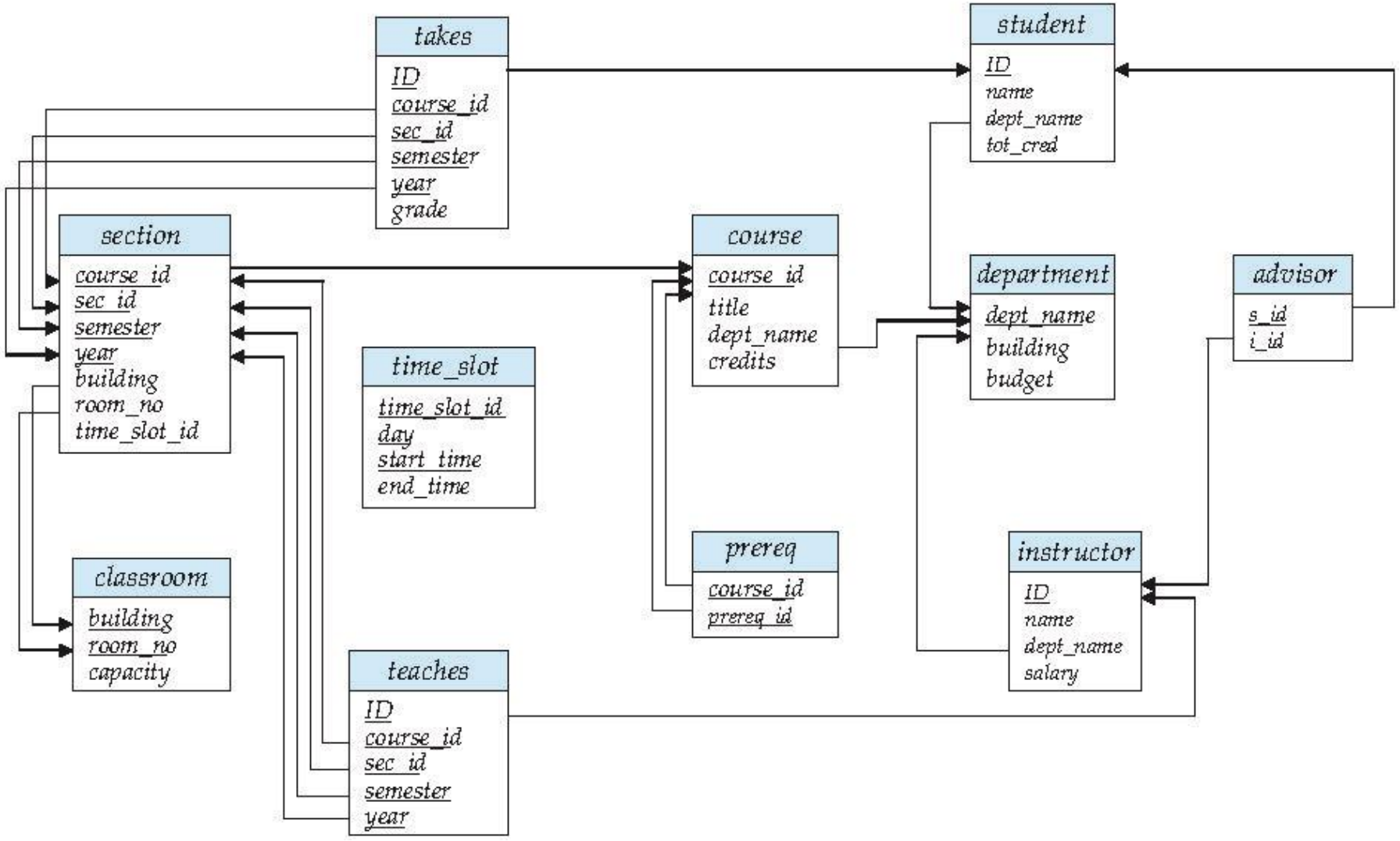
Chapter 3: Introduction to SQL

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.

Set Operations



Set Operations (Cont.)

- Find the salaries of all instructors that are less than the largest salary.
- Find all the salaries of all instructors
- Find the largest salary of all instructors.

Set Operations (Cont.)

