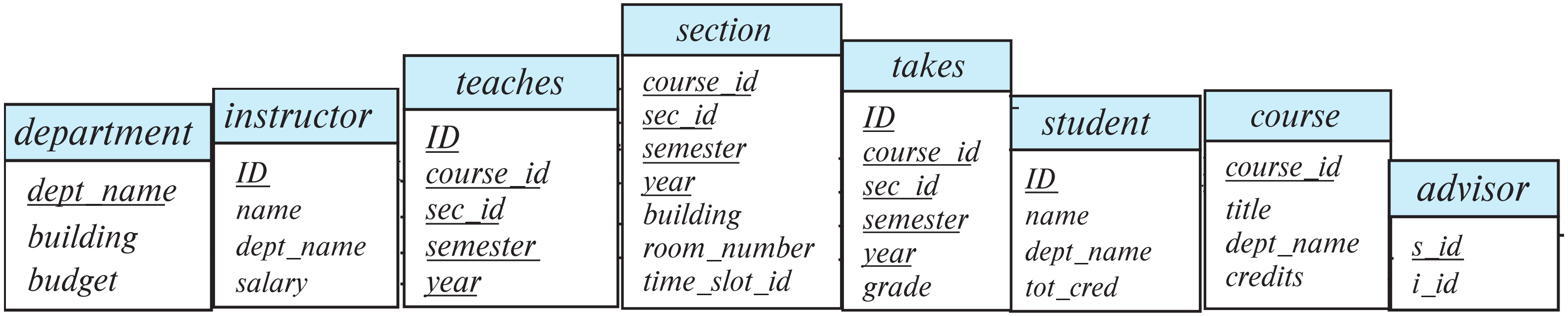
Introduction to SQL

Miao Qiao
The University of Auckland

Basic Query Structure

- A typical SQL query has the form:

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



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

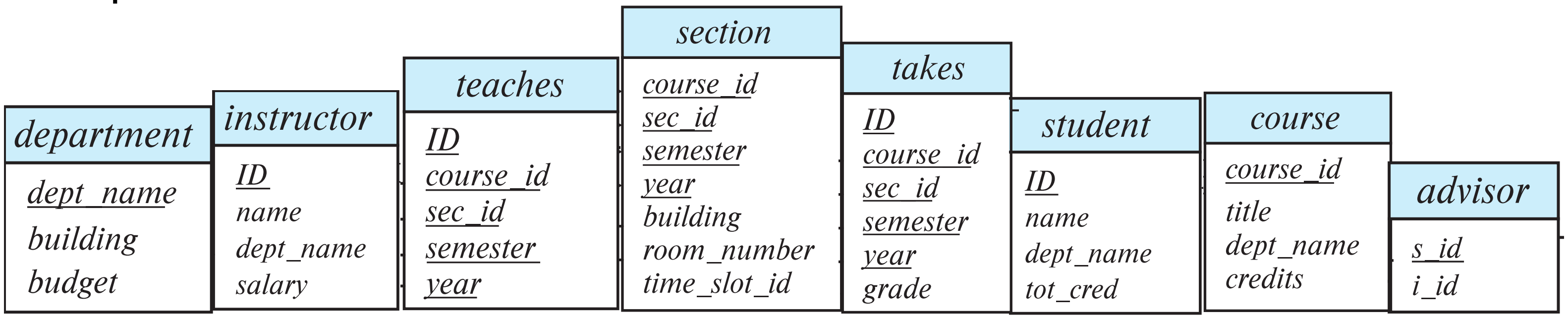
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Basic Query Structure (demo)

1. Find the names of all instructors
2. Find the department names of all instructors, and remove duplicates
3. Find the department names of all instructors, not removing duplicates
4. Find all attributes of instrctor show the entire instructor table
5. Find a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12
6. Find all instructors in Comp. Sci. dept with salary > 70000
7. Find the names of all instructors who have taught some course and the course_id
8. Find the names of all instructors in the Comp. Sci. department who have taught some course and the course_id
9. Find the names of all instructors who have a higher salary than some instructor in 'Comp. Sci'.



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

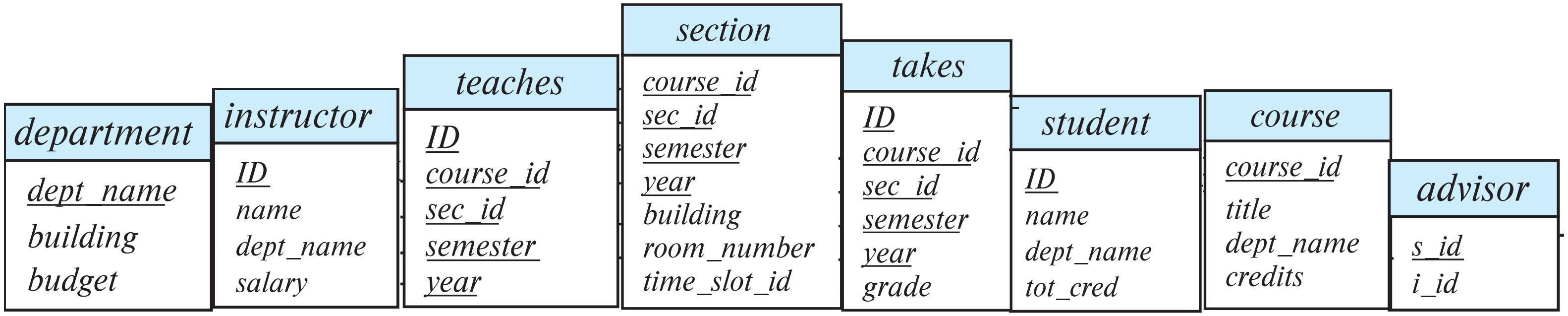
course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

Basic Query Structure (demo)

- Find the names of all instructors whose name includes the substring “in”.
- String Operations
  - The operator **like** uses patterns that are described using two special characters:
    - percent (%). The % character matches any substring.
    - underscore (_). The _ character matches any character

- Find the names of all instructors whose name has 4 characters.
- Find the names of all instructors whose name has at leaset 4 characters.



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

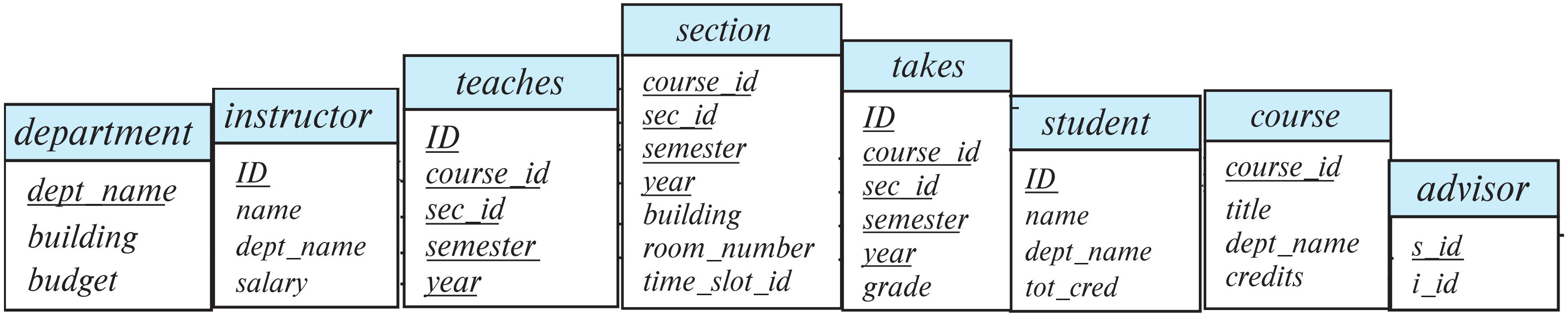
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Basic Query Structure (demo)

1. List in alphabetic order the names of all instructors
2. List in descending alphabetic order the names of all instructors
3. List in order of the combination of the names and salary of all instructors
4. Find the names of all instructors with salary between $90,000 and $100,000
5. Find courses that ran in Fall 2017 or in Spring 2018
6. Find courses that ran in Fall 2017 and in Spring 2018
7. Find courses that ran in Fall 2017 but not in Spring 2018
8. Find courses that ran in Fall 2017 or in Spring 2018, retain all duplications
9. Find all instructors whose salary is null
10. Find all instructors whose salary is not null
11. Null under and, or, with true/false



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Basic Query Structure (demo)

