



## Chapter 2: Intro to Relational Model

These are original slides from the Authors used/modified with permission.

**Database System Concepts, 7<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Outline

- Structure of Relational Databases
- Database Schema
- Keys
- Schema Diagrams
- Relational Query Languages
- The Relational Algebra



# Example of a *Instructor* Relation

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	attributes (or columns)
10101	Srinivasan	Comp. Sci.	65000	tuples (or rows)
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	



# Relation Schema and Instance

- $A_1, A_2, \dots, A_n$  are *attributes*
- $R = (A_1, A_2, \dots, A_n)$  is a *relation schema*

Example:

*instructor* = (*ID*, *name*, *dept\_name*, *salary*)

- A relation instance  $r$  defined over schema  $R$  is denoted by  $r(R)$ .
- The current values a relation are specified by a table
- An element  $t$  of relation  $r$  is called a *tuple* and is represented by a *row* in a table



# Attributes

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible (*single value and not multivalued structured*)
- The special value ***null*** is a member of every domain. Indicated that the value is “unknown”
- The null value causes complications in the definition of many operations



# Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

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



# Database Schema

- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
  - schema: *instructor (ID, name, dept\_name, salary)*
  - Instance:

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



# Keys

- Let  $K \subseteq R$
- $K$  is a **superkey** of  $R$  if values for  $K$  are sufficient to identify a unique tuple of each possible relation  $r(R)$ 
  - Example:  $\{ID\}$  and  $\{ID, name\}$  are both superkeys of *instructor*.
- Superkey  $K$  is a **candidate key** if  $K$  is minimal (could be used as a PK)
  - Example: *Instructor*  $\{ID, name, dept\_name, salary\}$ .
    - Suppose  $\{ID\}$  and  $\{name, dept\_name\}$  are super keys, then they are candidate keys.
- One of the candidate keys is selected to be the **primary key**.
  - Which one?  $\{ID\}$  or  $\{name, dept\_name\}$





# Keys (cont'd..)

- Primary key (PK) ensures entity integrity (i.e., each row in the table has its own identity)
- *Each PK has two requirements:*
  - All the values in the PK must be UNIQUE
  - No key attribute in PK can contain NULL.
- *Foreign key (FK) ensures referential integrity.*
  - Value in one relation must appear in another.
  - *FK values must match PK values in a related table*
- Referential integrity – (NEXT SLIDE)
  - FK contains a value that refers to an existing valid tuple (row) in another relation •
    - Every entry in *VEND\_CODE* in the PRODUCT table has either a null or a valid value in *VEND\_CODE* in the VENDOR table



# Foreign Keys (cont'd..)

- Referential integrity –
  - FK contains a value that refers to an existing valid tuple (row) in another relation.
    - Every entry in **VEND\_CODE** in the PRODUCT table has either a null or a valid value in **VEND\_CODE** in the VENDOR table

