



Chapter 2: Intro to Relational Model

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan
See 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 an *Instructor* Relation

The diagram illustrates a relation table for 'Instructor'. The table has four columns: *ID*, *name*, *dept_name*, and *salary*. There are 12 rows of data, each representing a tuple. Arrows point from the column headers to the column itself, labeled 'attributes (or columns)'. Arrows point from the row labels to the row, labeled 'tuples (or rows)'.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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



Attribute (屬性)

- 屬性指物件的特性, 或欲處理的資料
- The set of allowed values for each attribute is called the **domain** of the attribute
 - 例子: the domain of *ID* = {11, 12, 13, 14}
- Attribute values are (normally) required to be **atomic**; that is, indivisible (分割後沒有意義)
 - *multivalued* attribute values are **not** atomic-> 每個value是一個集合
例子: the domain of *phone-number* = {{6601, 6602}, {6611, 6622}}
 - *composite* attribute values are **not** atomic-> 每個value是不同類型資料的組合
例子: the domain of *phone-number* = {(02, 24622192)}
- The special value **null** is a member of every domain, indicating that the value is “unknown” or “does not exist”.
- The null value causes complications in the definition of many operations



Relation Schema and Instance (補充正式定義)

- 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*)

- Formally, given sets D_1, D_2, \dots, D_n , a **relation r** is a *subset of*
$$D_1 \times D_2 \times \dots \times D_n$$
Thus, a relation is **a set of n -tuples** (a_1, a_2, \dots, a_n) where each $a_i \in D_i$
- *Example:*

$D_1 = \{a, b, c\}$, $D_2 = \{1, 2\}$, $D_1 \times D_2 = \{(a, 1), (a, 2), (b, 1), (b, 2), (c, 1), (c, 2)\}$

$r = \{(a, 1), (a, 2), (b, 2)\}$

- The current values of a relation is called the **relation instance**.
A **relation (instance)** is represented as a **table**.
 - An **attribute** refers to a **column** of a table.
 - An element t of r is a **tuple**, represented by a **row** in a table.

A1	A2
a	1
a	2
b	2



Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
 - 資料的排序方式屬physical level, 非logical level
 - 寫query時不知資料如何排序;
- 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

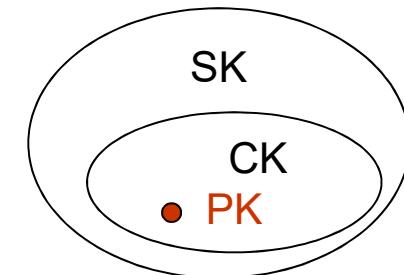
- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
 - schema: *instructor (ID, name, dept_name, salary)*
 - Instance:

<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
Example: $\{ID\}$ is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
 - which one? 通常依一般使用習慣, 找最有代表性的.
 - Another example: {系別, 年級, 班級, 座號}, {學號}, {身分證字號} 都是下面classmate表格的candidate key, 選 {學號}做為PK.



學號	身分證字號	系別	年級	班級	座號	姓名
0995701	A123456780	CS	3	A	1	張三
0995705	A123456781	CS	3	A	2	李四
0995711	A123456782	CS	3	B	1	王五
0985701	A123456783	CS	4	A	1	趙六
0995601	A123456784	EE	3	A	1	劉七



Foreign keys

- Foreign key constraint: Value in one relation must appear in another
 - Referencing relation, 如下面的instructor
 - Referenced relation, 如下面的department
 - Example: *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*
- 注意: 屬性值的唯一性會隨表格表示的資料不同而改變,如*dept_name* , 所以表格的PK會變化

