	year	age
2	John	2	20
3	Mary	4	21
4	David	3	22

**S2**

$$S1 \cap S2 = S1 - (S1 - S2)$$

sid	name	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21
4	David	3	22

$S1 \cup S2$

sid	name	year	age
1	Peter	3	22

$S1 - S2$

sid	name	year	age
2	John	2	20
3	Mary	4	21

$S1 \cap S2$



# Cartesian Product ( $\times$ )

$R3 := R1 \times R2$

- Pair each tuple **t1** of R1 with each tuple **t2** of R2.
- Concatenation **t1t2** is a tuple of R3.

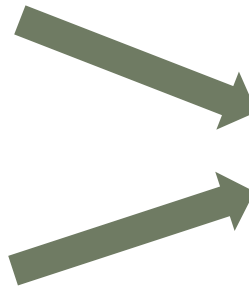
Example: R1 X R2

A	B
a1	b1
a2	b2

**R1**

B	C
b1	c1
b2	c2

**R2**



R1.A	R1.B	R2.B	R2.C
a1	b1	b1	c1
a1	b1	b2	c2
a2	b2	b1	c1
a2	b2	b2	c2

# Join ( $\bowtie$ )

*Condition Join:*  $R3 := R1 \bowtie_C R2$

- Take the product  $R1 \times R2$ , then apply  $\sigma_C$  to the result.
- $R1 \bowtie_C R2 = \sigma_C (R1 \times R2)$


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

**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



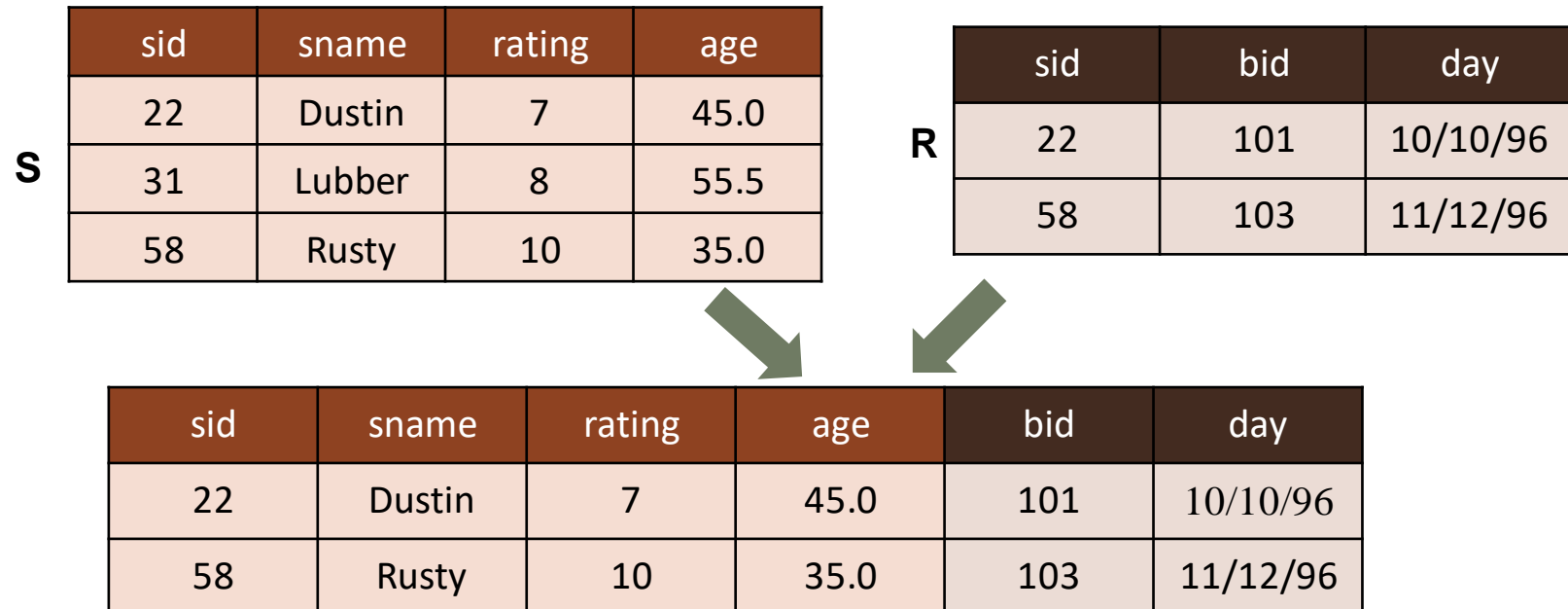
S.sid	sname	rating	age	R.sid	bid	day
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	58	103	11/12/96

