CSE 180, Database Systems I
Shel Finkelstein, UC Santa Cruz
Lecture 2

Chapter 2: Intro to Relational Model (including Relational Algebra)`

Database System Concepts, 7th Edition.

Modified by Shel Finkelstein for CSE 180

Important Notices

- Lab1 assignment has been posted on Piazza under Resources→Lab1. See Piazza announcements **(including corrections)** about Lab1.
  - General Information file about Labs has been posted under Resources→General Resources.
  - Lab1 will be discussed at Lab Sections.
    - You won't have enough information from Lectures to complete Lab1 until Monday, January 13, unless you go to Lab Section or read the textbook.
    - The files in BeerScriptsRI.zip under under Resources→Lab Section Notes can also help a lot.
    - Data loading file has been posted; use of that file is not required for Lab1, but it may help you test your solution. **But note that this file has an intentional error!**
  - Lab1 is due on Canvas as a zip file by **Sunday, January 19, 11:59pm**.
    - Late Lab Assignments will not be accepted.
    - Be sure that you post the correct file!
    - Canvas will be used for both Lab submission and grading.
  - This is the easiest Lab Assignment, so be sure to complete it … early!
- You should have Gradiance access this week via Lab Sections.
  - Gradiance Assignment #1 will be posted soon.

Can You Answer These Questions?

Not a homework; will answer in class later this week, but for CSE 180, you should know how to answer all of them.

0-If set S is {1,3,5,7} and set T is {2,3,5,7}, what are S UNION T and S INTERSECT T?

1-If set A is {1,2,3} and set B is {u,v,w,x,y}, how many ways can you pick pairs of items, with the first from A and the second from B?

2-If you have a set of employees (with names and salaries) where John makes 10K, George makes 20K, Ringo makes 30K and Paul makes 40K, what are the names of the employee(s) who make less than the average salary?

3-Can there ever be an employee who makes more than every employee? If so, give an example. If not, explain why not

4-Write the truth-table for p AND q, where p can be TRUE or FALSE and q can be TRUE or FALSE.

Relational Model and Cartesian Product

- The relational data model (Edgar F. Codd, 1970)
  - Data is described and represented by the mathematical concept of a *relation*.

- What is a relation?
  - A structure with rows and columns
  - A subset of a Cartesian Product of sets
  - What is the Cartesian Product of {a,b,c,d} and {1,2,3}?

Cartesian Product

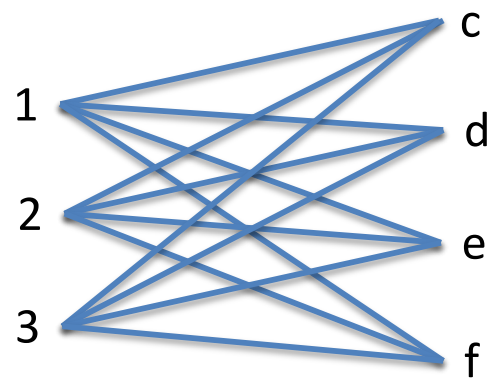
- What is the Cartesian Product of {a,b,c,d} and {1,2,3}?

  { (a,1), (a,2), (a,3),
  (b,1), (b,2), (b,3),
  (c,1), (c,2), (c,3),
  (d,1), (d,2), (d,3) }

- What are some examples of relations from that Cartesian Product?

Another Cartesian Product Example

- A: {1,2,3}
- B: {d,e,f,g}
- A × B = { (1,d), (1,e), (1,f), (1,g),
  (2,d), (2,e), (2,f), (2,g),
  (3,d), (3,e), (3,f), (3,g) }



- Suppose that C = {x,y}. What would A x B x C be?

Tuples and Relations

- *Tuple*:
  - A *k-tuple* is an ordered sequence of k values (not necessarily different)
    - (1, 2) is a binary tuple or 2-tuple.
    - (a, b, b) is a ternary tuple or 3-tuple.
    - (112, 'Ann', 'CS', 'F', 3.95) is a 5-tuple.

- If D_1, D_2, …, D_k are sets of elements, then the *Cartesian Product* $D_1 \times D_2 \times \ldots \times D_k$ is the set of all k-tuples (d_1, d_2, …, d_k) such that for all i with $1 \le i \le k$, $d_i \in D_i$

- *Relation*:
  - A *k-ary relation* is a subset of $D_1 \times D_2 \times \ldots \times D_k$, where each D_i is a set of elements.
  - D_i is the *domain (or datatype)* of the *i*th column of the relation.
  - Domains may be enumerated {'AMS', 'CMPS', 'TIM'}, or may be of standard types (INTEGER, FLOAT, DATE, …).