- These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

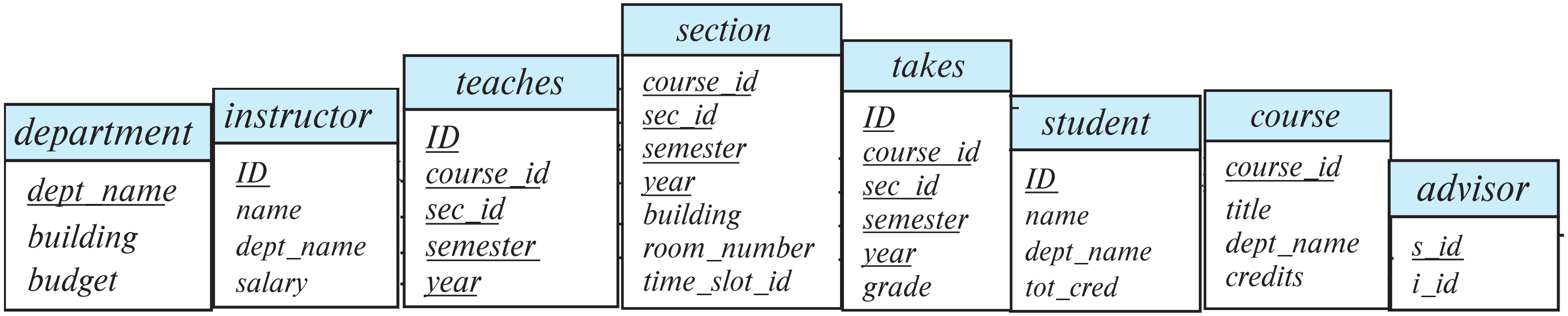
select $A_1, A_2, ..., A_n$ ← Aggregation function over values over multiple rows

from $r_1, r_2, ..., r_m$

where P

group by columns ← New clauses

having condition ←



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

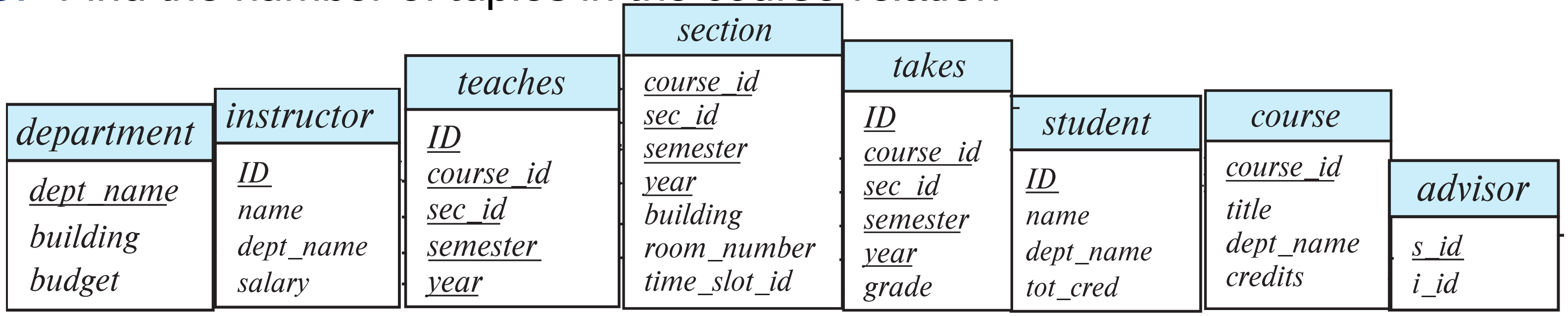
advisor: s_id, i_id

Basic Query Structure (demo)

- These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

1. Find the highest salary of any instructor
2. Find the average salary of instructors in the Computer Science department
3. Find the lowest salary of an instructor who have taught a course
4. Find the total number of instructors who teach a course in the Spring 2018 semester
5. Find the number of tuples in the *course* relation



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

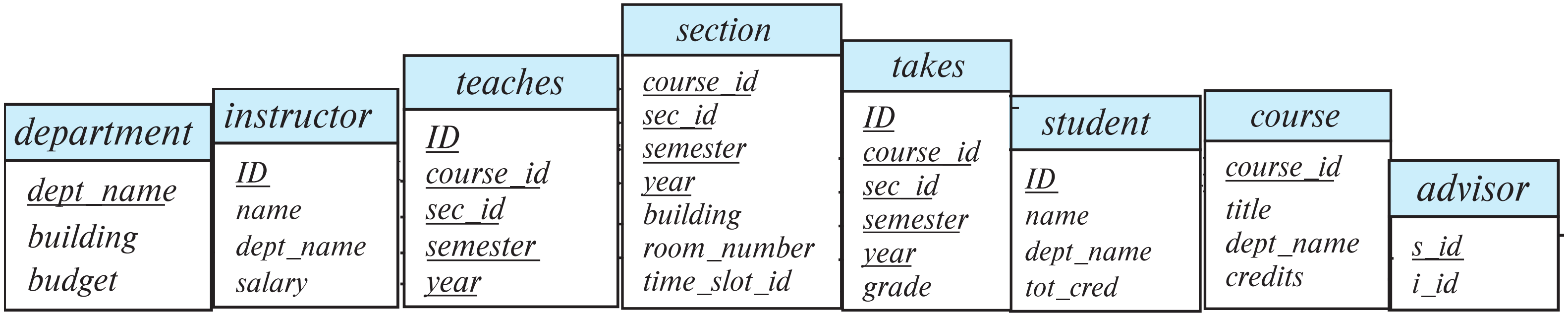
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Aggregate – Group By - Having

1. Find the average salary of instructors in each department
2. Find the names and average salaries of all departments whose average salary is greater than 42000
3. Find the names and average salaries of all departments over instructors whose salary is greater than 70000
4. Find the names and average salaries of all departments whose average salary is greater than 70000
5. Find the average salaries of instructors who have taught a course



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Nested Subqueries

- A **subquery** is a **select-from-where** expression that is nested within another query.
- The nesting can be done in the following SQL query

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

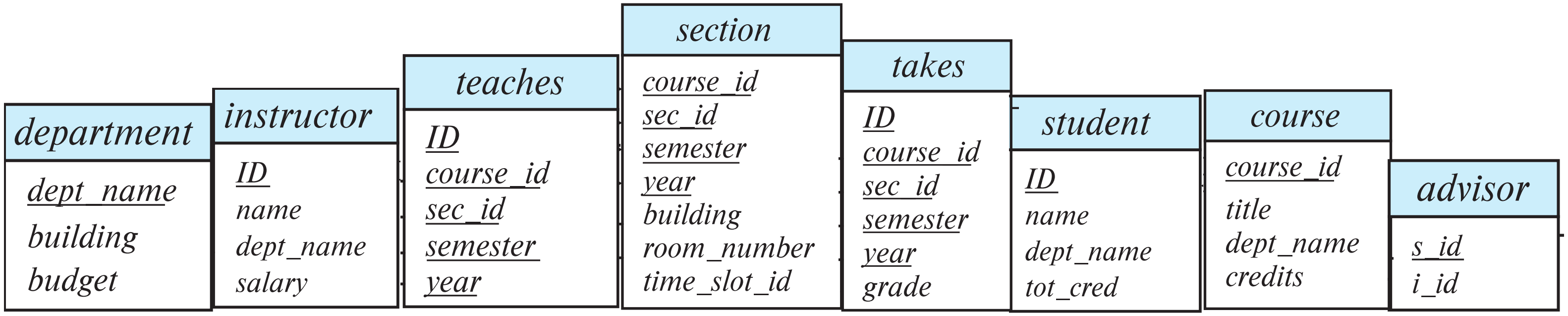
  as follows:

  - **From clause:** r_i can be replaced by any valid subquery
  - **Where clause:** P can be replaced with an expression of the form:

    B <operation> (subquery)

    B is an attribute and <operation> to be defined later.
  - **Select clause:**

    A_i can be replaced be a subquery that generates a single value.



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

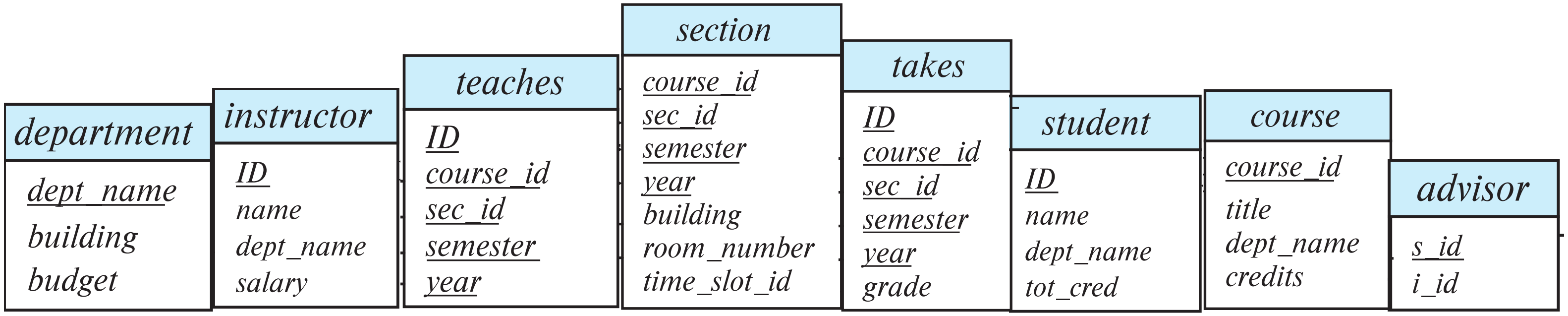
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

