Relational Algebra and Relational Calculus

Part II: Binary Operations

Dr. Rim Moussa

Outline

- Unary operations
  - SELECT
  - PROJECT
- **Binary Operations**
  - **CROSS PRODUCT (aka CARTESIAN PRODUCT, CROSS JOIN)**
  - **JOIN**
    - **INNER JOIN (aka EQUIJOIN)**
    - LEFT | RIGHT OUTER JOIN
    - **THETA JOIN**
    - **SEMI-JOIN and ANTI-JOIN**
  - **SET Operations**
    - **UNION**
    - **INTERSECTION**
    - **DIFFERENCE (aka MINUS, EXCEPT)**
  - DIVISION
- Existential and Universal Quantifiers

CROSS PRODUCT Operation

- The Cartesian-product operation, denoted by a cross (×), allows us to combine information from any two relations. We write the Cartesian product of relations *r1* and *r2* as *r1* × *r2*.
- Query: ***All combinations of Instructor and teaches***

RA Expression: **instructor × teaches**

relation relation

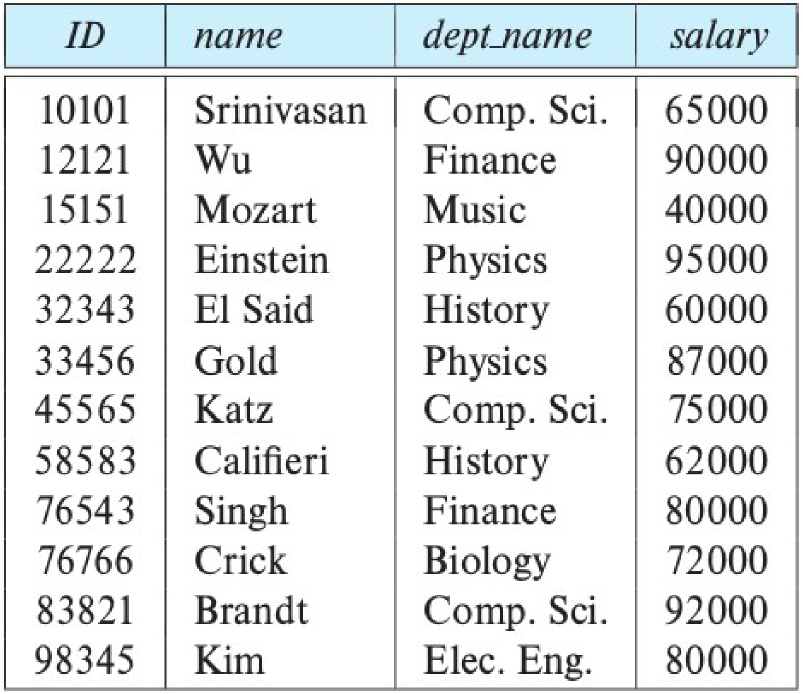
SQL statement:

SELECT *

FROM instructor, teaches;