Table name: PRODUCT  
Primary key: PROD\_CODE  
Foreign key: VEND\_CODE

Database name: Ch03\_SaleCo

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

link

Table name: VENDOR  
Primary key: VEND\_CODE  
Foreign key: none

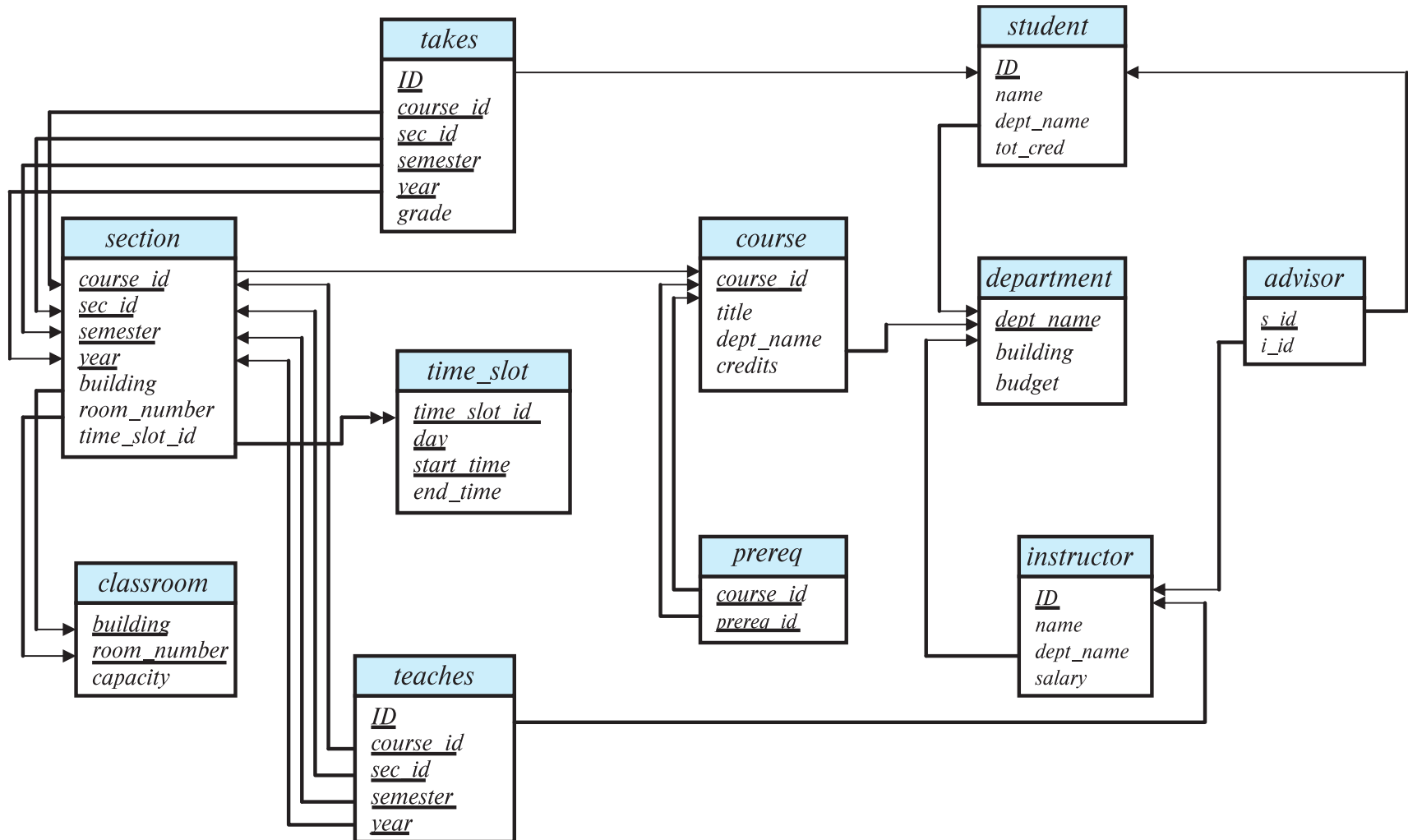
VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

SOURCE: Course Technology/Cengage Learning

SOURCE: Course Technology/Cengage Learning (Morris, 2020)



# Schema Diagram for University Database





# Relational Query Languages

- Procedural versus non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not Turing-machine equivalent
  - Consists of 6 basic operations



# Relational Algebra

- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operators
  - select:  $\sigma$
  - project:  $\Pi$
  - union:  $\cup$
  - set difference:  $-$
  - Cartesian product:  $\times$
  - rename:  $\rho$



# Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation:  $\sigma_p(r)$
- $p$  is called the **selection predicate**
- Example: select those tuples of the *instructor* relation where the instructor is in the “Physics” department.
  - Query