Subquery in Where Clause

1. Name all instructors whose name is either “Mozart” or Einstein”
2. Name all instructors whose name is neither “Mozart” nor Einstein”
3. Find the total number of (distinct) students who have taken course sections taught by the instructor with *ID* 10101 or 12121 or 15151
4. Find names of instructors with salary greater than that of some (at least one) instructor in the Computer Science department.
5. Find all instructors earning the highest salary (there may be more than one with the same salary).
6. Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

Definition of "some" Clause

- F <comp> **some** *r* ⇔ ∃ *t* ∈ *r* such that (F <comp> *t*)
  Where <comp> can be: $<, \leq, >, =, \neq$

$(5 < \textbf{some} \begin{array}{|c|}\hline 0\\\hline 5\\\hline 6\\\hline\end{array}) = \text{true}$ (read: 5 < some tuple in the relation)

$(5 < \textbf{some} \begin{array}{|c|}\hline 0\\\hline 5\\\hline\end{array}) = \text{false}$

$(5 = \textbf{some} \begin{array}{|c|}\hline 0\\\hline 5\\\hline\end{array}) = \text{true}$

$(5 \neq \textbf{some} \begin{array}{|c|}\hline 0\\\hline 5\\\hline\end{array}) = \text{true (since } 0 \neq 5)$

(= **some**) ≡ **in**

However, (≠ **some**) $\not\equiv$ **not in**

Definition of “all” Clause

- F <comp> **all** *r* ⇔ ∀ *t* ∈ *r* (F <comp> *t*)

(5 < **all** $\begin{array}{|c|}\hline 0\\\hline 5\\\hline 6\\\hline\end{array}$) = false

(5 < **all** $\begin{array}{|c|}\hline 6\\\hline 10\\\hline\end{array}$) = true

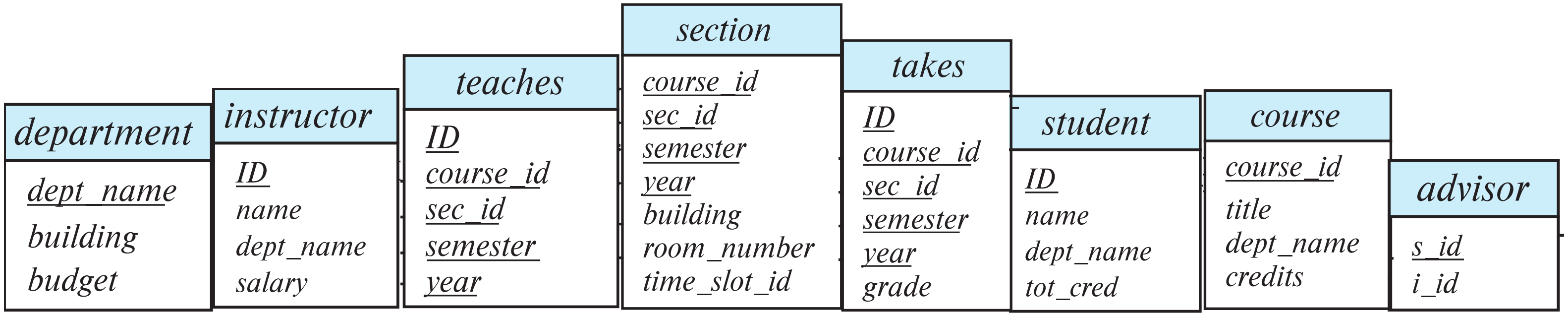
(5 = **all** $\begin{array}{|c|}\hline 4\\\hline 5\\\hline\end{array}$) = false

(5 ≠ **all** $\begin{array}{|c|}\hline 4\\\hline 6\\\hline\end{array}$) = true (since 5 ≠ 4 and 5 ≠ 6)

(≠ **all**) ≡ **not in**
However, (= **all**) ≢ **in**

Subquery in Where Clause

- The **exists** construct returns the value **true** if the argument subquery is nonempty.
  - Find all courses taught in both the Fall 2017 semester and in the Spring 2018 semester
- The **unique** construct tests whether a subquery has any duplicate tuples in its result.
- The **unique** construct evaluates to “true” if a given subquery contains no duplicates .
  - Find all courses that were offered at most once in 2017



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

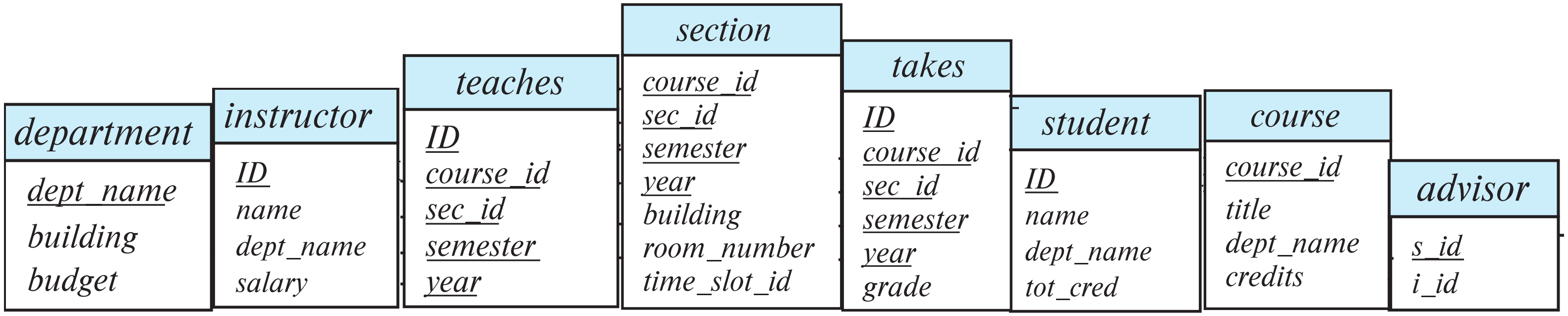
student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

advisor: s_id, i_id

Query Quest

1. Find names of instructors with salary greater than that of some (at least one) instructor in the Computer Science department.
   - Use self-join, some, exists, aggregation
2. Find all instructors earning the highest salary (there may be more than one with the same salary).
   - Use self-join, all, exists, aggregation



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

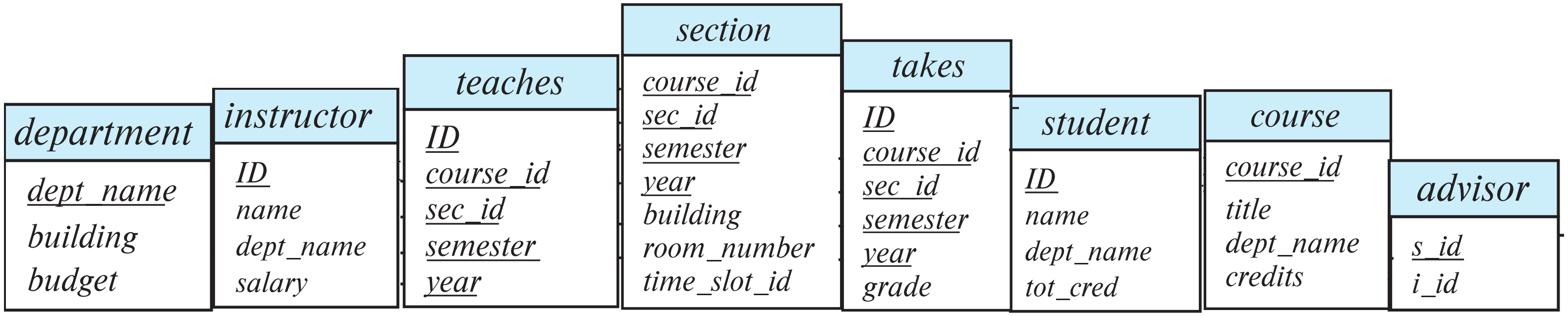
student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

advisor: s_id, i_id

Subquery in From and Select Clauses

1. Find the average instructors’ salaries of those departments where the average salary is greater than $42,000”
2. Find all departments with the maximum budget (with clause)
3. Find all departments where the total salary is greater than the average of the total salary at all departments
4. List all departments along with the number of instructors in each department