Another Practice Homework
(Not collected, will be discussed during next Lecture)

- If D_1 has n_1 elements and D_2 has n_2 elements, then how many elements are there in $D_1 \times D_2$?
- If D_i has n_i elements, then how many elements are there in the Cartesian Product $D_1 \times \ldots \times D_k$?
- If D_i has n_i elements, then how many relation instances can one construct from $D_1 \times \ldots \times D_k$?

Example of an *instructor* Relation Instance

Figure 2.1 in textbook, the *instructor* relation.

attributes (or columns)

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

tuples (or rows)

Attribute

- 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.
- The special value ***null*** is a member of every domain. Indicates that the value is “unknown” or “does not apply”.
- The null value complicates the definition of many operations.

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order).
- Example: *instructor* relation instance with unordered tuples
  - Figure 2.4 in textbook, Unsorted display of the *instructor* relation.
  - This is equivalent to the previous instance of the *instructor* relation.

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

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: i*nstructor* (*ID, name, dept_name, salary*)
  - Instance:

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

Schemas and Instances

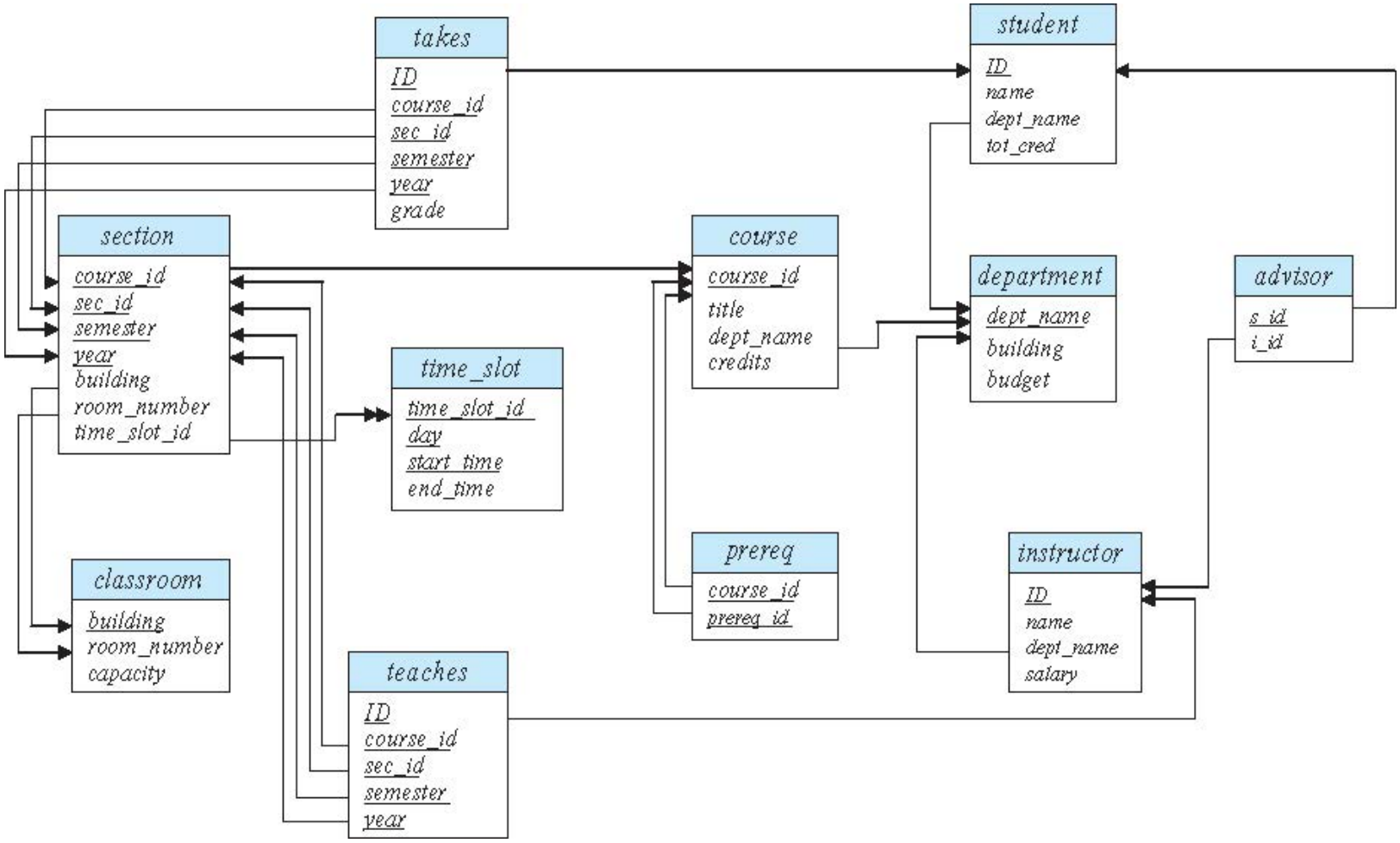
- **Relation Schema:** The logical structure of a relation, which has a name, a data type and (possibly) other "rules of the road" for rows in the relation.
  - Relation is the term use in the Relational Model to refer to a table.
- **Relational Instance** for a Relation Schema**:** A set of rows that follow the "rules of the road" type for that Relation Schema.
- **Database Schema:** A set of Relation Schemas that have different names.
- **Database Instance** for a Database Schema**:** A set of relation instances, with one instance for each of the relations in that Database Schema.