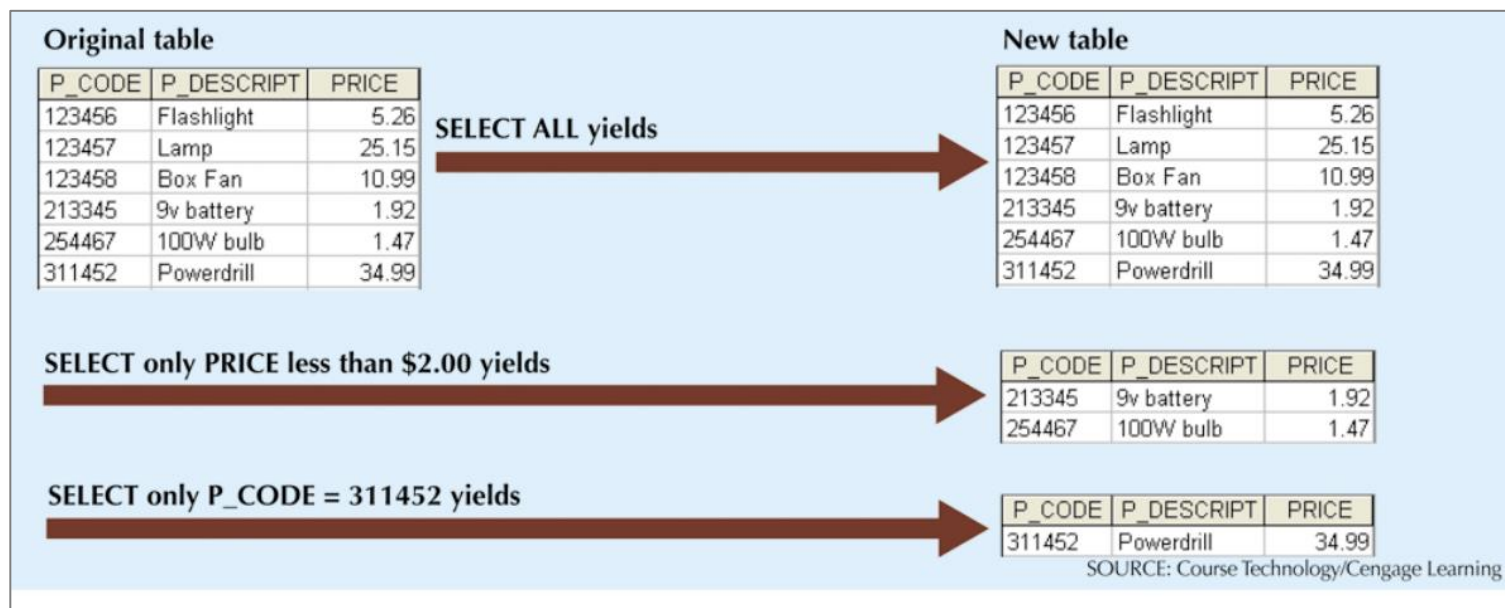
$$\sigma_{dept\_name="Physics"}(instructor)$$

- Result

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
33456	Gold	Physics	87000



# Select illustration



SOURCE: Course Technology/Cengage Learning (Morris, 2020)

- **SELECT** yields all values for all rows in a table that satisfy a given condition. Can also be used to list all rows in a table.
- Yields a horizontal subset of a table.



# Select Operation (Cont.)

- We allow comparisons using

$=, \neq, >, \geq, <, \leq$

in the selection predicate.

- We can combine several predicates into a larger predicate by using the connectives:

$\wedge$  (**and**),  $\vee$  (**or**),  $\neg$  (**not**)

- Example: Find the instructors in Physics with a salary greater \$90,000, we write:

$\sigma_{dept\_name="Physics" \wedge salary > 90,000} (instructor)$

- The select predicate may include comparisons between two attributes.

- Example, find all departments whose name is the same as their building name:
- $\sigma_{dept\_name=building} (department)$





# Project Operation

- A unary operation that returns its argument relation, with certain attributes left out.
- Notation:

$$\Pi_{A_1, A_2, A_3 \dots A_k}(r)$$

where  $A_1, A_2, \dots, A_k$  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



# Project Operation Example

- Example: eliminate the *dept\_name* attribute of *instructor*
- Query:

$$\Pi_{ID, name, salary} (instructor)$$

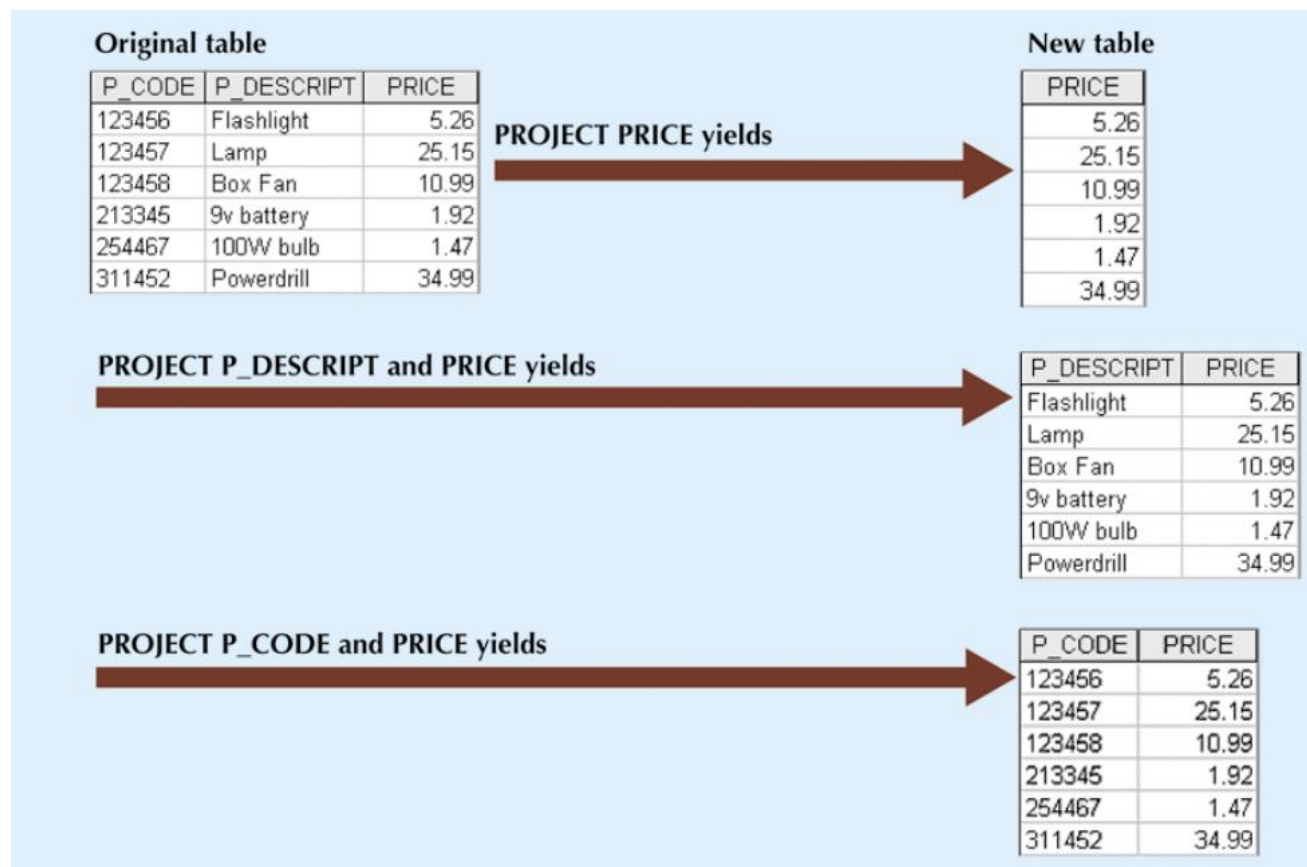
- Result:

<i>ID</i>	<i>name</i>	<i>salary</i>
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000



# Project illustration

- PROJECT Yields all values for selected attributes – a vertical subset of a table.



SOURCE: Course Technology/Cengage Learning (Morris, 2020)



# Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

$$\Pi_{name}(\sigma_{dept\_name = "Physics"}(instructor))$$

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.



# Cartesian-Product Operation

- The Cartesian-product operation (denoted by  $\times$ ) allows us to combine information from any two relations.
- Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

*instructor*  $\times$  *teaches*

- We construct a tuple of the result out of each possible pair of tuples: one from the *instructor* relation and one from the *teaches* relation (see next slide)
- Since the instructor *ID* appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
  - *instructor.ID*
  - *teaches.ID*



# The *instructor X teaches* table

<i>instructor.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
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

*instructor X teaches*

associates every tuple of *instructor* with every tuple of *teaches*.

- Most of the resulting rows have information about instructors who did NOT teach a particular course.