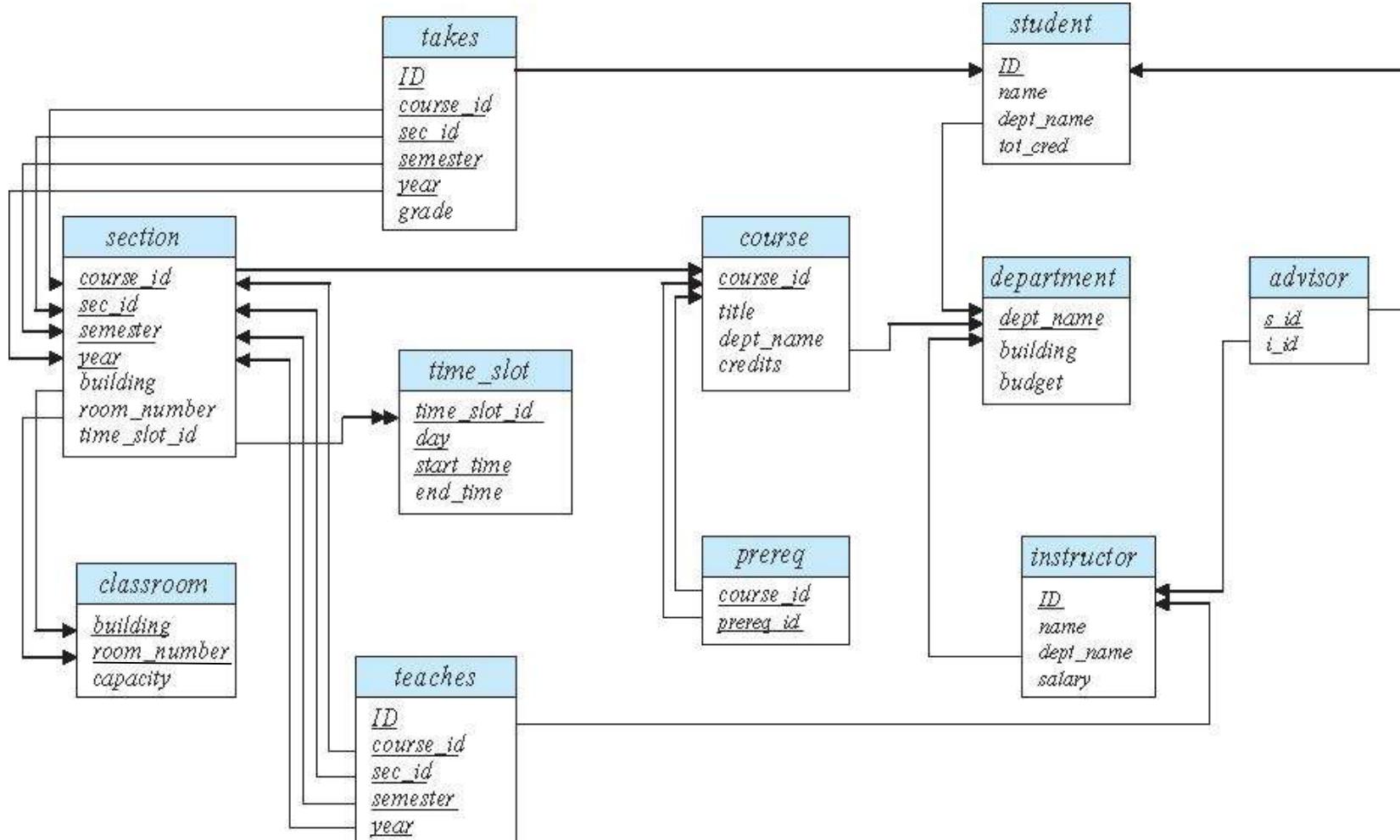
ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000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

dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

■ department
■ instructor



Schema Diagram for University Database





資料綱要常見寫法 (課本圖2.8)

- classroom (building, room_number, capacity)
- department (dept_name, building, budget)
- course (course_id, title, dept_name, credits)
- instructor (ID, name, dept_name, salary)
- section (course_id, sec_id, semester, year, building, room_number, time_slot_id)
- teaches (ID, course_id, sec_id, semester, year)
- student (ID, name, dept_name, tot_cred)
- takes (ID, course_id, sec_id, semester, year, grade)
- advisor(s_ID, i_ID)
- time_slot (time_slot_id, day, start_time, end_time)
- prereq (course_id, prereq_id)
- 底線表示PK。