Keys

- Let r be a relation whose attributes are R. Let K $\subseteq$ R
- *K* is a **superkey** of *R* if the values of the attributes in *K* are sufficient to identify a unique tuple of each possible relation instance *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?
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* that references *department.*

Schema Diagram for University Database



Relational Query Languages

- Procedural versus declarative (or non-procedural)
- "Pure" mathematical languages:
  - Relational Algebra
    - Queries are expressed as a sequence of operations on relations.
    - Procedural language
  - Relational Calculus
    - Queries are expressed as formulas in first-order logic.
    - Declarative language
- Ted Codd proved that Relational Algebra and Relational Calculus have the same expressive power.
- We will concentrate in this chapter on Relational Algebra
  - Not Turing-machine complete
  - Consists of 5 basic operations (plus 2 housekeeping operations)

Relational Algebra

- Relational Algebra is a procedural language consisting of 5 basic operations, each of which takes one or two relations as its input, and produces one new relation as its result.
- 5 basic operations
  - select: σ
  - project: Π
  - union: $\cup$
  - set-difference: $-$
  - Cartesian Product: x
- We'll also use 2 "housekeeping" operations that make queries easier to write.
  - assignment: $\leftarrow$
  - rename: ρ

Relation Algebra Operators

- Ted Codd proved that the relational algebra operators (σ , π , x , U , -) are independent of each other. That is, you can't define any of these 5 operators using the others.

- However, there are other important operators that can be expressed using (σ , π , x , U , -).
  - Join, Natural Join, Semi-Join, Outer Join
    - Join is discussed in this Lecture.
  - Set Intersection, which is also discussed in this Lecture.
  - Division (/ or $\div$), which we won't discuss in this class.

Multisets (also called Bags) vs. Sets

From basic set theory:

- Every element in a set is distinct.
  - E.g., {2,4,6} is a set but {2,4,6,2,2} is not a set
    - … or is same set as {2,4,6}
- A multiset (or bag) may contain repeated elements.
  - E.g., {{2,4,6}} is a bag. So is {{2,4,6,2,2}}.
    - Note that double set brackets in {{2,4,6}} indicate it's a bag, not a set
    - … but we won't use "double set brackets" notation in this class.
- The order among elements in a set or multiset is not important.
  - E.g., {2,4,6} = {4,2,6} = {6,4,2}
  - {{2,4,6,2,2}} = {{2,2,2,6,4}} = {{6,2,2,4,2}}.
- It is possible to define a multiset Relational Algebra, but we'll focus on set-oriented Relational Algebra in this class.
  - Even though duplicates do matter in practical databases.

Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
  - $\sigma_p(r)$ is a unary operation, whose operand is r.
- Example: Select the tuples in the *instructor* relation where the instructor is in the Physics department.
  - Query

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

  - Result:

ID	*name*	*dept_name*	*salary*
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 logical connectives:

  $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
- Example: To find the instructors in Physics with a salary greater $90,000, we write:

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

- 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

- **project** is a unary operation that returns its argument relation, but with only a subset of the attributes kept.
- Notation:

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

where r is a relation, and A_1, A_2 , …, A_k are attributes of relation r.

- The result is defined as the relation of *k* columns obtained by keeping only the columns that are listed
- Duplicate rows removed from result, since relations are sets.
  - It is possible to define a multiset relational algebra, but we won't do that in this class.
  - Duplicates do matter in practical databases. Any idea why?

Project Operation (Cont.)

- Example: Keep just the *ID*, *name* and *salary* attributes of *instructor*, getting rid of the *dept_name* attribute.
- Query:

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

- Result:

ID	*name*	*salary*
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 relation, so 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.

$$\Pi_{name}(\sigma_{dept\_name = \text{“Physics”}} (instructor))$$

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

Cartesian Product Operation

- The **Cartesian Product** operation (denoted by X) allows us to combine information from any two relations.
- Example: the Cartesian Product of the relations *instructor* and t*eaches* is written as:

  instructor X *teaches*

- We construct a tuple of the result from 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* relation

- instructor has 12 tuples (Figure 2.1)
- teaches has 15 tuples (Figure 2.7)
- How many tuples are in the Cartesian Product?