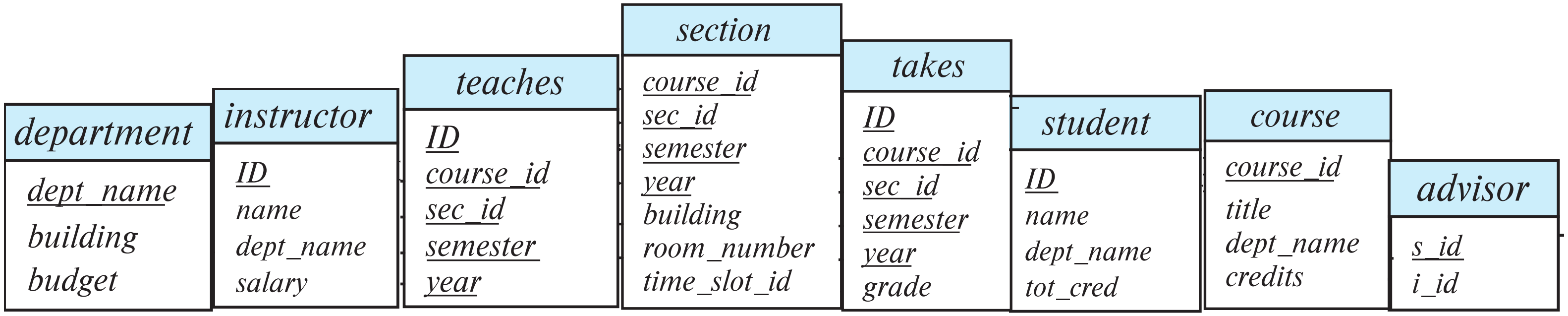


department	instructor	teaches	section	takes	student	course	advisor
dept_name	ID	ID	course_id	ID	ID	course_id	s_id
building	name	course_id	sec_id	course_id	name	title	i_id
budget	dept_name	sec_id	semester	sec_id	dept_name	dept_name	
	salary	semester	year	semester	tot_cred	credits	
		year	building	year			
			room_number	grade			
			time_slot_id				

With Clause

- The **with** clause provides a way of defining a temporary relation whose definition is available only to the query in which the **with** clause occurs.
- Find all departments with the maximum budget

```
with max_budget (value) as
        (select max(budget)
         from department)
select department.name
from department, max_budget
where department.budget = max_budget.value;
```



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

advisor: s_id, i_id

Modification of the Database

- Deletion of tuples from a given relation.
- Insertion of new tuples into a given relation
- Updating of values in some tuples in a given relation

- **delete from** *a relation* **where** condition

Deletion

- Delete all instructors

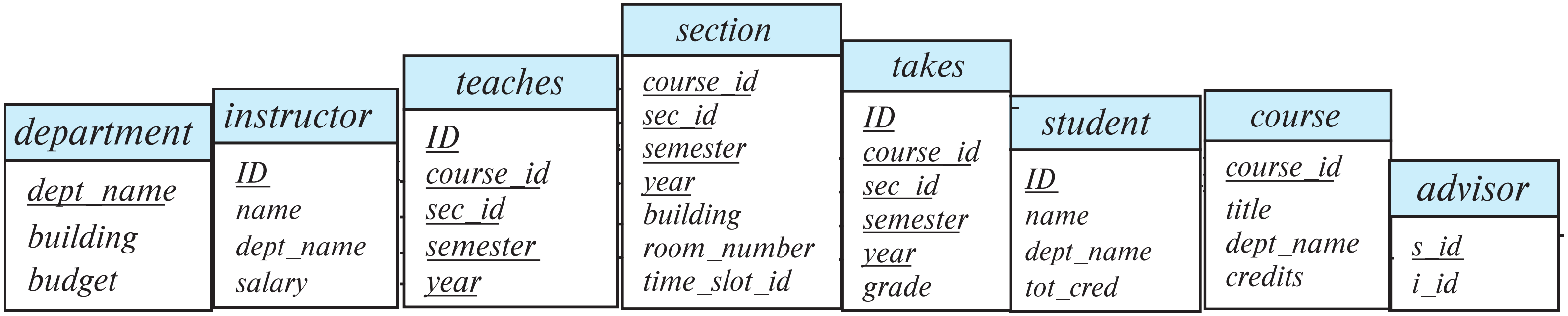
```
delete from instructor
```

- Delete all instructors from the Finance department

```
delete from instructor
where dept_name= 'Finance';
```

- *Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building.*

```
delete from instructor
where dept name in (select dept name
                    from department
                    where building = 'Watson');
```



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

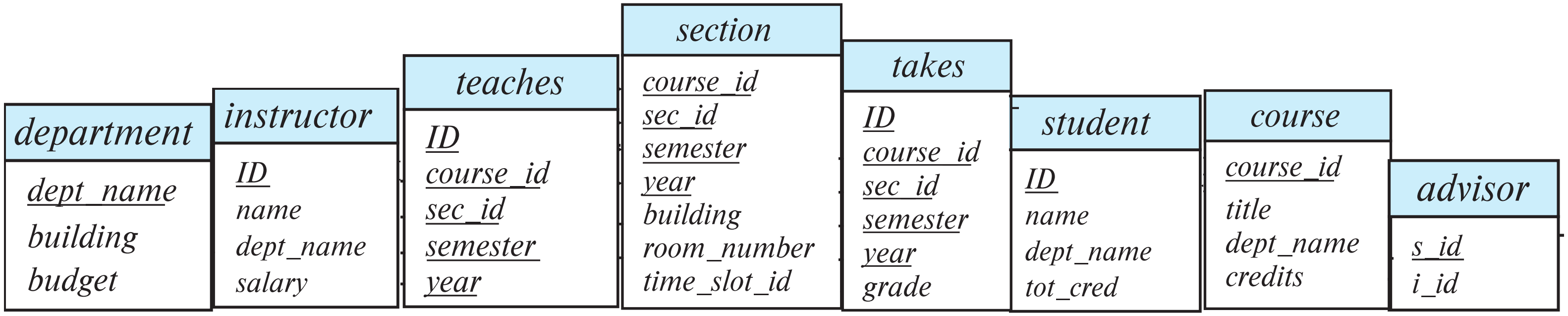
advisor: s_id, i_id

Deletion (Cont.)

- Delete all instructors whose salary is less than the average salary of instructors

```
delete from instructor
where salary < (select avg (salary)
                from instructor);
```

- Problem: as we delete tuples from *instructor*, the average salary changes
- Solution used in SQL:
  1. First, compute **avg** (salary) and find all tuples to delete
  2. Next, delete all tuples found above (without recomputing **avg** or retesting the tuples)



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

Insertion

- Add a new tuple to *course*

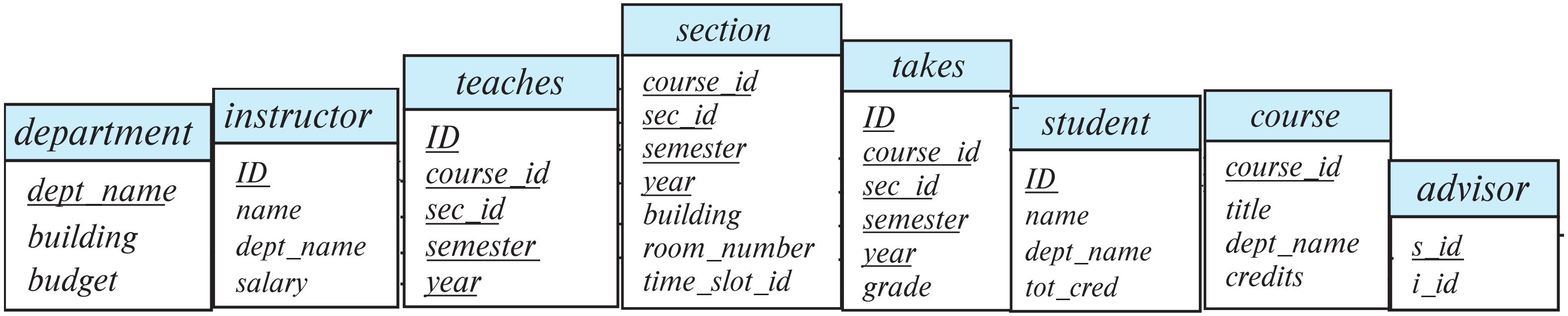
  insert into *course*
  values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

- or equivalently

  insert into *course* (*course_id*, *title*, *dept_name*, *credits*)
  values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

- Add a new tuple to *student* with *tot_creds* set to null

  insert into *student*
  values ('3003', 'Green', 'Finance', *null*);



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

Insertion (Cont.)

- Make each student in the Music department who has earned more than 144 credit hours an instructor in the Music department with a salary of $18,000.

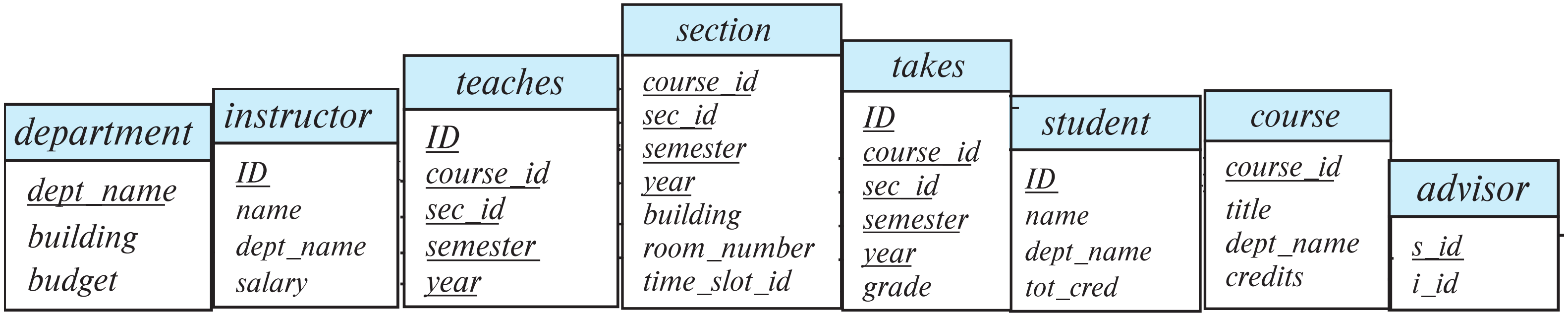
  insert into *instructor*
  select *ID, name, dept_name, 18000*
  from *student*
  where *dept_name* = 'Music' **and** *total_cred* > 144;

- The **select from where** statement is evaluated fully before any of its results are inserted into the relation.

