Database Management Systems (CSN-351)
SQL (Contd.)

BTech 3rd Year (CS) + Minor + Audit

Instructor: **R**anita **B**iswas
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand - 247 667, India

Rename Operation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

Rename Operation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

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

Rename Operation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

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

select *T.name*, *S.course_id*
from *instructor* **as** *T*, *teaches* **as** *S*
where *T.ID*= *S.ID*;

Rename Operation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

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

select *T.name*, *S.course_id*
from *instructor* **as** *T*, *teaches* **as** *S*
where *T.ID*= *S.ID*;

- Find the names of all instructors whose salary is greater than at least one instructor in the Biology department.

Rename Operation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

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

select *T.name*, *S.course_id*
from *instructor* **as** *T*, *teaches* **as** *S*
where *T.ID*= *S.ID*;

- Find the names of all instructors whose salary is greater than at least one instructor in the Biology department.

select distinct *T.name*
from *instructor* **as** *T*, *instructor* **as** *S*
where *T.salary* > *S.salary* **and** *S.dept_name* = 'Biology';

String Operations

- Percent (%): The % character matches any substring.
- Underscore (_): The _ character matches any character.

String Operations

- Percent (%): The % character matches any substring.
- Underscore (_): The _ character matches any character.
- 'Intro%' matches any string beginning with "Intro".
- '%Comp%' matches any string containing "Comp" as a substring.
- '___'matches any string of exactly three characters.
- '___%' matches any string of at least three characters.

String Operations (contd.)

department(*dept_name*, *building*, *budget*)

- Find the names of all departments whose building name includes the substring 'Watson'.

String Operations (contd.)

department(*dept_name*, *building*, *budget*)

- Find the names of all departments whose building name includes the substring 'Watson'.

```
select dept_name
from department
where building like '%Watson%';
```

Attribute Specification in Select Clause

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

select *instructor.**
from *instructor*, *teaches*
where *instructor.ID*= *teaches.ID*;

Ordering the Display of Tuples

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- List in alphabetic order all instructors in the Physics department.

Ordering the Display of Tuples

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- List in alphabetic order all instructors in the Physics department.

```
select name
from instructor
where dept_name = 'Physics'
order by name;
```

Ordering the Display of Tuples

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- List in alphabetic order all instructors in the Physics department.

```
select name
from instructor
where dept_name = 'Physics'
order by name;
```

```
select *
from instructor
order by salary desc, name asc;
```

Ordering the Display of Tuples

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- List in alphabetic order all instructors in the Physics department.

```
select name
from instructor
where dept_name = 'Physics'
order by name;
```

```
select *
from instructor
order by salary desc, name asc;
```

- List the entire instructor relation in descending order of salary. If several instructors have the same salary, order them in ascending order by name.

Where Clause Predicates

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the names of instructors with salary amounts between 90,000 and 100,000

Where Clause Predicates

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the names of instructors with salary amounts between 90,000 and 100,000

```
select name
from instructor
where salary <= 100000 and salary >= 90000;
```

Where Clause Predicates

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the names of instructors with salary amounts between 90,000 and 100,000

```
select name
from instructor
where salary <= 100000 and salary >= 90000;
```

```
select name
from instructor
where salary between 90000 and 100000;
```

Where Clause Predicates (contd.)

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

- Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course.

Where Clause Predicates (contd.)

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

- Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course.

```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID and dept_name = 'Biology';
```

Where Clause Predicates (contd.)

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

teaches(<u>ID</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)

- Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course.

```
select name, course_id
from instructor, teaches
where instructor.ID= teaches.ID and dept_name = 'Biology';
```

```
select name, course_id
from instructor, teaches
where (instructor.ID, dept_name) = (teaches.ID, 'Biology');
```

Set Operations

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 semester.

Set Operations

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 semester.

```
select course_id
from section
where semester = 'Fall' and year= 2009;
```

Set Operations

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 semester.

```
select course_id
from section
where semester = 'Fall' and year= 2009;
```

- The set of all courses taught in the Spring 2010 semester.

Set Operations

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 semester.

```
select course_id
from section
where semester = 'Fall' and year= 2009;
```

- The set of all courses taught in the Spring 2010 semester.

```
select course_id
from section
where semester = 'Spring' and year= 2010;
```

Union Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught either in Fall 2009 or in Spring 2010, or both.

Union Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught either in Fall 2009 or in Spring 2010, or both.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
union
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Union Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught either in Fall 2009 or in Spring 2010, or both.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
union
(select course_id
 from section
 where semester = 'Spring' and year= 2010);

(select course_id
 from section
 where semester = 'Fall' and year= 2009)
union all
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Intersect Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 as well as in Spring 2010.

Intersect Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 as well as in Spring 2010.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
intersect
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Intersect Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- The set of all courses taught in the Fall 2009 as well as in Spring 2010.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
intersect
(select course_id
 from section
 where semester = 'Spring' and year= 2010);

(select course_id
 from section
 where semester = 'Fall' and year= 2009)
intersect all
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Except Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- All courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

Except Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- All courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
except
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Except Operation

section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

- All courses taught in the Fall 2009 semester but not in the Spring 2010 semester.

```
(select course_id
 from section
 where semester = 'Fall' and year= 2009)
except
(select course_id
 from section
 where semester = 'Spring' and year= 2010);

(select course_id
 from section
 where semester = 'Fall' and year= 2009)
except all
(select course_id
 from section
 where semester = 'Spring' and year= 2010);
```

Duplicate Retention

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

- *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*

Null Values

- *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 (contd.)

- Any comparison with *null* returns *unknown*.
  Example: $5 < null$ or $null <> null$ or $null = null$
- Three-valued logic using the truth 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*.

Aggregate Functions

- Average: **avg**
- Minimum: **min**
- Maximum: **max**
- Total: **sum**
- Count: **count**

Basic Aggregation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the average salary of instructors in the Computer Science department.

Basic Aggregation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the average salary of instructors in the Computer Science department.

```
select avg (salary)
from instructor
where dept_name= 'Comp. Sci.';
```

Basic Aggregation

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the average salary of instructors in the Computer Science department.

```
select avg (salary)
from instructor
where dept_name= 'Comp. Sci.';
```

```
select avg (salary) as avg_salary
from instructor
where dept_name= 'Comp. Sci.';
```

Basic Aggregation (contd.)

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the total number of instructors who teach a course in the Spring 2010 semester.

Basic Aggregation (contd.)

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

- 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;
```

Basic Aggregation (contd.)

instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

- 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.

Basic Aggregation (contd.)

instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

- 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;
```

Aggregation with Grouping

instructor(<u>ID</u>, *name*, *dept_name*, *salary*)

- Find the average salary in each department.

Aggregation with Grouping

instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)

- Find the average salary in each department.

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

Aggregation with Grouping

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

- Find the average salary in each department.

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

- Find the number of instructors in each department who teach a course in the Spring 2010 semester.

Aggregation with Grouping

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

- Find the average salary in each department.

```
select dept_name, avg (salary) as avg_salary
from instructor
group by dept_name;
```

- Find the number of instructors in each department who teach a course in the Spring 2010 semester.

```
select dept_name, count (distinct ID) as instr_count
from instructor natural join teaches
where semester = 'Spring' and year = 2010
group by dept_name;
```