CROSS PRODUCT Operation

Instructor

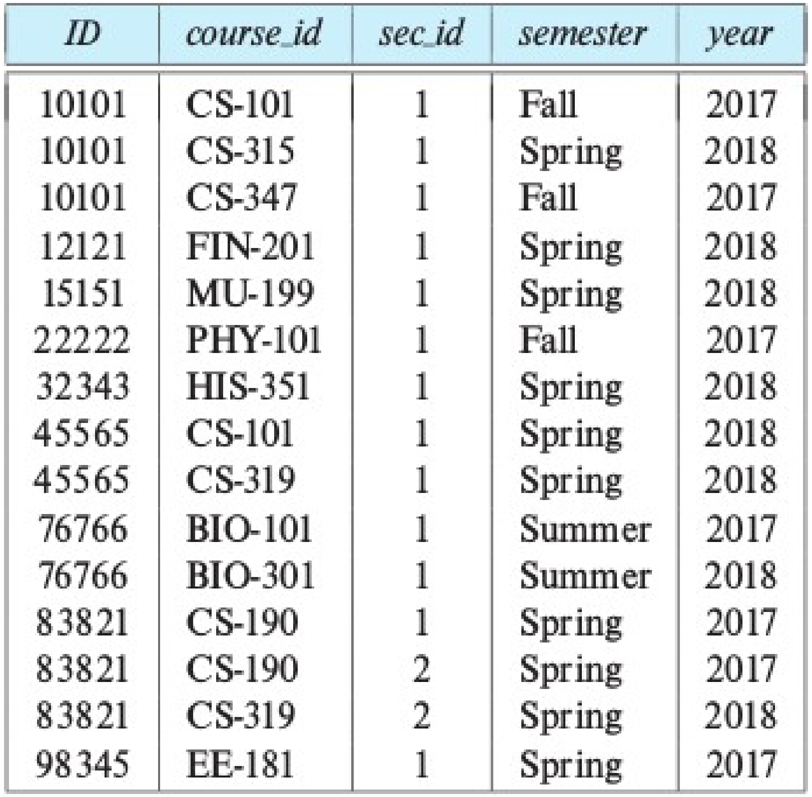


ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Cardinality(*Instructor*) = |*Instructor*| = 12

×

teaches



ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Cardinality(*teaches*) = |*teaches*| =15

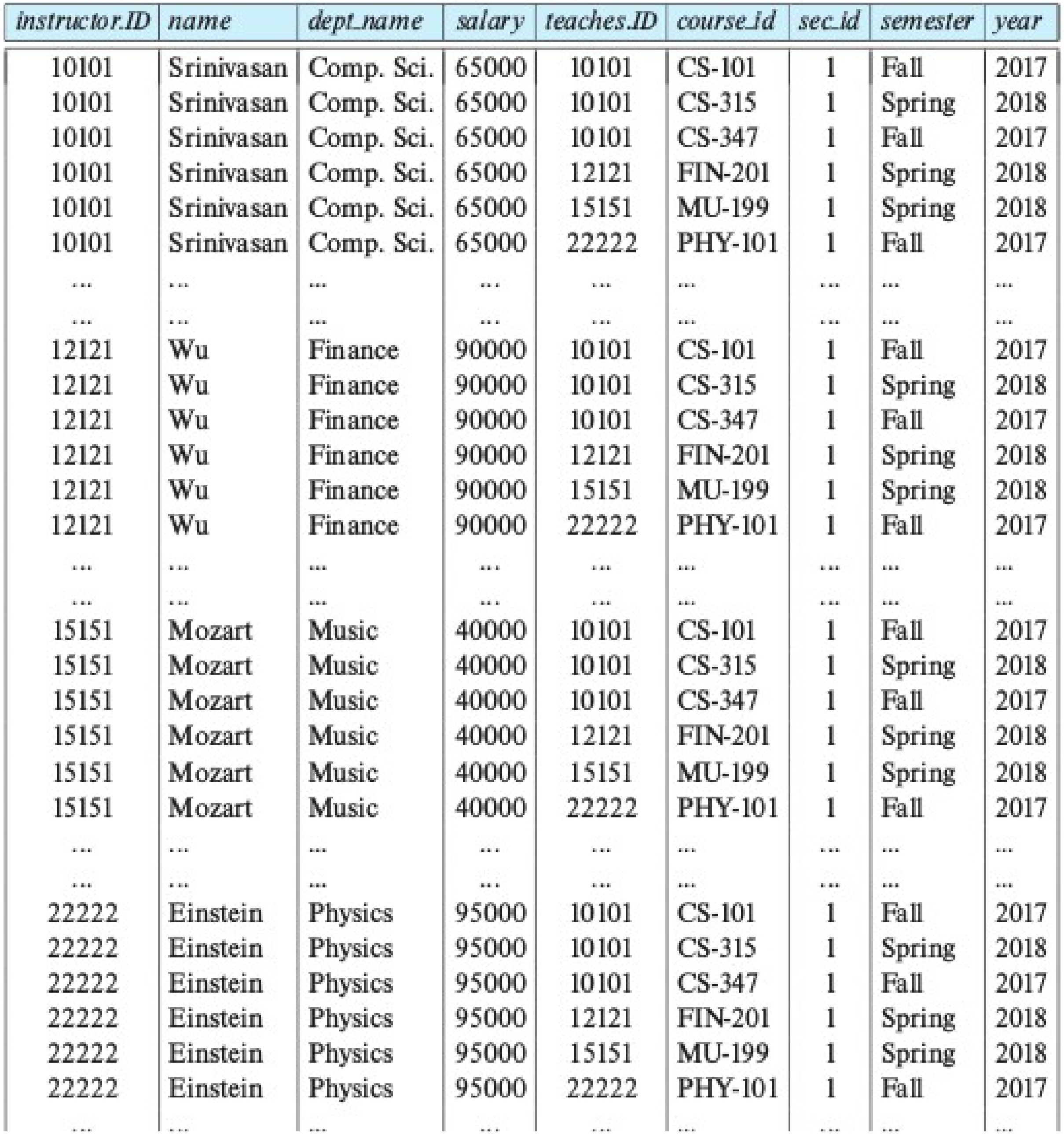
➡ Cross product (all combinations): 12 x 15 = 180 records