Otherwise queries like

insert into *table*1 **select** * **from** *table*1

would cause problem



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

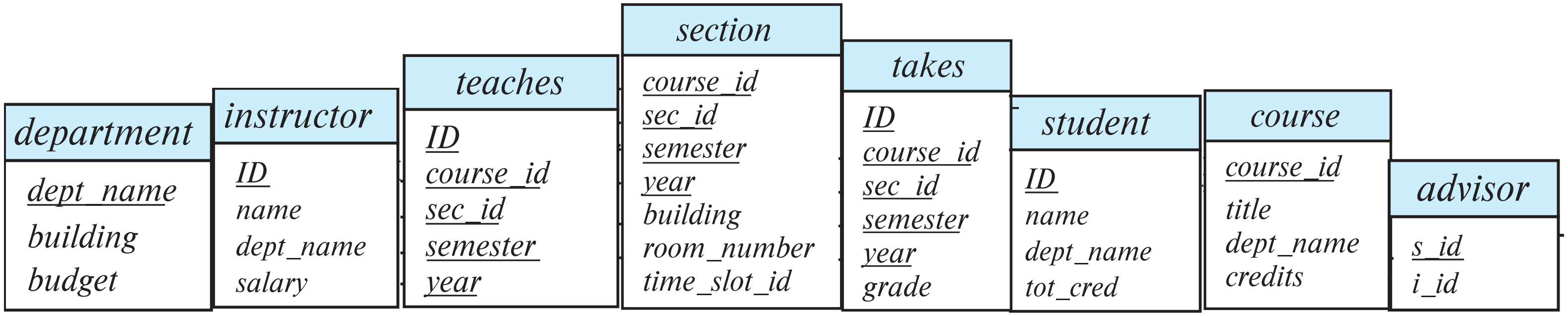
advisor
s_id
i_id

Updates

1. Give a 5% salary raise to all instructors

2. Give a 5% salary raise to those instructors who earn less than 70000

3. Give a 5% salary raise to instructors whose salary is less than average



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

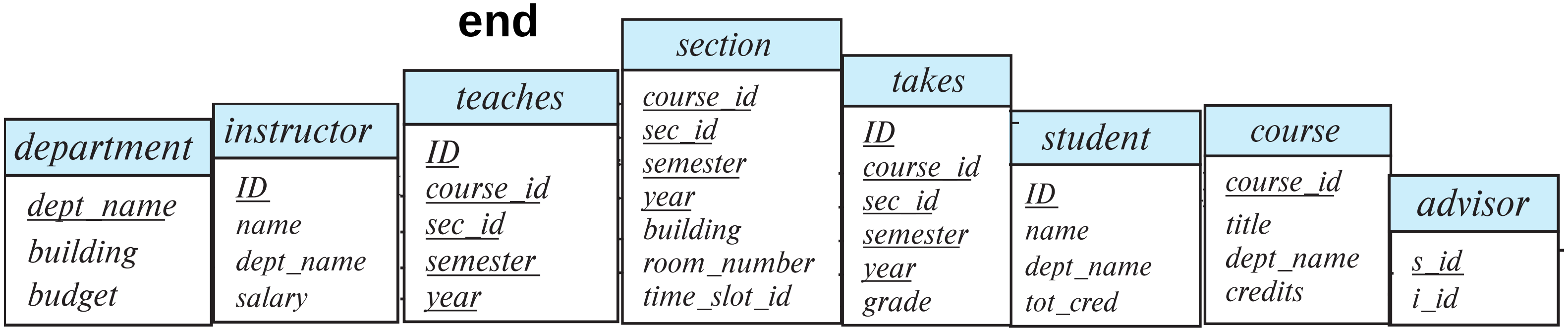
advisor: s_id, i_id

Updates (Cont.)

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%

  -
    ```
    update instructor
                  set salary = salary * 1.03
                  where salary > 100000;
              update instructor
                   set salary = salary * 1.05
                   where salary <= 100000;
    ```
  - The order is important
  - Can be done better using the **case** statement

```
   update instructor
       set salary = case
                          when salary <= 100000 then salary * 1.05
                          else salary * 1.03
                         end
```



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

advisor: s_id, i_id

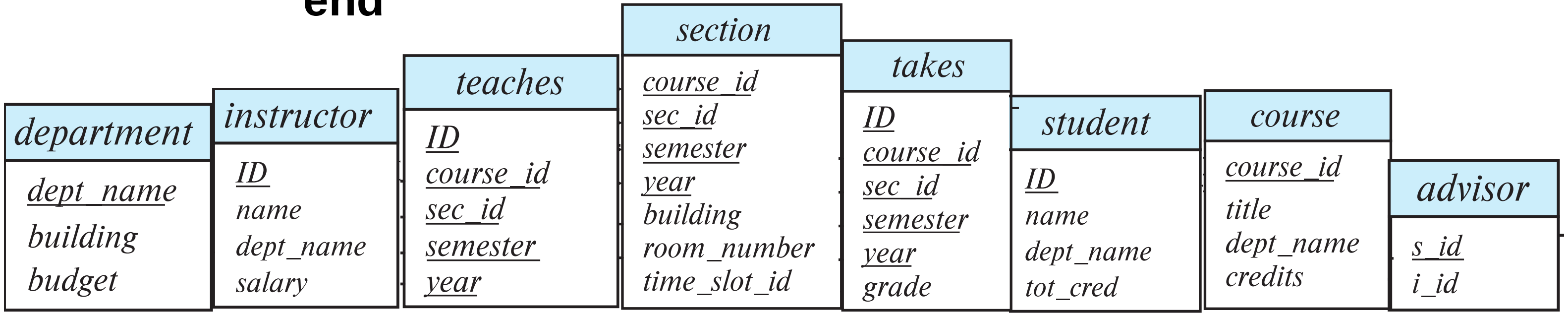
Updates with Scalar Subqueries

- Recompute and update tot_creds value for all students

