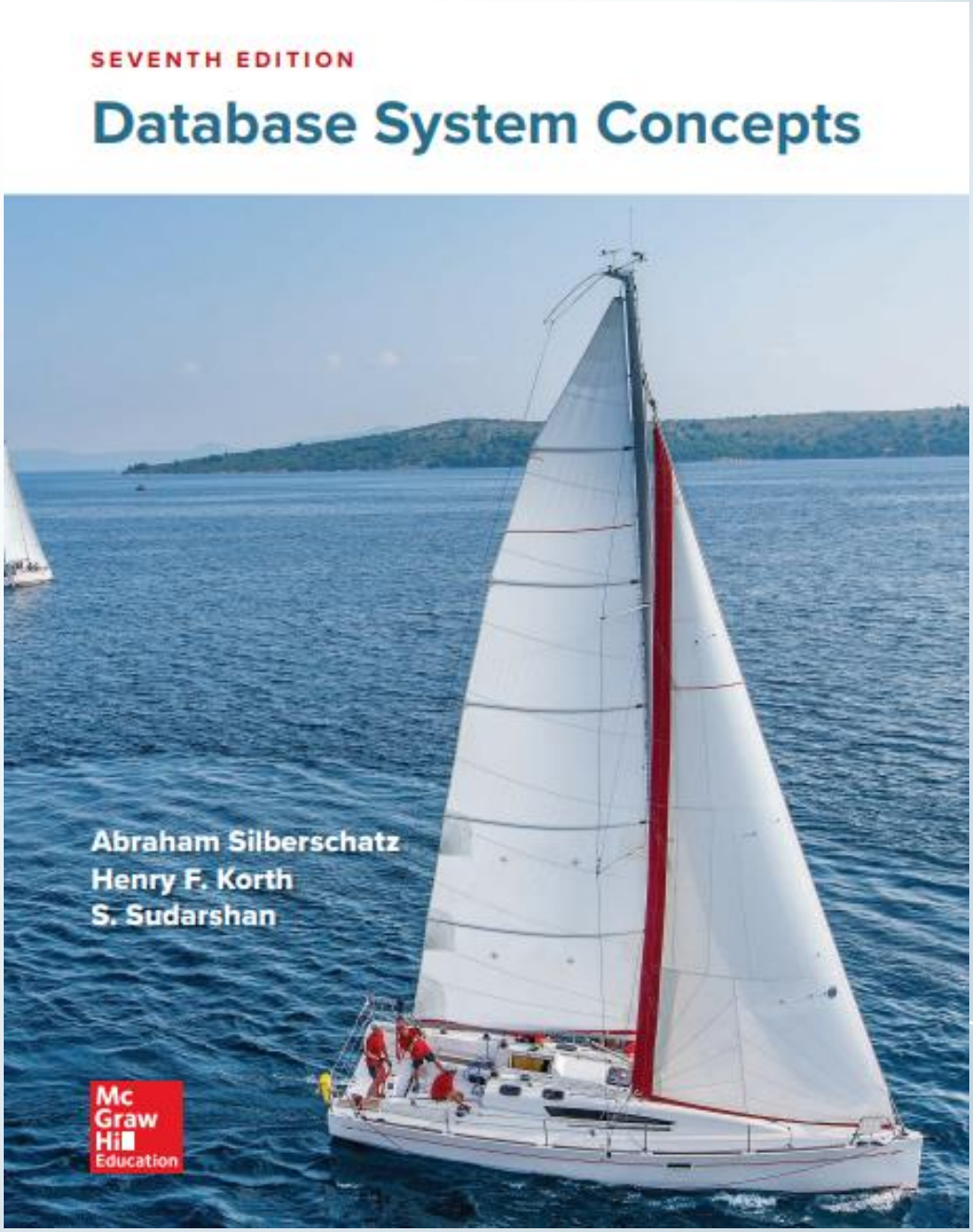
اصول طراحی پایگاه داده

By Dr. Taghinezhad

Mail:
a0taghinezhad@gmail.com



Chapter 2: Intro to Relational Model

Database System Concepts, 7th Ed.

Outline

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

Example of a *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)

Relation Schema and Instance

- A_1, A_2, …, A_n are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ 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
- 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

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: *instructor* (*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

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?
- **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* referencing *department*

Keys

- What are the Superkeys?
- What are the Candidate Keys?

