Database Systems
Lecture02 – Introduction to Relational Model

SUNG KYUN KWAN UNIVERSITY

Beomseok Nam (남범석)
bnam@skku.edu

Example of a 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 Types

- The set of allowed values for each attribute is called the **domain** (D_i) of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are ***attributes***

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

  Example:

  instructor = (*ID, name, dept_name, salary*)

- a relation is a set of tuples

- a ***tuple*** is represented by a *row* in a table

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
  - i.e., a relation is a set.
- 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

- A database consists of multiple relations
- e.g.) a university database consists of the following relations

  instructor
  student
  advisor

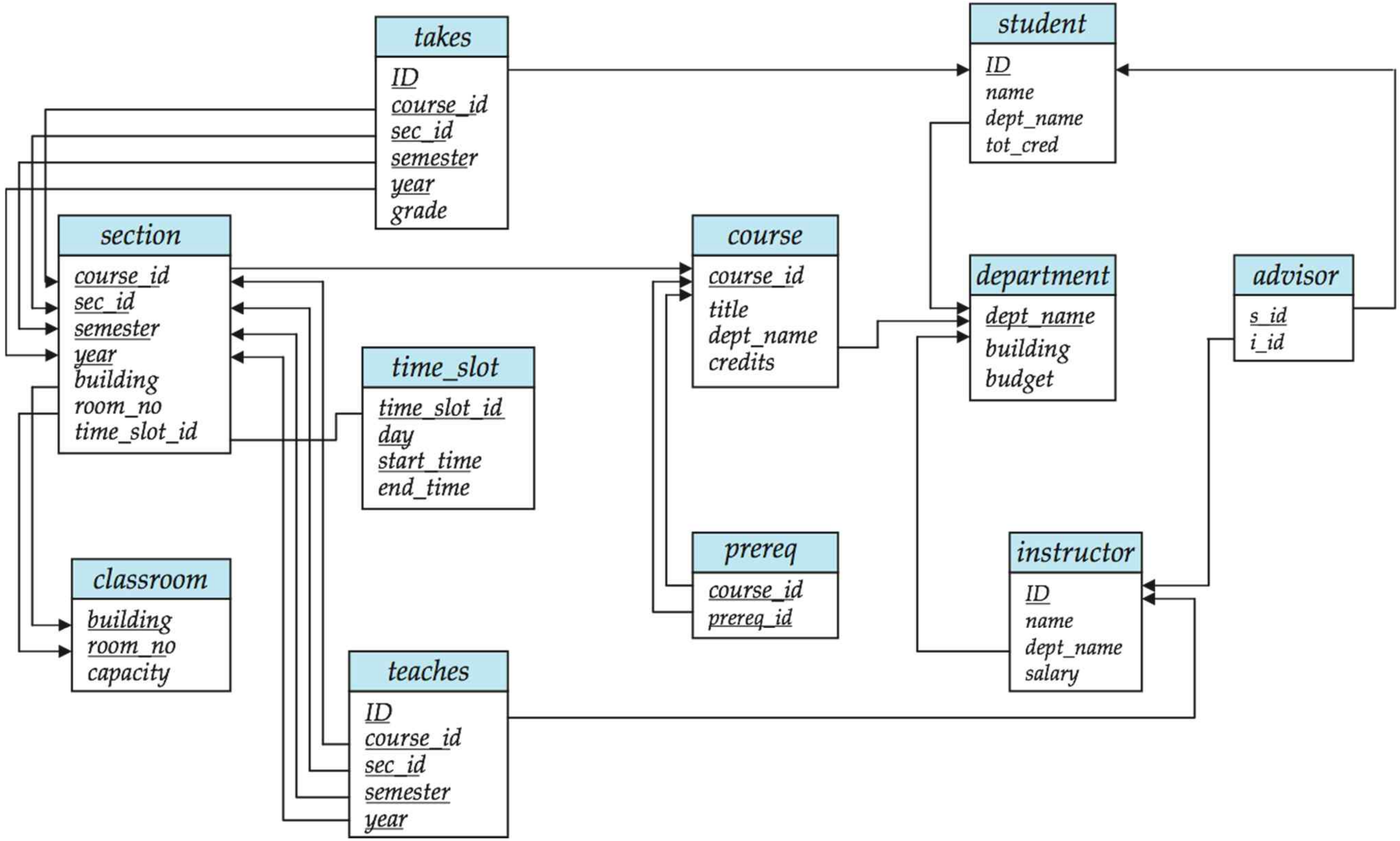
- Bad design:
  univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)
  results in
  - repetition of information (e.g., two students have the same instructor)
  - the need for null values (e.g., represent a student with no advisor)