- Find the salaries of all instructors that are less than the largest salary.
  - **select distinct** *T.salary*
    from *instructor* **as** *T, instructor* **as** *S*
    where *T.salary* < *S.salary*
- Find all the salaries of all instructors
  - **select distinct** *salary*
    from *instructor*
- Find the largest salary of all instructors.
  - (**select** "second query")
    except
    (**select** "first query")

Example

Instructor
Salary
5
8
1

T
Salary
5
8
1

S
Salary
5
8
1

T	S
Salary	Salary
5	5
5	8
5	1
8	5
8	8
8	1
1	5
1	8
1	1

Temp	
T	S
Salary	Salary
5	8
1	5
1	8

Temp	
T	S
Salary	Salary
5	8
1	5
1	8

Instructor
Salary
5
8
1

—

Temp
T.Salary
5
1

=

Result
Salary
8

Set Operations (Cont.)

- Set operations **union**, **intersect**, and **except**
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates use the corresponding multiset versions **union all**, **intersect all** and **except all**.
- Suppose a tuple occurs *m* times in *r* and *n* times in *s*, then, it occurs:
  - *m* + *n* times in *r* **union all** *s*
  - min(*m*,*n*) times in *r* **intersect all** *s*
  - max(0, *m* – *n*) times in *r* **except all** *s*

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of the attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null*
  - Example: 5 + *null* returns null
- The predicate **is null** can be used to check for null values.
  - Example: Find all instructors whose salary is null.

    select *name*
    from *instructor*
    where *salary* **is null**

Null Values and Three Valued Logic

- Three values – *true*, *false*, *unknown*
- Any comparison with *null* returns *unknown*
  - Example*: 5 < null or null <> null or null = null*
- Three-valued logic using the value *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: *(true* **and** *unknown)* = *unknown,*
    (false **and** *unknown) = false,*
    (unknown **and** *unknown) = unknown*
  - NOT*:* *(***not** *unknown) = unknown*
  - "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*
- Result of **where** clause predicate is treated as *false* if it evaluates to *unknown*

Where predicate with null

- Example: Find all instructors whose salary is greater than 5000.

  select *name*
  from *instructor*
  where *salary > 5000*

Suppose a record has salary = null.

What value will the predicate return for that record ?

Will it be treated as true or false ?

Where predicate with null

- Example: Find all instructors whose salary is greater than 5000.

  select *name*
  from *instructor*
  where *salary > 500*

Suppose a record has salary = null.

What value will the predicate return for that record ?

Will it be treated as true or false ?

As false

null: select distinct, set operations

- Eliminating duplicate tuples

  select distinct *name*
  from *instructor*
  where *salary > 500*

Suppose two records have name as null

Will both the records be retained ?

null: select distinct, set operations

- Eliminating duplicate tuples

  select distinct *name*
  from *instructor*
  where *salary > 500*

Suppose two records have name as null

Will both the records be retained ?

No, only one will be retained – treatment different from predicates

- Same treatment for set operations

Aggregate Functions

- These functions operate on the (collection) 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

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

Aggregate Functions (Cont.)

- Find the average salary of instructors in the Computer Science department
  - **select avg** (*salary*)
    from *instructor*
    where *dept_name*= ’Comp. Sci.’;
- Find the total number of instructors who teach a course in the Spring 2010 semester
  - **select count** (**distinct** *ID*)
    from *teaches*
    where *semester* = ’Spring’ **and** *year* = 2010;
- Find the number of tuples in the *course* relation
  - **select count** (*)
    from *course*;

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
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

Aggregate Functions – Group By

- Find the average salary of instructors in each department

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

Aggregate Functions – Group By

- Find the average salary of instructors in each department
  - **select** *dept_name*, **avg** (*salary*) **as** *avg_salary*
    from *instructor*
    group by *dept_name*;

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

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregation (Cont.)

- Attributes in **select** clause outside of aggregate functions must appear in **group by** list
  - /* erroneous query */
    select *dept_name*, *ID*, **avg** (*salary*)
    from *instructor*
    group by *dept_name*;
  - Which id must be output corresponding to a group (dept_name) is not known

Aggregate Functions – Having Clause

- Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary)
from instructor
group by dept_name
having avg (salary) > 42000;
```

Note: predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

Order of evaluation of aggregate queries

select $A_1, A_2, ..., A_n$	*5*
from $r_1, r_2, ..., r_m$	*1*
where P	*2*
group by $A_1, A_2, ..., A_n$	*3*
having P_2	*4*

1. **from** clause is evaluated first to get a relation
2. If a **where** clause is present, its predicate is applied
3. Tuples are then grouped as per **group by** if it exists
4. **having** clause is applied to each group; groups that do not satisfy the clause are removed
5. **select** clause uses tuples of the remaining groups to generate the result

Null Values and Aggregates

- Total all salaries

  select sum (*salary*)
  from *instructor*

  - Above statement ignores null amounts
  - Result is *null* if all amounts are null
- All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes
- What if collection has only null values?
  - count returns 0
  - all other aggregates return null