```
update student S
set tot_cred = (select sum(credits)
                from takes, course
                where takes.course_id = course.course_id  and
                      S.ID= takes.ID.and
                      takes.grade <> 'F' and
                      takes.grade is not null);
```

- Sets *tot_creds* to null for students who have not taken any course
  - Instead of **sum**(*credits*), use:

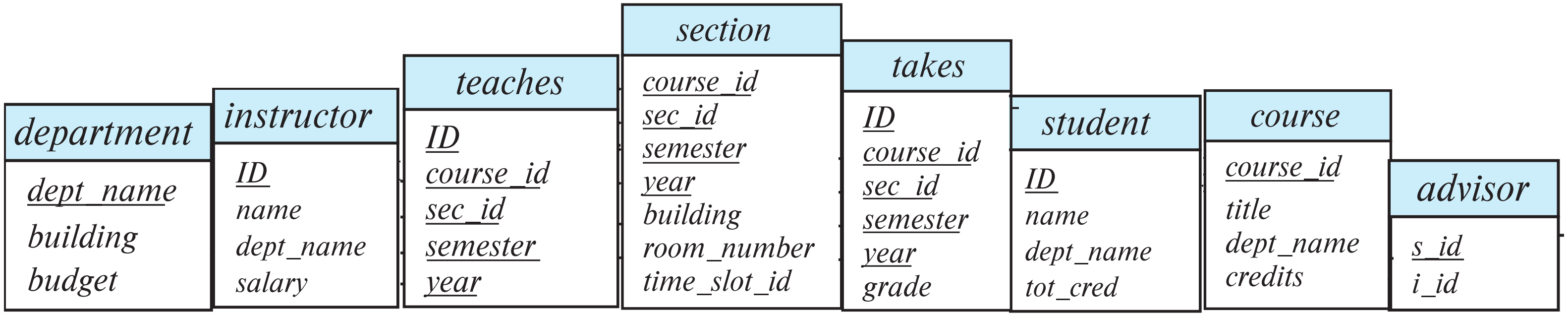
```
case
   when sum(credits) is not null then sum(credits)
   else 0
end
```



department: dept_name, building, budget
instructor: ID, name, dept_name, salary
teaches: ID, course_id, sec_id, semester, year
section: course_id, sec_id, semester, year, building, room_number, time_slot_id
takes: ID, course_id, sec_id, semester, year, grade
student: ID, name, dept_name, tot_cred
course: course_id, title, dept_name, credits
advisor: s_id, i_id

Join conditions

- List the names of students instructors along with the titles of courses that they have taken
  - Natural Join with **Using** Clause
  - Join condition with **on** condition



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

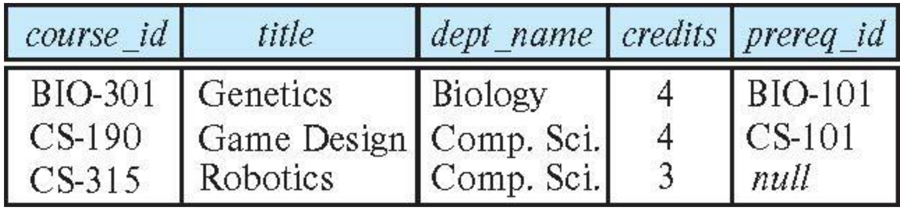
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

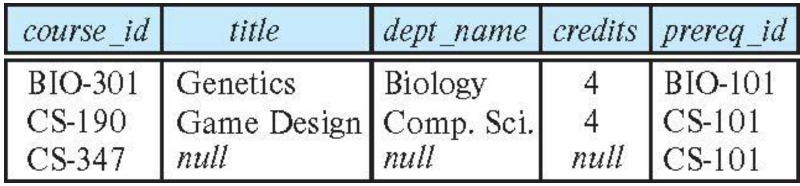
advisor
s_id
i_id

Join types

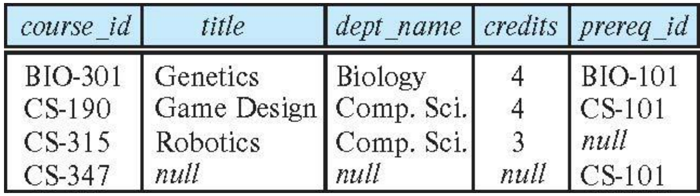
- Three forms of outer join:
  - Natural left outer join *course* ⟕ *prereq*
  - Natural right outer join *course* ⟖ *prereq*
  - Natural full outer join *course* ⟗ *prereq*



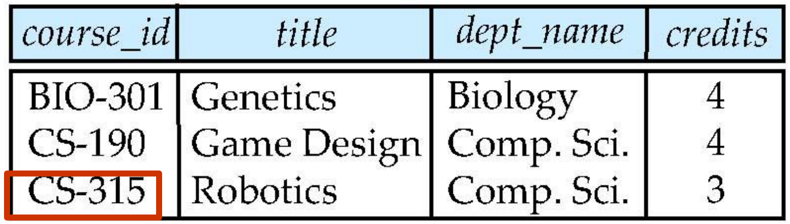
course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*



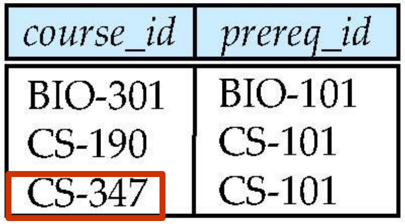
course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101



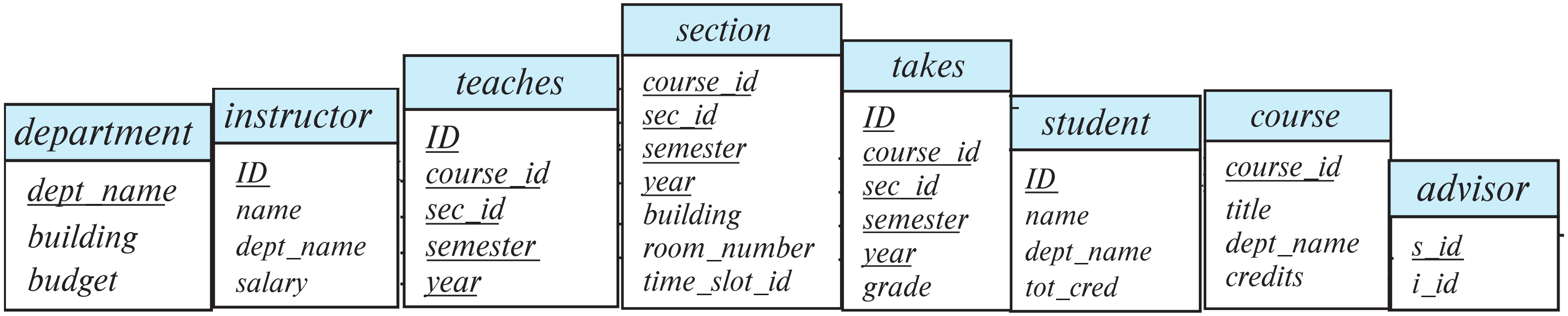
course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101



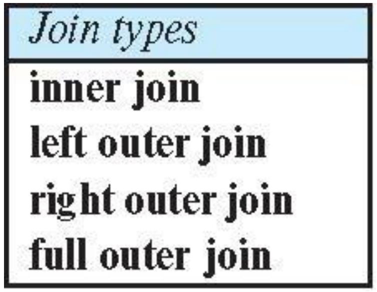
course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3



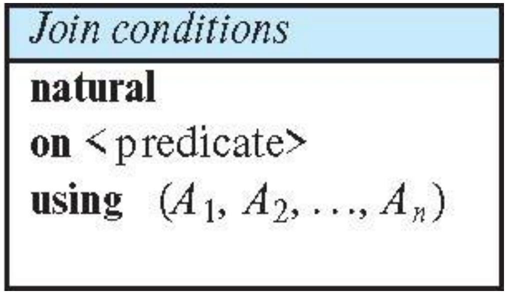
course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101



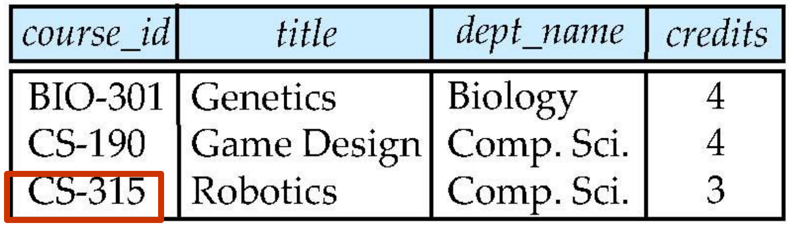
Join



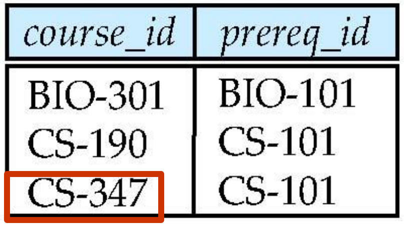
Join types
inner join
left outer join
right outer join
full outer join



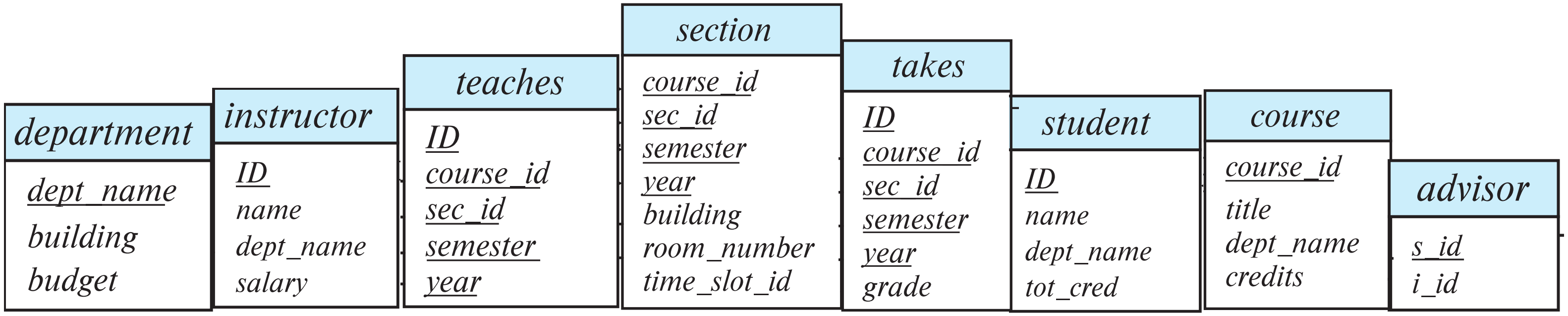
Join conditions
natural
on < predicate>
using $(A_1, A_2, \ldots, A_n)$



course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3



course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

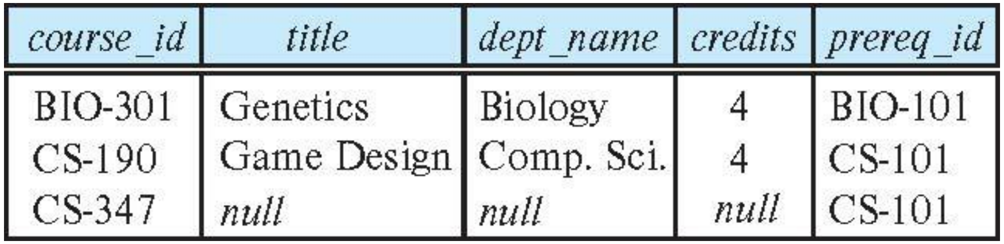
student: *ID*, *name*, *dept_name*, *tot_cred*

course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