Keys

- Let $K \subseteq R$
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple in relation *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**.
- **Foreign key** constraint: Value in one relation must appear in another relation

Schema Diagram for University Database



Relational Query Languages

- "Pure" languages that have the same expression power with structured query language (SQL).
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus

- Relational operators
  - Relational query languages use 5 simple relational operators to represent user query

Selection of tuples (σ)

- Relation r

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

- Select tuples with A=B and D > 5

$$\sigma_{\text{A=B and D > 5}}\,(r)$$

A	B	C	D
α	α	1	7
β	β	23	10

Projection of attributes (Π)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- Select A and C
  - Projection

$\Pi_{A,\,C}\,(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Joining two relations – Cartesian Product (x)

- 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

Union of two relations (∪)

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations (–)

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r – s:

A	B
α	1
β	1

Set Intersection of two relations (∩)

- Relation *r, s*:

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

$r \cap s$

A	B
α	2

Joining two relations – Natural Join ($\bowtie$)

- Let *r* and *s* be relations on schemas *R* and *S* respectively.
  Then, the "natural join" of relations r and s is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on R
    - *t* has the same value as t_s on *S*

Natural Join Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s

- Natural Join

  r ⋈ s

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

Recap: Relational Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary>=85000}}(instructor)$
	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.
⋈ (Natural Join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
× (Cartesian Product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

Ch 3 Introduction to SQL

Data Definition Language

- The SQL **data-definition language (DDL)** allows the specification of information about relations, including:

- The schema for each relation.
- The domain of values associated with each attribute.
- Integrity constraints
- And as we will see later, also other information such as
  - The set of indices to be maintained for each relations.
  - Security and authorization information for each relation.
  - The physical storage structure of each relation on disk.

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified length *n.*

- **int.** Integer
- **smallint.** Small integer

- **numeric(p,d).** Fixed point number
  - with user-specified precision of *p* digits, with *d* digits to the right of decimal point.
- **real.** Floating point number
- **float(n).** Floating point number
  - with user-specified precision of at least *n* digits.

Create Table Construct

- An SQL relation is defined using the **create table** command:

$$\textbf{create table } r\ (A_1\ D_1,\ A_2\ D_2,\ ...,\ A_n\ D_n,\\ (\text{integrity-constraint}_1),\\ ...,\\ (\text{integrity-constraint}_k))$$

  - r is the name of the relation
  - each A_i is an attribute name in the schema of relation r
  - D_i is the data type of values in the domain of attribute A_i

- Example:

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2))

- **insert into** *instructor* **values** ('10211', 'Smith', 'Biology', 66000);
- **insert into** *instructor* **values** ('10211', null, 'Biology', 66000);

Integrity Constraints in Create Table

- **not null**
- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*

Example: Declare *dept_name* as the primary key for *department*

.

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department)*

primary key declaration on an attribute automatically ensures **not null**

And a Few More Relation Definitions

- **create table** *student* (
  ID **varchar**(5),
  name **varchar**(20) not null,
  dept_name **varchar**(20),
  tot_cred **numeric**(3,0),
  primary key (*ID*),
  foreign key *(dept_name*) **references** *department)*);

- **create table** *takes* (
  ID **varchar**(5),
  course_id **varchar**(8),
  sec_id **varchar**(8),
  semester **varchar**(6),
  year **numeric**(4,0),
  grade **varchar**(2),
  primary key *(ID, course_id, sec_id, semester, year),*
  foreign key (*ID*) **references** *student,*
  foreign key (*course_id, sec_id, semester, year*) **references**
  section);

And more still

- **create table** *course* (
  course_id **varchar**(8) **primary key**,
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  foreign key *(dept_name*) **references** *department)*);

  - Primary key declaration can be combined with attribute declaration as shown above

Drop and Alter Table Constructs

- **drop table** *student*
  - Deletes the table schema and its contents
- **delete from** *student*
  - Deletes all contents of table, but retains table schema
- **alter table**
  - **alter table** *r* **add** *A D*
    - where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
    - All tuples in the relation are assigned *null* as the value for the new attribute.
  - **alter table** *r* **drop** *A*
    - where *A* is the name of an attribute of relation *r*
    - <u>Dropping of attributes is not supported by many databases</u>

Basic Query Structure

- The SQL **data-manipulation language (DML)** provides the ability to query information, and insert, delete and update tuples
- A typical SQL query has the form:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.