CROSS PRODUCT Operation

instructor × teaches

Nota:

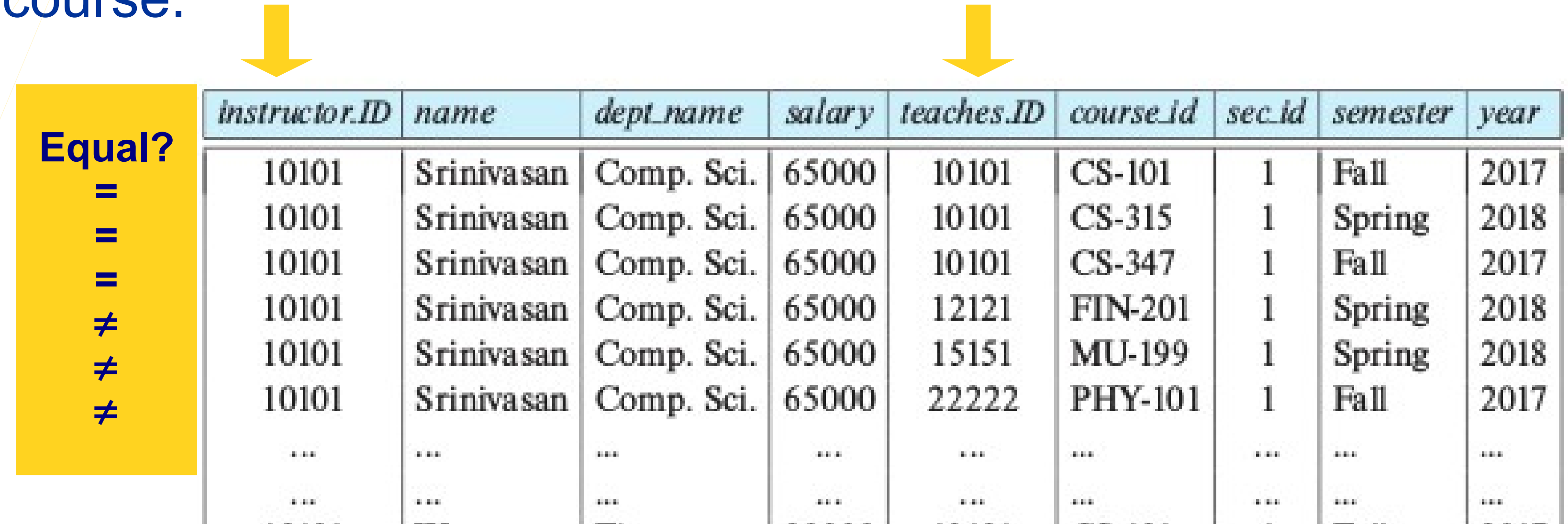
- *Table.Attribute* notation
- *Instructor.ID*: ID attribute in *Instructor* relation
- *Teaches.ID*: ID attribute in *teaches* relation



instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2018
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2017
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2018
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2017
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2017
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2018
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2017
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2018
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...

JOIN Operation

- The Cartesian product associates every instructor with every course that was taught, regardless of whether that instructor taught that course.



Equal?	*instructor.ID*	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
=	10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
=	10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
=	10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
≠	10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
≠	10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
≠	10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...

  - Instructor 10101 teaches courses: CS-101, CS-315, CS-347
  - Instructor 10101 doesn't teache courses: FIN-201, MU-199,PHY-101
- The equi-join operation is SELECT (σ) applied to Cartesian product

  RA Expression:

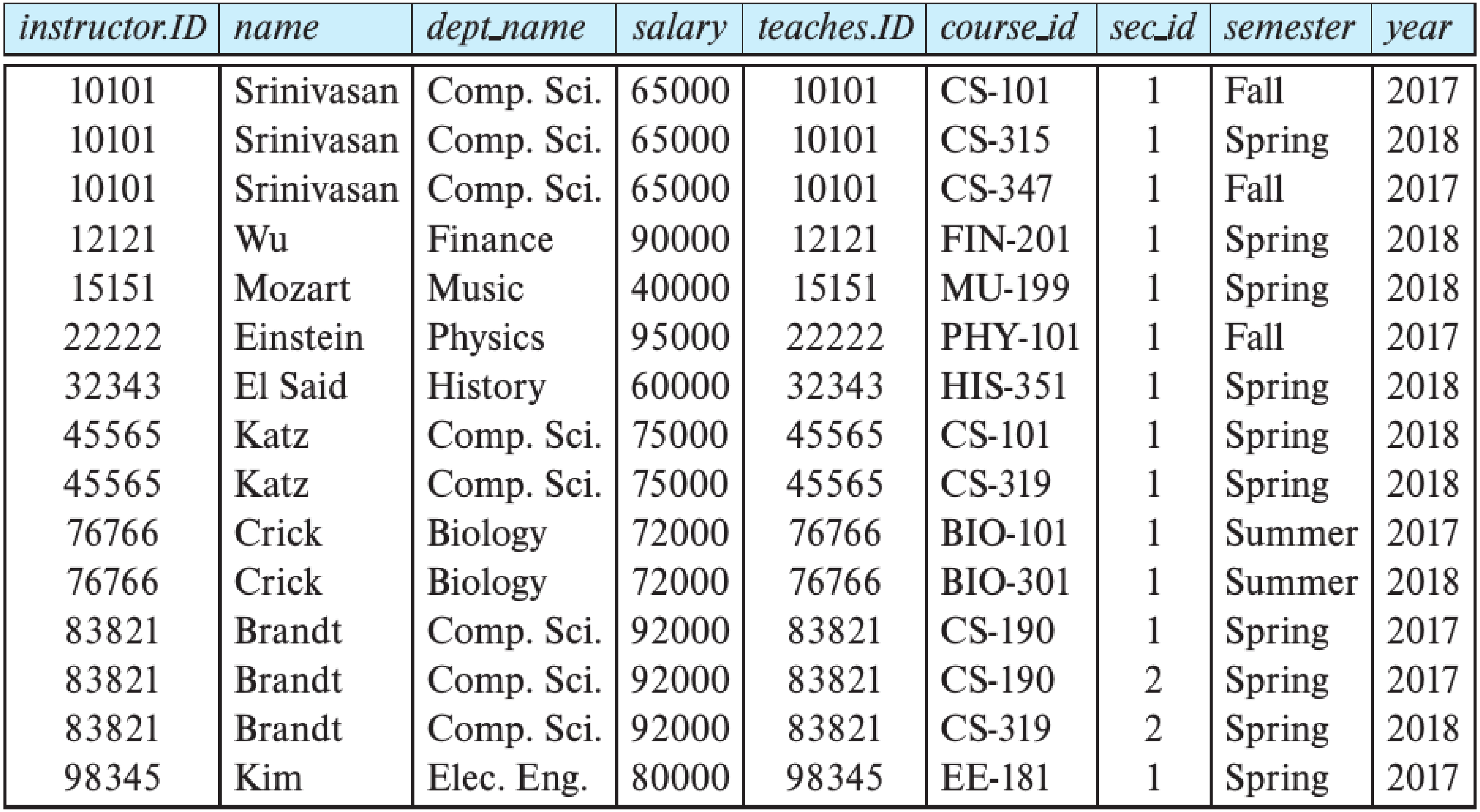
  $\sigma_{\text{instructor.ID = teaches.ID}}\ (\text{instructor} \times \text{teaches})$ or

  $\text{instructor} \bowtie_{\text{instructor.ID = teaches.ID}} \text{teaches}$

EQUIJOIN Operation

SQL statement:

```
SELECT *
FROM  instructor, teaches
WHERE instructor.ID = teaches.ID;
```



instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

THETA-JOIN Operation

- A JOIN operation with a general join condition is called a THETA JOIN .

  $$R(A1, A2, \dots An)$$

  $$S(B1, B2, \dots, Bm)$$

  $$J = R \bowtie_{\text{condition}} S$$

  where each <condition> is of the form **Ai θ Bj** ,

  Ai is an attribute of R,

  Bj is an attribute of S,

  Ai and Bj have the same domain,

  and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.