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

Example

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Ardabil
s3	Pooladin	Isfahan

P

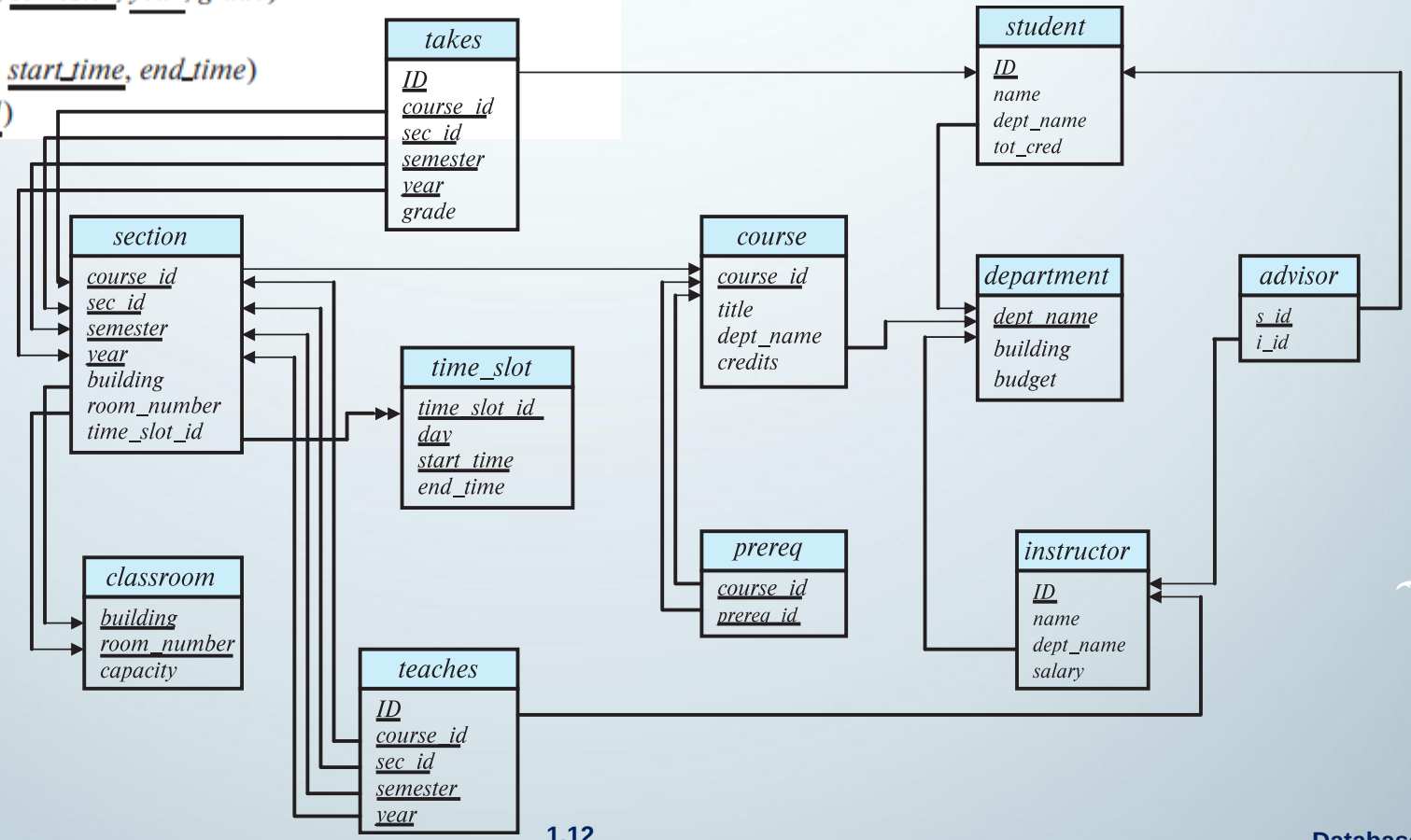
P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Ardabil
P3	Blue	Brass	Isfahan
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Schema Diagram for University Database

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*)



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: σ
  - project: Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ

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

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 connectives:

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

  $$\sigma_{dept\_name=\text{“}Physics\text{”} \wedge salary > 90,000}(instructor)$$
- The select predicate may include comparisons between two attributes.
  - Example, find all departments whose name is the same as their building name:

  $$\sigma_{dept\_name=building}(department)$$

Select Operation – selection of rows (tuples)

- Relation r: how to select A=B and D bigger than 5

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

- Select r where A=B and D>5

Project Operation

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

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

where A_1, A_2, …, 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 the result, since relations are sets

Project Operation Example

- Example: eliminate the *dept_name* attribute of *instructor*
- 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 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.