# Join ( $\bowtie$ )

**Equi-Join:** A special case of condition join where the condition  $c$  contains only equalities.

**Natural Join:** Equi-join on *all common fields*.

Example:  $S \bowtie R$



# Division (/)

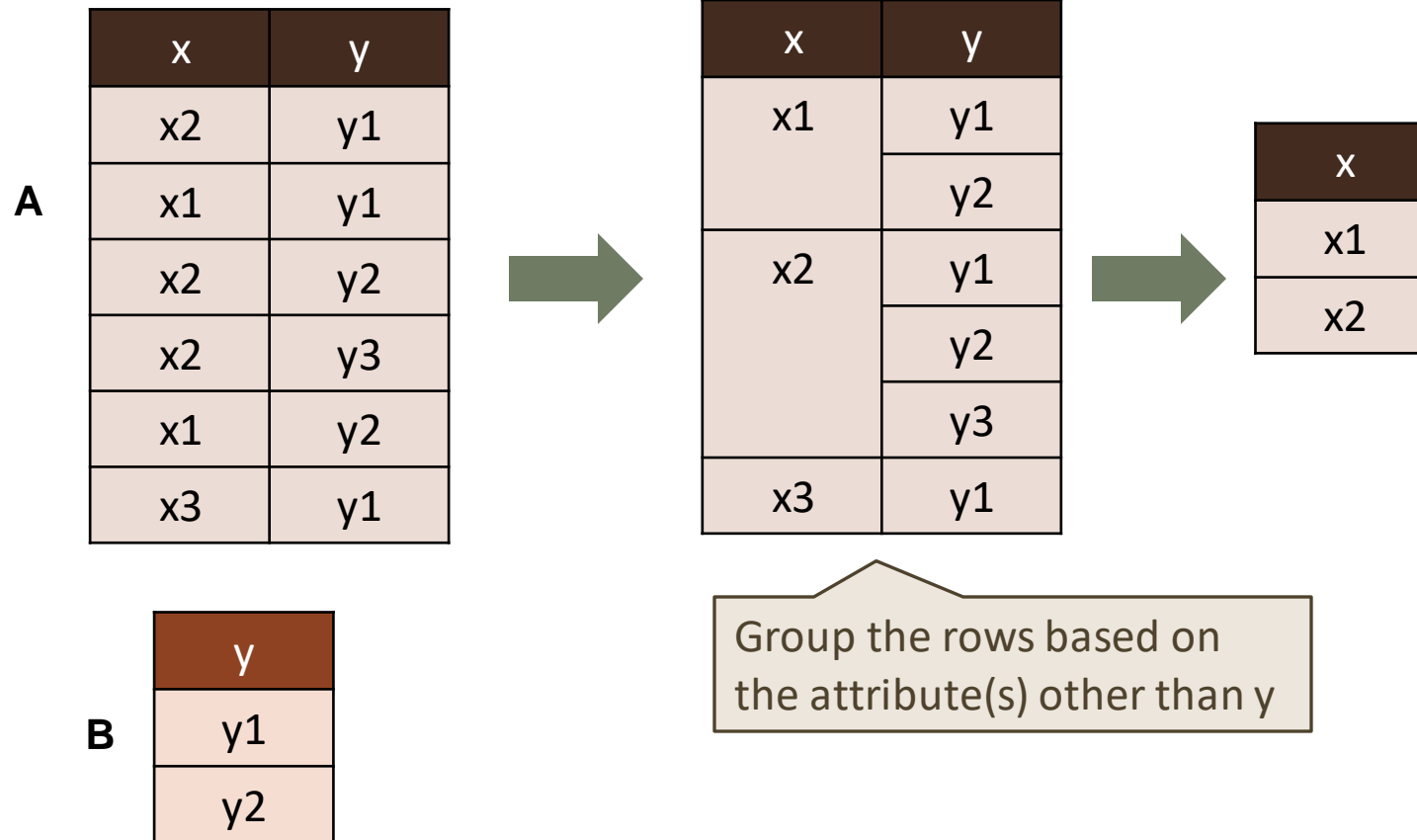
---

A / B

- Consider that A has two collection of fields **x** and **y**. B has one collection of fields **y**, with the same domain as in A. (collection can be of size  $\geq 1$ )
- The division operation A/B is the set of all x values such that for **every y value in a tuple of B**, where there is a tuple  $\langle x, y \rangle$  in A.

# Division (/)

Example: A / B



# Rename ( $\rho$ )

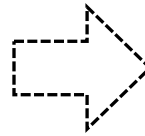
---

$\rho(T1, S1)$  or  $\rho(T1(F), S1)$

- F is called the **renaming list**:
- oldname or position  $\rightarrow$  newname
- e.g.  $\rho(T1(\text{name} \rightarrow \text{firstname}), S1)$

sid	name	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21

**S1**



sid	firstname	year	age
1	Peter	3	22
2	John	2	20
3	Mary	4	21

**T1**

# Rename ( $\rho$ )

---

Store the temporary result for later use

- e.g.  $\rho(T1, \sigma_{\text{year}>2}(S1))$

Compare the tuples in the same relation

- e.g.  $\rho(T1, S1)$   
 $\rho(T2, S1)$   
 $\sigma_{T1.\text{age} > T2.\text{age}} (T1 \times T2)$

# Precedence

---

Precedence of relational operators:

1. [SELECT, PROJECT, RENAME] (highest).
2. [PRODUCT, JOIN].
3. INTERSECTION.
4. [UNION, Set-Difference].

Note you can always insert **parentheses** to force the order you desire.



# Practice

---

Consider the following relations containing airline flight information:

Flights(flno, from, to, distance, departs, arrives)