- Note: The result of an SQL query is also a relation.

The select Clause

- The **select** clause list the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra

- Example: find the names of all instructors:

select *name*
from *instructor*

- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g. *Name* ≡ *NAME* ≡ *name*

The select Clause (Cont.)

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select**.**
- Find the names of all departments of instructor, and remove duplicates

select distinct *dept_name*
from *instructor*

- The keyword **all** specifies that duplicates should not be removed.

select all *dept_name*
from *instructor*

The select Clause (Cont.)

- An asterisk in the select clause denotes "all attributes"

  select *
  from *instructor*

- The **select** clause can contain arithmetic expressions involving the operation, +, –, $*$, and /, and operating on constants or attributes of tuples.

- The query:

  select *ID, name, salary/12*
  from *instructor*

  would return all tuples from the *instructor* relation, except that the value of the attribute *salary* is divided by 12.

The where Clause

- The **where** clause specifies conditions that the result must satisfy

- To find all instructors in Comp. Sci. dept with salary > 80000

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

- Comparison results can be combined using the logical connectives **and, or,** and **not.**

The from Clause

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product *instructor X teaches*

select *
from *instructor, teaches*

  - generates every possible instructor – teaches pair, with all attributes from both relations
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra)

Cartesian Product: *instructor X teaches*

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

teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

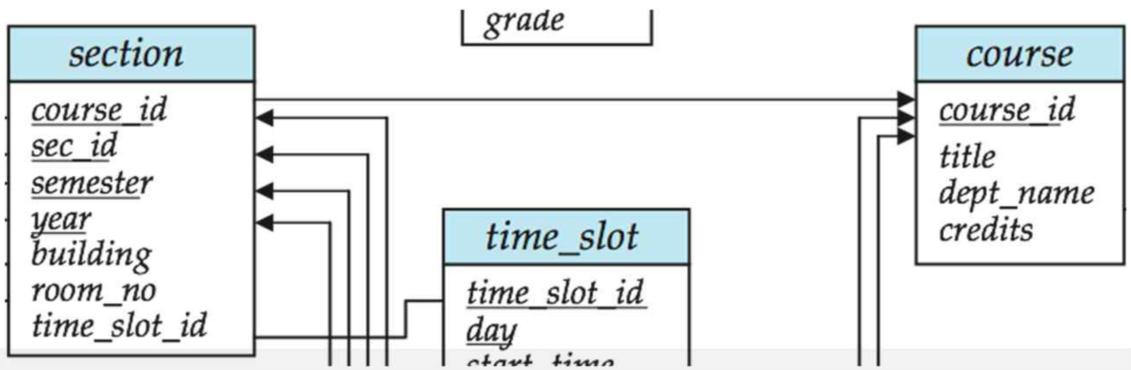
Joins

- For all instructors who have taught some course, find their names and the course ID of the courses they taught.

select *name, course_id*
from *instructor, teaches*
where *instructor.ID = teaches.ID*

- Find the course ID, semester, year and title of each course offered by the Comp. Sci. department

select *section.course_id, semester, year, title*
from *section, course*
where *section.course_id = course.course_id* **and**
dept_name = 'Comp. Sci.'



Natural Join

- Natural join matches tuples with the same values for all common attributes, and retains only one copy of each common column
- **select** *
  from *instructor* **natural join** *teaches*;

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010

Natural Join Example

- List the names of instructors along with the course ID of the courses that they taught.