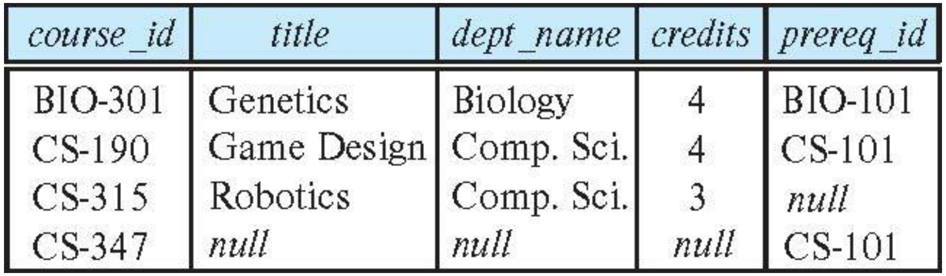
Joined Relations – Examples

- *course* **natural right outer join** *prereq*

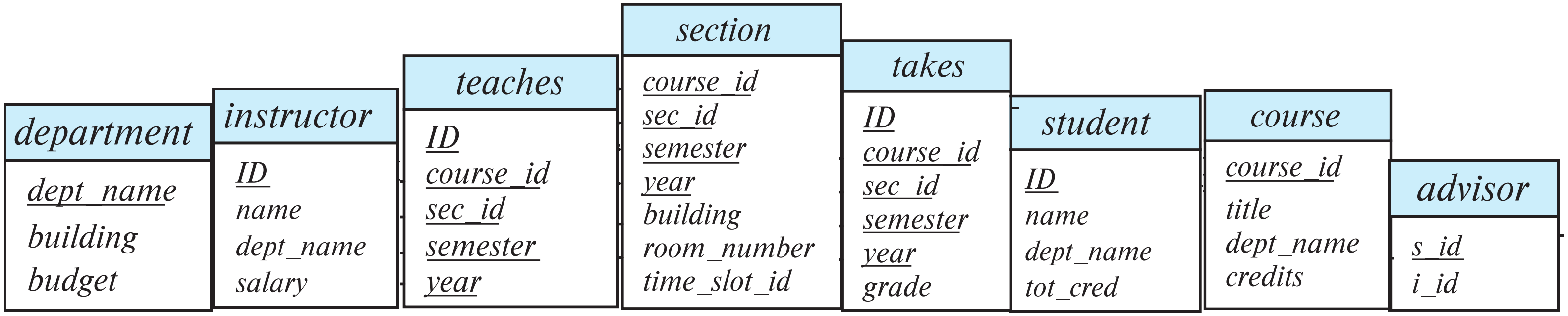


course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

- *course* **full outer join** *prereq* **using** (*course_id*)



course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

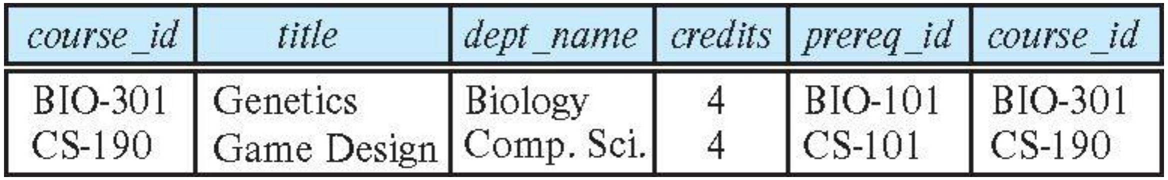
student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

advisor
s_id
i_id

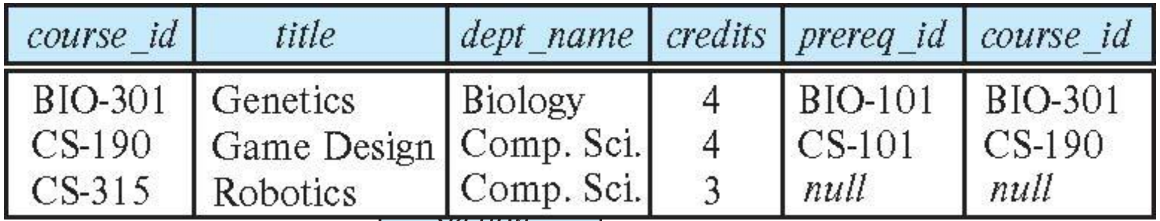
Joined Relations – Examples

- *course* **inner join** *prereq* **on**
  course.course_id = prereq.course_id

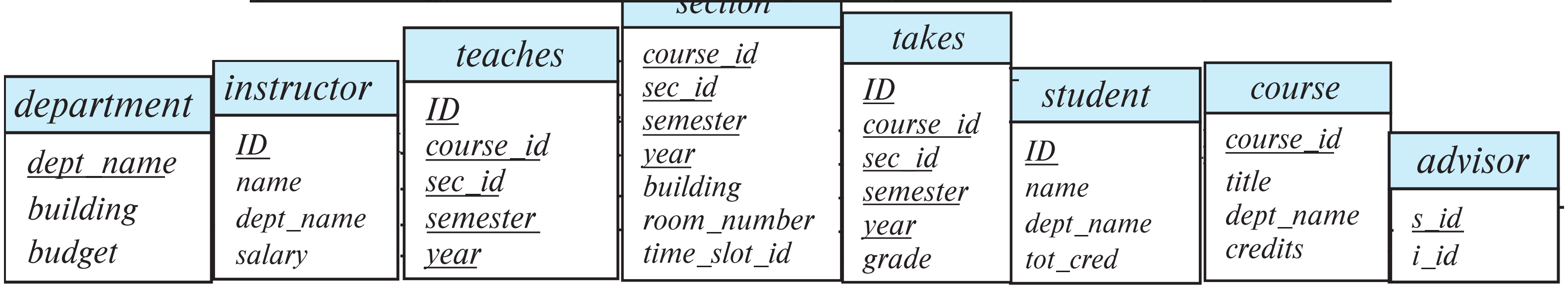


course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190

- What is the difference between the above, and a natural join?
- *course* **left outer join** *prereq* **on**
  course.course_id = prereq.course_id



course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190
CS-315	Robotics	Comp. Sci.	3	*null*	*null*



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

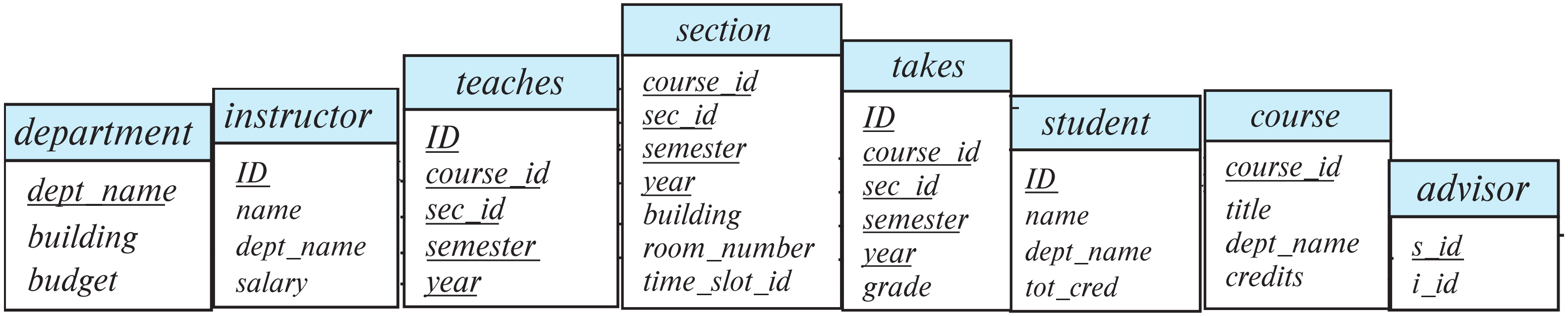
Joined Relations – Examples

- *course* **natural right outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

- *course* **full outer join** *prereq* **using** (*course_id*)

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101



department
dept_name
building
budget

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

section
course_id
sec_id
semester
year
building
room_number
time_slot_id

takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

course
course_id
title
dept_name
credits

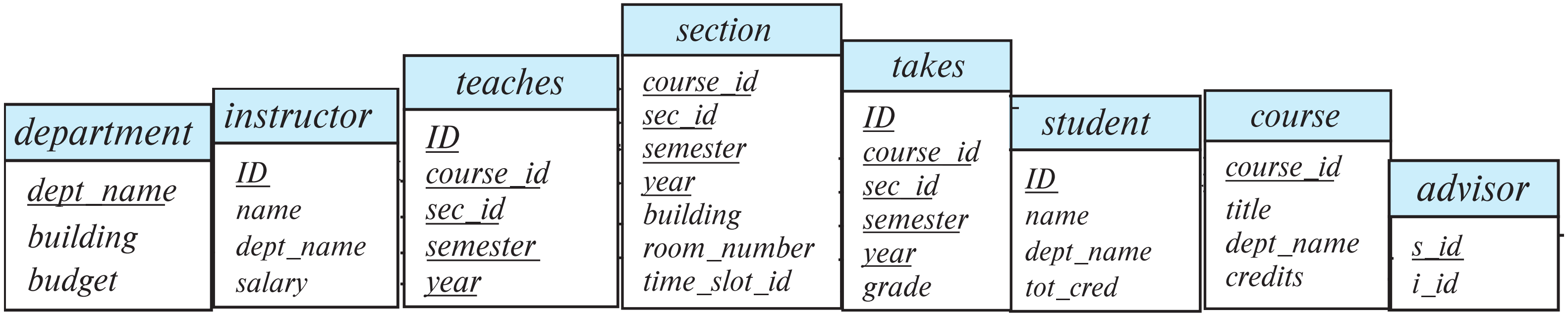
advisor
s_id
i_id

View

- A **view** provides a mechanism to hide certain data from the view of certain users.
  - Consider a person who needs to know an instructors name and department, but not the salary. This person should see a relation described, in SQL, by

    select *ID*, *name*, *dept_name*
    from *instructor*
- **create view** *v* **as** < query expression >

1. A view of instructors without their salary
2. Find all instructors in the Biology department
3. Create a view of department salary totals



department: *dept_name*, *building*, *budget*

instructor: *ID*, *name*, *dept_name*, *salary*

teaches: *ID*, *course_id*, *sec_id*, *semester*, *year*

section: *course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*

takes: *ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*

student: *ID*, *name*, *dept_name*, *tot_cred*

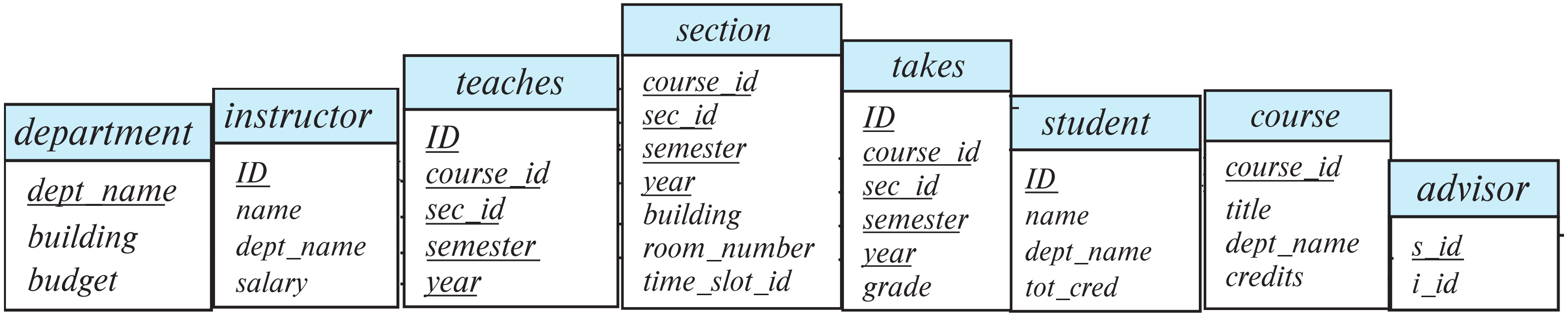
course: *course_id*, *title*, *dept_name*, *credits*

advisor: *s_id*, *i_id*

Define Views using other Views