- To get only those tuples of “*instructor X teaches*” that pertain to instructors and the courses that they taught, we write:

$\sigma_{instructor.id = teaches.id} (instructor \times teaches)$

- We get only those tuples of “*instructor X teaches*” that pertain to instructors and the courses that they taught.
- The result of this expression, shown in the next slide



# Join Operation (Cont.)

- The table corresponding to:

$$\sigma_{instructor.id = teaches.id}(instructor \times teaches)$$

<i>instructor.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
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





# Join Operation (Cont.)

- The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- Consider relations  $r(R)$  and  $s(S)$
- Let “theta” be a predicate on attributes in the schema  $R \cup S$ . The join operation  $r \bowtie_{\theta} s$  is defined as follows:

$$r \bowtie_{\theta} s = \sigma_{\theta} (r \times s)$$

- Thus

$$\sigma_{instructor.id = teaches.id} (instructor \times teaches)$$

- Can equivalently be written as

$$instructor \bowtie_{Instructor.id = teaches.id} teaches.$$



# Union Operation

- The union operation allows us to combine two relations
- Notation:  $r \cup s$
- 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** (example: 2<sup>nd</sup> column of  $r$  deals with the same type of values as does the 2<sup>nd</sup> column of  $s$ )
- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$\Pi_{course\_id} (\sigma_{semester="Fall" \wedge year=2017} (section)) \cup$

$\Pi_{course\_id} (\sigma_{semester="Spring" \wedge year=2018} (section))$



# Union Operation (Cont.)

- Result of:

$$\Pi_{course\_id} (\sigma_{semester="Fall" \wedge year=2017}(section)) \cup \\ \Pi_{course\_id} (\sigma_{semester="Spring" \wedge year=2018}(section))$$

<i>course_id</i>
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101



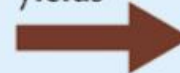
# Union illustration

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00
123458	Box Fan	10.99

yields



P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

SOURCE: Course Technology/Cengage

SOURCE: Course Technology/Cengage Learning (Morris, 2020)

- Combines all rows from two tables, excluding duplicate rows.
- • The tables must have union-compatible
  - i.e., same number of columns and their corresponding columns share the same domains.



# Set-Intersection Operation

- The set-intersection operation allows us to find tuples that are in both the input relations.
- Notation:  $r \cap s$
- Assume:
  - $r, s$  have the *same arity*
  - attributes of  $r$  and  $s$  are compatible
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

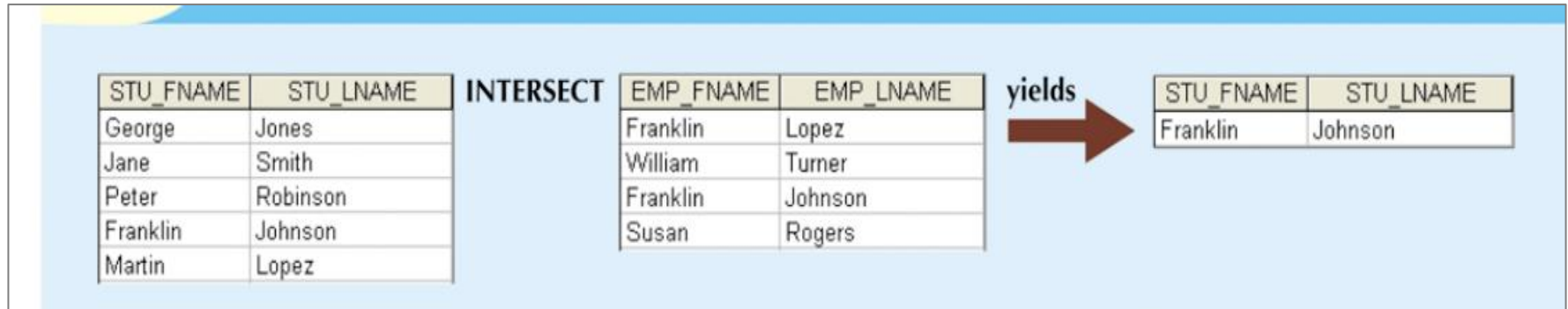
$$\Pi_{course\_id} (\sigma_{semester="Fall" \wedge year=2017}(section)) \cap \Pi_{course\_id} (\sigma_{semester="Spring" \wedge year=2018}(section))$$