Relational Query Languages

- Language in which users request information from the database.
- 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 turning-machine equivalent
 - Consists of 6 basic operations
- 算術代數: $1+5*3$
- 學習relational algebra的重點
 - 使用哪些relation
 - 使用哪些operator



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: σ
 - project: Π
 - union: \cup
 - set difference: $-$
 - Cartesian product: \times
 - rename: ρ <- 上課不會教
- Two additional operators
 - join: \bowtie
 - Intersection: \cap



Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- p is called the **selection predicate (限制式)**
 - 兩個屬性值互相比較，或屬性值和常數比較
 - 多個限制式可用and或or連接起來（看下一頁）
- 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 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 than \$90,000, we write:
$$\sigma_{dept_name = "Physics"} \wedge salary > 90,000 (instructor)$$
- Then 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 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 (Cont.)

- 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



Composition of Relational Operations

- The result of a relational-algebra operation is a relation, and therefore 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 attributes 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

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
...
...

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

■ teaches



Join Operation

- The Cartesian-Product
 - $instructor \times teaches$ 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 (關於) instructors and the courses that they taught, we write:
$$\sigma_{instructor.id = teaches.id} (instructor \times teaches)$$
- The result of this expression, shown in the next slide



Join Operation (Cont.)

- The table corresponding to:

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

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



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 “union” 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
- $$instructor \bowtie_{Instructor.id = teaches.id} teaches.$$
- Natural join 要求兩個表格共同屬性其值相等的資料列才能配對輸出，寫法為

instructor \bowtie *teaches*

第四章會再做進一步說明



練習

- Find the titles of courses in the Comp. Sci. department

Answer:

- Output the attributes ID and building of instructors.

(老師的編號和辦公室所在的建築物名稱)

Answer:



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: 2nd column of r deals with the same type of values as does the 2nd column of s)
- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$$\begin{aligned} & \Pi_{course_id} (\sigma_{semester="Fall"} \wedge year=2017 (section)) \cup \\ & \Pi_{course_id} (\sigma_{semester="Spring"} \wedge year=2018 (section)) \end{aligned}$$



Union Operation (Cont.)

- Result of:

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

course_id	course_id	sec_id	semester	year	building	room_number	time_slot_id
CS-101	BIO-101	1	Summer	2017	Painter	514	B
CS-315	BIO-301	1	Summer	2018	Painter	514	A
CS-319	CS-101	1	Fall	2017	Packard	101	H
CS-347	CS-101	1	Spring	2018	Packard	101	F
FIN-201	CS-190	1	Spring	2017	Taylor	3128	E
HIS-351	CS-190	2	Spring	2017	Taylor	3128	A
MU-199	CS-315	1	Spring	2018	Watson	120	D
PHY-101	CS-319	1	Spring	2018	Watson	100	B
	CS-319	2	Spring	2018	Taylor	3128	C
	CS-347	1	Fall	2017	Taylor	3128	A
	EE-181	1	Spring	2017	Taylor	3128	C
	FIN-201	1	Spring	2018	Packard	101	B
	HIS-351	1	Spring	2018	Painter	514	C
	MU-199	1	Spring	2018	Packard	101	D
	PHY-101	1	Fall	2017	Watson	100	A

■ section



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.

$$\begin{aligned} & \prod_{course_id} (\sigma_{semester='Fall'} \wedge year=2017 (section)) \cap \\ & \prod_{course_id} (\sigma_{semester='Spring'} \wedge year=2018 (section)) \end{aligned}$$

- Result

course_id
CS-101



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

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

course_id
CS-347
PHY-101



※ 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.

$$\text{Physics} \leftarrow \sigma_{\text{dept_name} = \text{“Physics”}}(\text{instructor})$$
$$\text{Music} \leftarrow \sigma_{\text{dept_name} = \text{“Music”}}(\text{instructor})$$
$$\text{Physics} \cup \text{Music}$$

- 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, ρ , 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:

$$\rho_{x(A_1, A_2, \dots, A_n)}(E)$$



Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about 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 (Cont.)

- Example: Find all information about instructors in the Physics department, including the courses taught by them.
- 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.