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

- 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 X) allows us to combine information from any two relations.
- Example: the **Cartesian** product of the relations ***instructor*** and ***teaches*** is written as:

  instructor X *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

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
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

joining two relations -- Cartesian-product

- Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-product – naming issue

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*r.B*	*s.B*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Join Operation

- The Cartesian-Product

  instructor X ***teaches***

  associates every tuple of instructors 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:

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

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

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

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*)

Union Operation

- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

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*)

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

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

Union of two relations

- Relations *r, s:*

اجتماع دو رابطه رابطه ای است که تاپلهایش در یک یا هر دو رابطه وجود دارند. یعنی رکوردهایی ۲ جدول با هم ترکیب شده و رکوردهای تکراری یکبار نوشته می شوند.

- r ∪ s:
- r union s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

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.

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

- Result

course_id
CS-101

Set intersection of two relations

- Relation *r, s*:

A	*B*
α	1
α	2
β	1

r

A	*B*
α	2
β	3

s

- $r \cap s$
- *r intersect s*

A	*B*
α	2

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

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

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

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 ← 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, 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:

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

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

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

Query 2

$$\sigma_{dept\_name=\text{“Physics”}}(\sigma_{salary > 90.000}(instructor))$$

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

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

Query 2

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

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

Example

Name of P2 producers

- اسامی تهیه کنندگان قطعه P2

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Example

- اسامی تهیه کنندگان قطعه P2 را تهیه میکنند

Name of P2 producers

$$\Pi_{Sname}\,(\sigma_{P\#=P2}\,(S \infty SP))$$

S **join** SP **giving** Temp1
Select Temp1 **where** P# = ‘P2’ **Giving** Temp2
Project Temp2 [Sname]

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Example

- اسامی تهیه کنندگان را که حداقل یک قطعه قرمز رنگ تهیه می کنند

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Example

- اسامی تهیه کنندگان را که حداقل یک قطعه قرمز رنگ تهیه می کنند

$Temp1 \leftarrow \sigma_{Color='Red'}(P)$
$Temp2 \leftarrow \prod_{p\#}(Temp1)$
$Temp3 \leftarrow (temp2 \infty SP)$
$Temp4 \leftarrow \prod_{s\#}(Temp3)$
$Temp5 \leftarrow \prod_{s\#}(Temp4 \infty S)$
$Temp6 \leftarrow \prod_{sname}(Temp5)$

Select P where color='Red' Giving Temp1
Project Temp1 [P#] Giving Temp2
Temp2 Join SP Giving Temp3
Project Temp3 [S#] Giving Temp4
Temp4 Join S Giving Temp5
Project Temp5 [Sname] Giving Temp6

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Example

- اسامی تهیه کنندگانی را بیابید که قطعه P2 را تهیه نمی کنند

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Example

- اسامی تهیه کنندگانی را بیابید که قطعه P2 را تهیه نمی کنند

$$\Pi_{Sname}\left[\left[\Pi_{s\#}(S)-\left(\Pi_{s\#}(\sigma_{P\#='P2'}(SP))\right)\right]\infty S\right]$$

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

پیوند شرطی

- عملگر پیوند شرطی (θ-join)
- این عملگر، زیر مجموعه ای از ضرب دکارتی است که شرط θ روی سطرهای آن اعمال شده باشد. ستونهای خروجی معادل ستونهای ضرب دکارتی است.
- $\mathbf{X}_{\theta}$
- مثال خروجی دستور زیر

$$S \; \mathbf{X}_{s.city>=P.city} \; P$$

S#	Sname	City	P#	Color	Type	City
s1	Fanavaran	Tehran	P1	Red	Iron	Tehran
s1	Fanavaran	Tehran	P2	Green	Copper	Tabriz
s1	Fanavaran	Tehran	P4	Red	Iron	Tehran
s2	Iran Segment	Tabriz	P2	Green	Copper	Tabriz
s3	Pooladin	Tabriz	P3	Blue	Brass	Tabriz