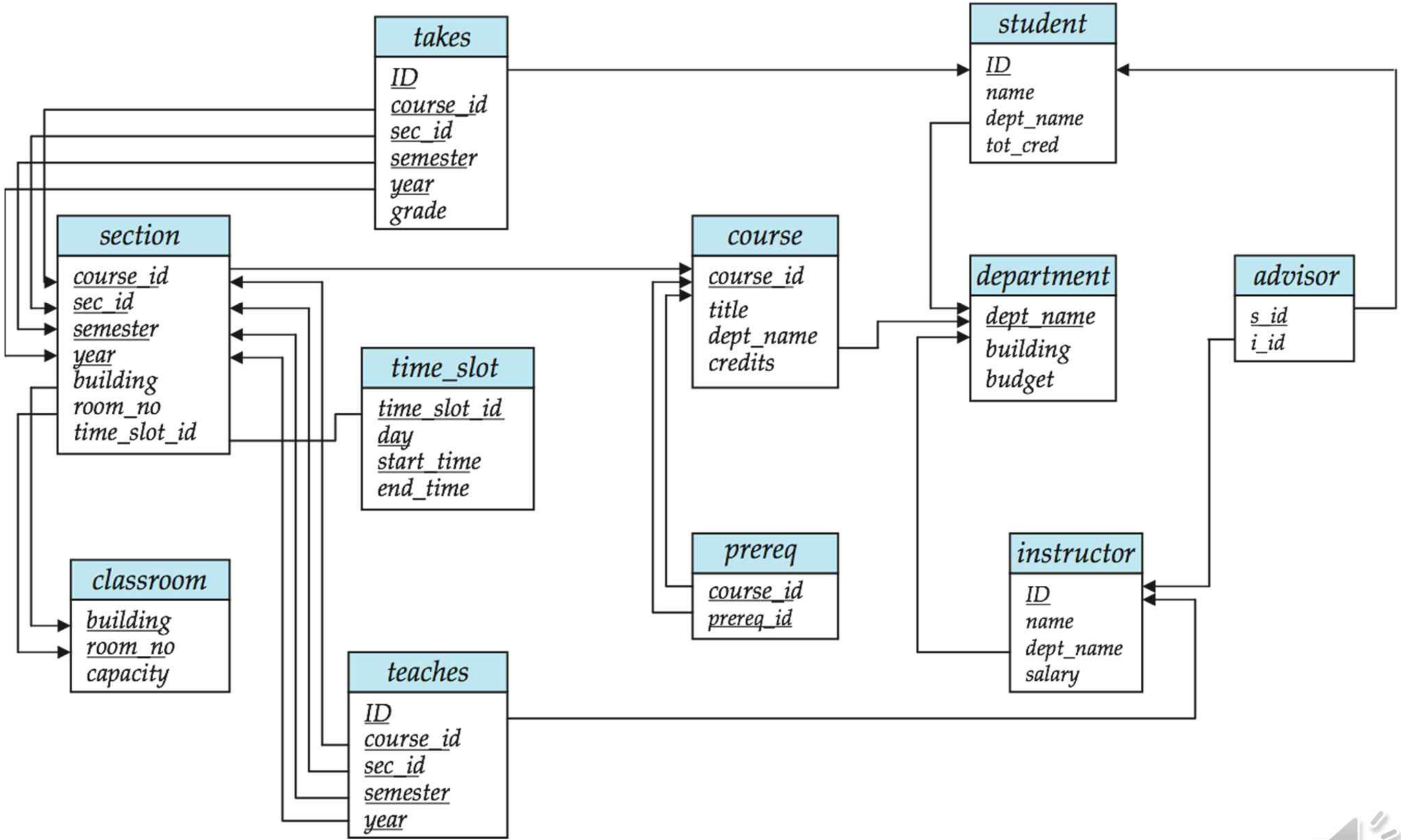
  - **select** *name*, *course_id*
    from *instructor, teaches*
    where *instructor.ID* = *teaches.ID*;

  - **select** *name*, *course_id*
    from *instructor* **natural join** *teaches*;

Natural Join (Cont.)

- Danger in natural join: beware of unrelated attributes with same name which get equated incorrectly

- List the names of instructors along with the the titles of courses that they teach
  - Incorrect version (makes course.dept_name = instructor.dept_name)
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches* **natural join** *course*;
  - Correct version
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches*, *course*
      where *teaches*.*course_id* = *course*.*course_id*;
  - Another correct version
    - **select** *name*, *title*
      from (*instructor* **natural join** *teaches*)
      join *course* **using**(*course_id*);



takes
ID
course_id
sec_id
semester
year
grade
student
ID
name
dept_name
tot_cred
section
course_id
sec_id
semester
year
building
room_no
time_slot_id
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
time_slot
time_slot_id
day
start_time
end_time
classroom
building
room_no
capacity
prereq
course_id
prereq_id
instructor
ID
name
dept_name
salary
teaches
ID
course_id
sec_id
semester
year

Natural Join (Cont.)

- Danger in natural join: beware of unrelated attributes with same name which get equated incorrectly

- List the names of instructors along with the the titles of courses that they teach
  - Incorrect version (makes course.dept_name = instructor.dept_name)
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches* **natural join** *course*;
  - Correct version
    - **select** *name*, *title*
      from *instructor* **natural join** *teaches*, *course*
      where *teaches*.*course_id* = *course*.*course_id*;
  - Another correct version
    - **select** *name*, *title*
      from (*instructor* **natural join** *teaches*)
      join *course* **using**(*course_id*);