Aircraft(aid, aname, cruisingrange)

Certified(eid, aid)

Employees(eid, ename, salary)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft (otherwise, he or she would not qualify as a pilot), and only pilots are certified to fly.

# Query 1

---

Flights(flno, from, to, distance, departs, arrives)  
Aircraft(aid, aname, cruisingrange)  
Certified(eid, aid)  
Employees(eid, ename, salary)

Find the *eids* of pilots certified for some Boeing aircraft.

*Sol1:  $\pi_{eid}(\sigma_{aname='Boeing'}(Aircraft \bowtie Certified))$*

*Sol2:  $\pi_{eid}((\sigma_{aname='Boeing'}Aircraft) \bowtie Certified)$*

# Query 2

---

Flights(flno, from, to, distance, departs, arrives)  
Aircraft(aid, aname, cruisingrange)  
Certified(eid, aid)  
Employees(eid, ename, salary)

Find the *aids* of all aircrafts that can be used on non-stop flights from New York to Los Angeles.

$\rho(NYToLA, \sigma_{from='New York' \wedge to='Los Angeles'}(Flights))$

$\pi_{aid}(\sigma_{cruisingrange > distance}(Aircraft \times NYToLA))$

# Query 3

---

Flights(flno, from, to, distance, departs, arrives)

Aircraft(aid, aname, cruisingrange)

Certified(eid, aid)

Employees(eid, ename, salary)

Find the *eids* of employees who are certified for the largest number of aircrafts.

This query can NOT be expressed in relational algebra because there is no operator to count.

# Remarks

---

- There is **no aggregate function** in relational algebra
- Queries written in relational algebra **can** be expressed by SQL
- Queries written in SQL **may not** be expressed by relational algebra
- A useful practice is to compare Relational Algebra with SQL.
  - Given a query by relational algebra, rewrite it by SQL; vice versa.