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

  $$instructor \times teaches$$

  associates every tuple of instructor with every tuple of teaches.
  - Most of the resulting rows have information about instructors who did NOT have that particular teaching assignment.
- To get only those tuples of "*instructor* X *teaches* " that pertain to instructors and their teaching assignments, we can write:

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

  - We get only those tuples of "*instructor* X *teaches*" that pertain to instructors and their teaching assignments.
- The result of this expression is shown on the next slide.

Join Operation (Cont.)

- The relation corresponding to the result of:

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

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

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

- Join is <u>not</u> one of Codd's original Relational Algebra operations.
  - However, it is a <u>Derived Operation</u>, since it can be derived from them.

Example of a *section* Relation Instance

Figure 2.6 in textbook: The *section* relation.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	1	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

Union Operation

- The **union** operation allows us to combine all the tuples that are in either or both relation.
- Notation: $r \cup s$
- For $r \cup s$ to be valid, they must be **union-compatible**.
  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible.**
     - 2nd attribute of *r* must have the same type of values as the 2nd attribute of *s*, and the same for all the other attributes of *r* and *s*.
- Resulting relation has the same type as *r* and *s*
  - … using attributes of *r* to name the columns in the result
- Example: To find all the courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both:

$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”}\ \wedge\ year=2017}(section))\ \cup$$

$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”}\ \wedge\ year=2018}(section))$$

Example of a *section* Relation Instance

Coloring Fall 2017 and Spring 2018 tuples in the *section* relation.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	1	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

Union Operation (Cont.)

- Result of:

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

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Set-Difference Operation

- The set-difference operation allows us to find the tuples that are in the first relation but are not in the second relation.
- Notation $r - s$
- Set-differences must be taken between Union-Compatible relations.
  - r and s must have the same arity.
  - attribute domains of r and s must be compatible,
- Resulting relation has the same type as r and s
  - … using attributes of r to name the columns in the result
- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

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

course_id
CS-347
PHY-101

Intersection Operation

- The intersection operation allows us to find the tuples that are in both of the input relations.
- Notation: $r \cap s$
- Assume that r and s are union-compatible.
  - *r*, *s* have the *same arity.*
  - All the attributes of *r* and *s* are compatible.
- Resulting relation has the same type as *r* and *s*
  - … using attributes of *r* to name the columns in the result
- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

$$\Pi_{course\_id}\,(\sigma_{semester=\text{“Fall”}\ \wedge\ year=2017}(section))$$

$$\cap\ \Pi_{course\_id}\,(\sigma_{semester=\text{“Spring”}\ \wedge\ year=2018}(section))$$

- Result:

course_id
CS-101

Intersection Operation is Derived

- Intersection is a Derived Operation in Relational Algebra.
  - How can it be derived from Codd's original operations?

$$r \cap s = r - (r - s)$$
$$= s - (s - r)$$

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 instructors in the "Physics" and Music department.

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

$$MusicInstructors \leftarrow \sigma_{dept\_name="Music"}(instructor)$$

$$PhysicsInstructors \cup MusicInstructors$$

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

The Rename Operation

- The result of an relational-algebra expression does not have a name The **rename** operator, ρ , provides a way to name the result of the expression.
- The expression:

$$\rho_x(E)$$

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

- Another form of the rename operation provides a way to name the attributes, as well as the expression.

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

- What does the following expression return?

$$\Pi_{instr1.name}(\sigma_{instr1.name = instr2.name \wedge instr1.ID <> instr2.ID}$$
$$[\rho_{instr1}(instructors) \times \rho_{instr2}(instructors)])$$

Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about instructors in the Physics department whose salary is greater than 90,000.
- Query 1

$$\sigma_{dept\_name=\text{“Physics”} \wedge salary > 90000}(instructor)$$

- Query 2

$$\sigma_{dept\_name=\text{“Physics”}}(\sigma_{salary > 90000}(instructor))$$

- The two queries are not identical.
  - They are, however, **equivalent**. That is, they give the same result on any database instance.

Equivalent Queries (Cont.)

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

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

- Query 2

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

- These two queries are <u>not</u> identical.
  - Notice the subtle difference as to when select is performed.
  - They are, however, **equivalent**. That is, they give the same result on any legal database instance.

- We'll say a lot more about Equivalent Queries later in the course.