SEMI-JOIN

- The result contains the attributes of one table
- Query: *Instructors who teach*

RA expression:

$$\sigma_{instructor.ID \in \Pi \{teaches.ID\} (teaches)} (instructor)$$

instructor ⋉ teaches (-->contains only attributes of Instructor)

SQL statement:

```
SELECT *
FROM  instructor
WHERE instructor.ID IN (SELECT teaches.ID  FROM teaches);
---
SELECT *
FROM  instructor i
WHERE EXISTS (SELECT *
                                  FROM teaches  WHERE i.ID = t.ID);
```

ANTI-JOIN

- Query: *Instructors who do not teach*

RA expression:

$$\sigma_{\text{instructor.ID} \notin \Pi \{\text{teaches.ID}\} (\text{teaches})} (\text{instructor})$$

instructor ▷ teaches (-->contains only attributes of Instructor)

SQL statement:

```
SELECT *
FROM  instructor
WHERE instructor.ID NOT IN (SELECT teaches.ID  FROM teaches);
---
SELECT *
FROM  instructor i
WHERE NOT EXISTS (SELECT *
                         FROM teaches  WHERE i.ID = t.ID);
```

AUTO-JOIN

- Query: *Pairs of Instructors who are in the same department*

RA expression:

i1 ← instructor

i2 ← instructor

$$\text{result} \leftarrow \sigma_{\text{i1.ID} < \text{i2.ID}} (\text{ i1} \bowtie_{\text{i1.dept\_name} = \text{i2.dept\_name}} \text{i2})$$

SQL statement:

```
SELECT i1.ID, i1.name,i2.ID, i2.name,  i1.dept_name
FROM  instructor i1, instructor i2
WHERE  i1.dept_name = i2.dept_name
AND i1.ID < i2.ID;
```

SET UNION Operation

- The result of UNION operation, denoted by **R ∪ S**, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

RA expression **$R = \Pi_{\{Fnane,Lname\}}$ (Instructor) ∪ $\Pi_{\{Fn, Ln\}}$ (Student)**

SQL Statement

```
SELECT Fname, Lname
FROM Instructor
UNION
SELECT Fn, Ln
FROM Student;
```

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

∪

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

=

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

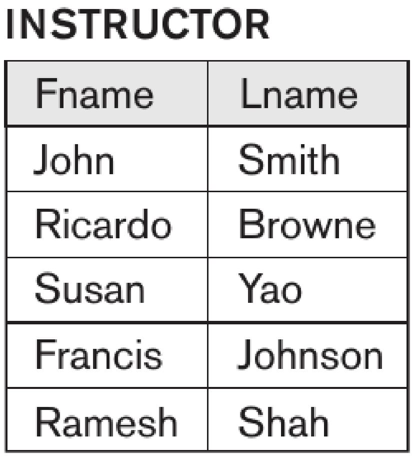
SET INTERSECTION Operation

- The result of INTERSECTION operation, denoted by **R ∩ S**, is a relation that includes all tuples that are in both R and S.

RA expression **$R = \Pi_{\{Fnane,Lname\}}$ (Instructor) ∩ $\Pi_{\{Fn, Ln\}}$ (Student)**

SQL Statement

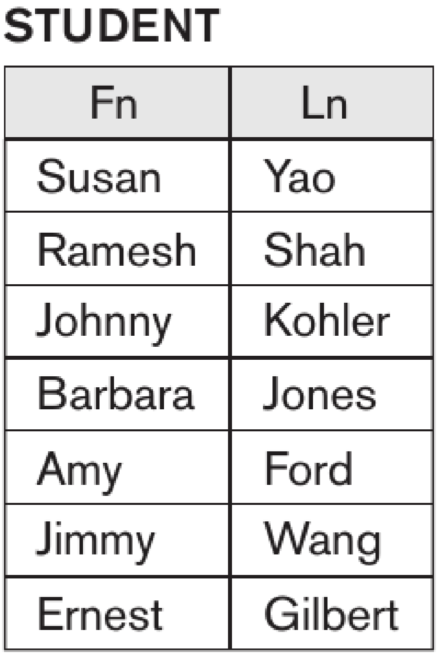
```
SELECT Fname, Lname
FROM Instructor
INTERSECT
SELECT Fn, Ln
FROM Student;
```



INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

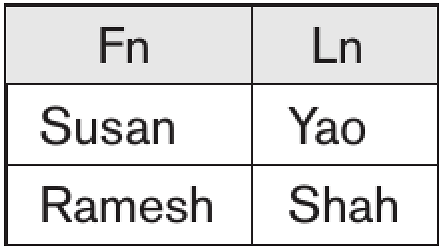
∩



STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

=



Fn	Ln
Susan	Yao
Ramesh	Shah

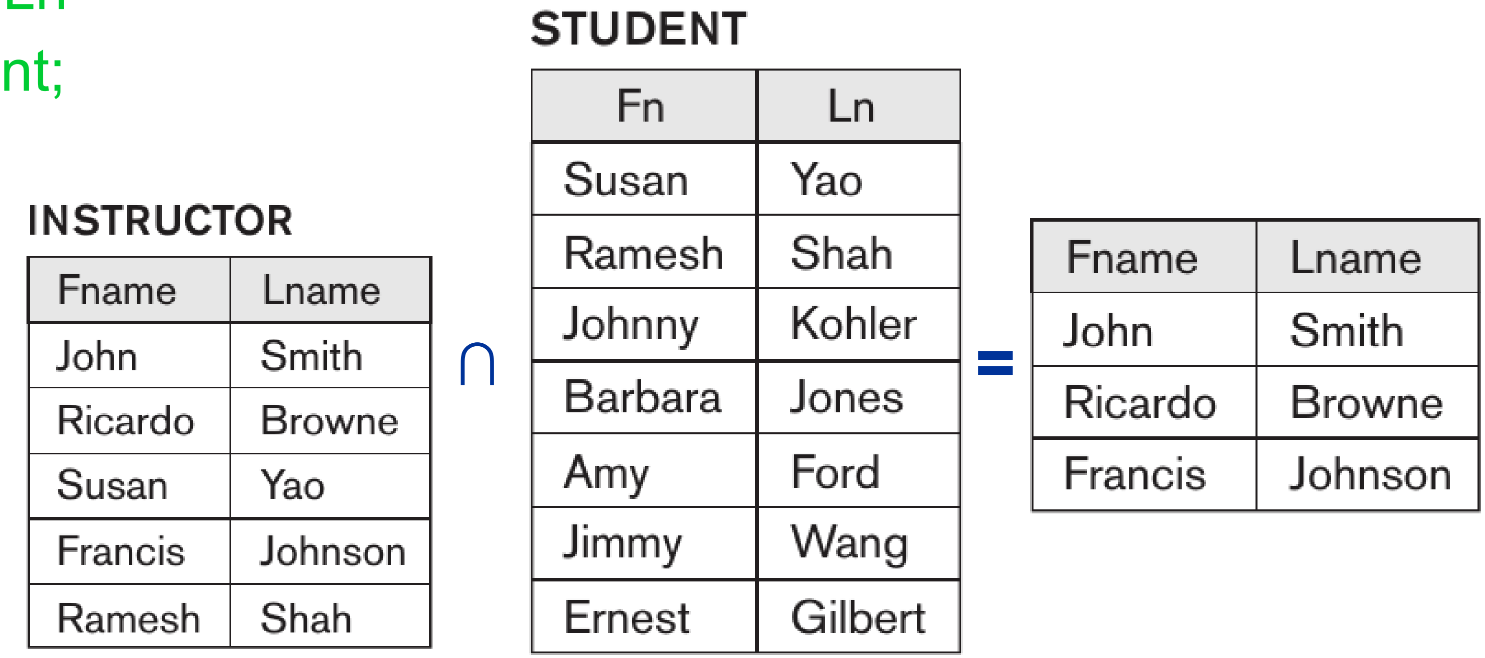
SET DIFFERENCE Operation

- The result of SET DIFFERENCE (aka MINUS, EXCEPT), denoted by **R – S**, is a relation that includes all tuples that are in R but not in S.

RA expression **$R = \Pi_{\{Fnane,Lname\}}$ (Instructor) - $\Pi_{\{Fn, Ln\}}$ (Student)**

SQL Statement

```
SELECT Fname, Lname
FROM Instructor
EXCEPT
SELECT Fn, Ln
FROM Student;
```



INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

∩

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

=

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

SET UNION, INTERSECT, DIFFERENCE Operations

- **Type Compatibility**
  - Two relations *R(A1, A2, ... , An)* and *S(B1, B2 , ... , Bn)* are said to be type compatible if they have the same degree *n* and if *domain(Ai)* = *domain(Bi)* for 1 ≤ i ≤ *n*

    The two relations have the same number of attributes and each corresponding pair of attributes has the same domain (Ai-Bi).
- **Commutativity**
  - Both UNION and INTERSECTION are commutative operations; that is,

    $R \cup S = S \cup R$

    $R \cap S = S \cap R$
  - The MINUS operation is not commutative; that is, in general,

    $R - S \neq S - R$
- **Associativity**
  - UNION and INTERSECTION can be treated as n-ary operations applicable to any number of relations because both are also associative operations; that is,

    $R \cup (S \cup T) = (R \cup S) \cup T$

    $(R \cap S) \cap T = R \cap (S \cap T)$