-
  ```
  create view physics_fall_2017 as
     select course.course_id, sec_id, building, room_number
     from course, section
     where course.course_id = section.course_id
              and course.dept_name = 'Physics'
              and section.semester = 'Fall'
              and section.year = '2017’;
  ```

-
  ```
  create view physics_fall_2017_watson as
      select course_id, room_number
      from physics_fall_2017
      where building= 'Watson';
  ```



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

course: course_id, title, dept_name, credits

advisor: s_id, i_id

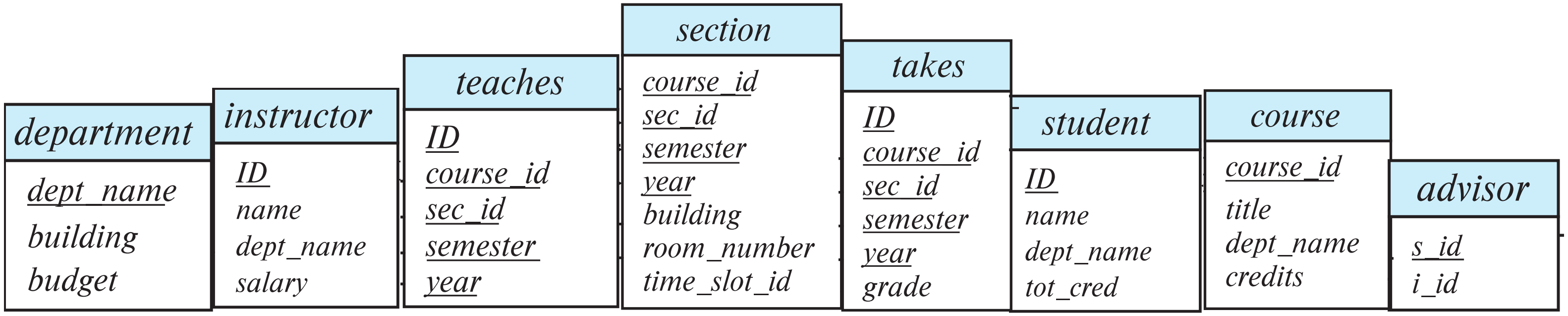
View Expansion

- Expand the view :

```
create view physics_fall_2017_watson  as
   select course_id, room_number
   from physics_fall_2017
   where building= 'Watson'
```

- To:

```
create view physics_fall_2017_watson as
   select course_id, room_number
   from (select course.course_id, building, room_number
        from course, section
        where course.course_id = section.course_id
            and course.dept_name = 'Physics'
            and section.semester = 'Fall'
            and section.year = '2017')
   where building= 'Watson';
```



department: dept_name, building, budget

instructor: ID, name, dept_name, salary

teaches: ID, course_id, sec_id, semester, year

section: course_id, sec_id, semester, year, building, room_number, time_slot_id

takes: ID, course_id, sec_id, semester, year, grade

student: ID, name, dept_name, tot_cred

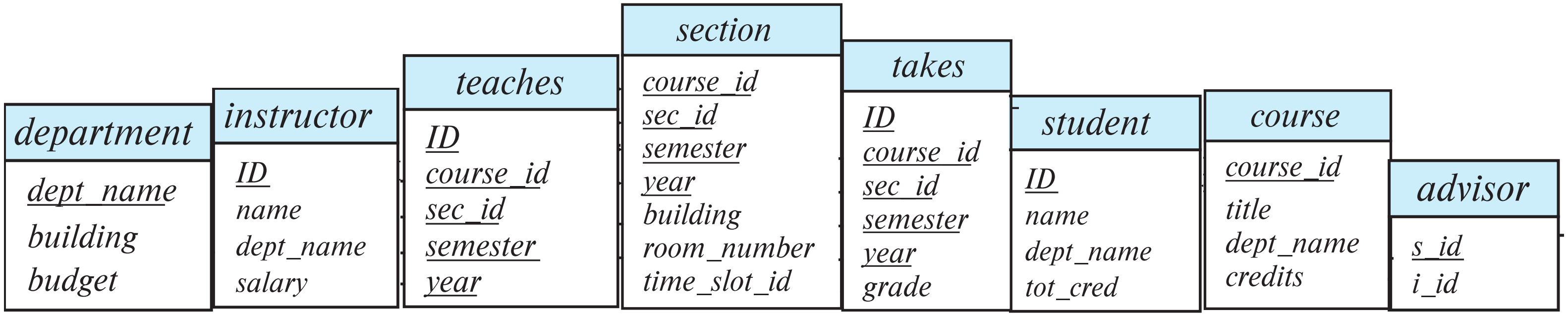
course: course_id, title, dept_name, credits

advisor: s_id, i_id

Views

- Materialized Views
- Update Views
  - Add a new tuple to *faculty* view which we defined earlier

    insert into *faculty* **values** ('30765', 'Green', 'Music');



Some Updates Cannot be Translated Uniquely

-
  ```
  create view instructor_info as
        select ID, name, building
         from instructor, department
         where instructor.dept_name = department.dept_name;
  ```

-
  ```
  insert into instructor_info values ('69987', 'White', 'Taylor');
  ```
- Issues
  - Which department, if multiple departments in Taylor?
  - What if no department is in Taylor?

- Most SQL implementations allow updates only on simple views
  - The **from** clause has only one database relation.
  - The **select** clause contains only attribute names of the relation, and does not have any expressions, aggregates, or **distinct** specification.
  - Any attribute not listed in the **select** clause can be set to null
  - The query does not have a **group** by or **having** clause.

FIN

Any questions?