- Result

<i>course_id</i>
CS-101



# Set Intersection illustration



*SOURCE: Course Technology/Cengage Learning (Morris, 2020)*

- *Yields ONLY rows found in both relations.*



# Set Difference Operation

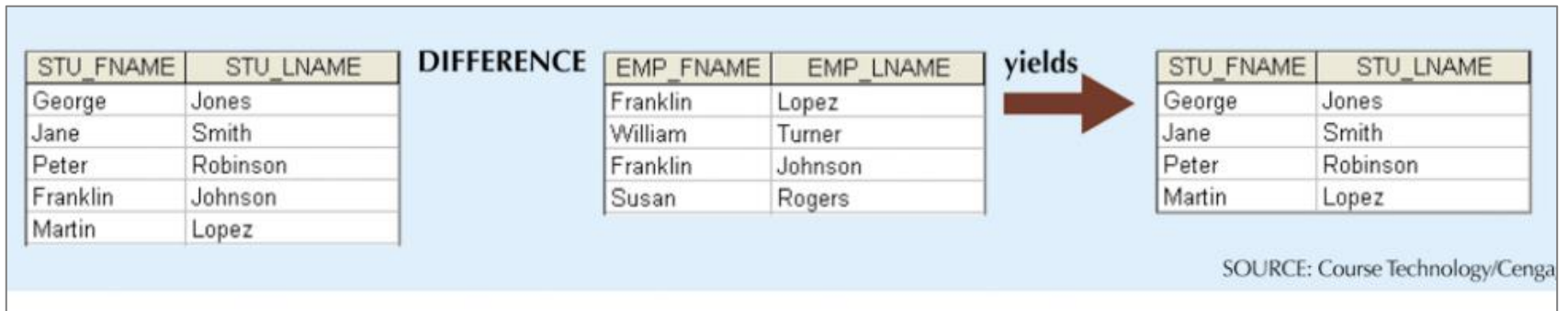
- The set-difference operation allows us to find tuples that are in one relation but are not in another.
- Notation  $r - s$
- Set differences must be taken between **compatible** relations.
  - $r$  and  $s$  must have the **same** arity
  - attribute domains of  $r$  and  $s$  must be compatible
- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$\Pi_{course\_id} (\sigma_{semester="Fall" \wedge year=2017}(section)) -$   
 $\Pi_{course\_id} (\sigma_{semester="Spring" \wedge year=2018}(section))$

<i>course_id</i>
CS-347
PHY-101



# Set Difference illustration



SOURCE: Course Technology/Cengage Learning (Morris, 2020)

- Yields all rows in one table that are not found in the other table
  - Subtracts one table from the other
  - The order of the tables is important





# The Assignment Operation

- It is convenient at times to write a relational-algebra expression by assigning parts of it to temporary relation variables.
- The assignment operation is denoted by  $\leftarrow$  and works like assignment in a programming language.
- Example: Find all instructor in the “Physics” and Music department.

$Physics \leftarrow \sigma_{dept\_name=“Physics”}(instructor)$

$Music \leftarrow \sigma_{dept\_name=“Music”}(instructor)$

$Physics \cup Music$

- With the assignment operation, a query can be written as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as the result of the query.



# The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator,  $\rho$ , is provided for that purpose
- The expression:

$$\rho_x(E)$$

returns the result of expression  $E$  under the name  $x$

- Another form of the rename operation:

$$\rho_{x(A1,A2, \dots An)}(E)$$



# Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000

- Query 1

$$\sigma_{dept\_name="Physics" \wedge salary > 90,000} (instructor)$$

- Query 2

$$\sigma_{dept\_name="Physics"} (\sigma_{salary > 90,000} (instructor))$$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.



# Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department
- Query 1

$\sigma_{dept\_name="Physics"}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$

- Query 2

$(\sigma_{dept\_name="Physics"}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$

- The two queries are not identical; they are, however, equivalent -- they give the same result on any database.



# references

- Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. 2011. *Database system concepts* (7th ed ed.). McGraw-Hill, New York
- MORRIS,CARLOS 2020. *DATABASE PRINCIPLES fundamentals of design, implementation, and management*. CENGAGE LEARNING EMEA.



## End of Chapter 2