Natural Join

- R=

A	B
X	Y
X	Z
Y	Z
Z	V

S=

B	C
Z	U
V	W
Z	V

- R |×| S=

A	B	C
X	Z	U
X	Z	V
Y	Z	U
Y	Z	V
Z	V	W

Semi-join

- مشابه پیوند طبیعی است با این تفاوت که فقط ستونهای جدول اول را می دهد.

$$(\sigma_{city = ''Tehran''}(P) \propto (\sigma_{Qty > 200}(SP)$$

S

S#	Sname	City
s1	Fanavaran	Tehran
s2	Iran Segment	Tabriz
s3	Pooladin	Tabriz

P

P#	Color	Type	City
P1	Red	Iron	Tehran
P2	Green	Copper	Tabriz
P3	Blue	Brass	Shiraz
P4	Red	Iron	Tehran

SP

S#	P#	Qty
s1	P1	300
s2	P2	200
s3	P3	400
s2	P1	300
s2	P2	400
s3	P2	200

Outer Join

- ۱- فراپیوند چپLeft Outer Join : L-O-JOIN
- ۲- فراپیوند راستRight Outer Join:
- ۳- فراپیوند کامل Full Outer Join
- فراپیوند چپ گونه ای از عملگر پیوند طبیعی است با این تفاوت که علاوه بر تاپلهای پیوند شدنی از دو رابطه، تاپلهای نشدنی از رابطه چپ هم ، پیوند شده با مقادیر NULL، در جواب وارد می شود در فراپیوند راست، تاپلهای پیوند نشدنی از رابطه سمت راستی ، پیوند شده با مقادیر NULL وارد می شوند.
- مثال:
- S L-O-JOIN SP

S

S#	Sname	Status
s1	Sn1	10
s2	Sn2	15
S3	Sn3	10
S4	Sn4	20

SP

S#	P#	Qty
S1	P1	200
S2	P3	400
S2	P4	100
S3	P1	300

Outer Join

- مثال:
- S L-O-JOIN SP

S#	Sname	Status	P#	Qty
s1	Sn1	10	P1	200
s2	Sn2	15	P3	400
s2	Sn2	15	P4	100
S3	Sn3	10	P1	300
S4	Sn4	20	NULL	NULL

S

S#	Sname	Status
s1	Sn1	10
s2	Sn2	15
S3	Sn3	10
S4	Sn4	20

SP

S#	P#	Qty
S1	P1	200
S2	P3	400
S2	P4	100
S3	P1	300

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID, salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor) - \Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

Divide operator

Suppose we have this extra table, in the Bank database

Account-types

Type
checking
savings

Suppose we would like to know which customers have alltypes of accounts.

Divide Operation

÷ or / divide $(\pi_{\text{Owner, Type}}\text{Account}) \div \text{Account-types}$

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Account-types	Type
	checking
	savings

	Owner
	J. Smith

Find account owners who have ALL types of accounts.

Divide Definition

- For R + S where R (r1, r2, r3, r4) and S(s1, s2)

- Since S has two attributes, there must be two attributes in R (say r3 and r4) defined on the same domains, respectively, as **s1** and **s2**. We could say that(**r3**, **r4**) is union-compatible with (**s1**, **s2**).

- The query answer has the remaining attributes (**r1, r2**).
- And the answer has a tuple, (r1, r2), in the answer if the(r1, r2) value appears with *every* S tuple in R.

When to divide?

- Not supported as a primitive operator, but useful for expressing queries like:
  - Find customers who have <u>all</u> types of accounts.
- Let **A have 2** fields, ***x , y***, B have only field ***y***.
  - A/B= $\{(x) | \exists (x,y) \in A \quad \forall (y) \in B\}$
  - i.e., A/B contains all x tuples (customers) such that for every y tuple (account type) in B, there is an xy tuple in A.
  - Or. If the set of y values (account types) associated with an x value(customer) in A contains all y values in B, the x value is in A/B.
- In general, x and y can be any lists of fields; y is the list of fields in B, and x U y is the list of fields of A.- A/B =

Division Example

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

End of Chapter 2