UCS310
Database Management System

Introduction to SQL : Nested Subqueries

Lecture-09

Date: 30 Jan 2023

Dr. Sumit Sharma
Assistant Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology, Patiala

Recap

- Basic Operations:
  - Select
  - Insert, Delete, Alter, Drop
  - Cartesian Product and Natural Join,
  - Rename
- String Operations: like
- Order By: asc, desc
- Set Operations: union, intersect, except
- Aggregate Functions: avg, sum, min, max, count,
- Group by, having

Nested Subqueries

- SQL provides a mechanism for the nesting of 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 $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
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 by a subquery that generates a single value

Subqueries in the where Clause

Set Membership

- **in connective** tests for set membership
- Find courses offered in Fall 2022 and in Spring 2023
- Nesting the subquery in the where clause of an outer query

```
select distinct course_id
from section
where semester = 'Fall' and year= 2022 and
        course_id in (select course_id
                      from section
                      where semester = 'Spring' and year= 2023);
```

- Find courses offered in Fall 2022 but not in Spring 2023

```
select distinct course_id
from section
where semester = 'Fall' and year= 2022 and
        course_id  not in (select course_id
                           from section
                           where semester = 'Spring' and year=2023);
```

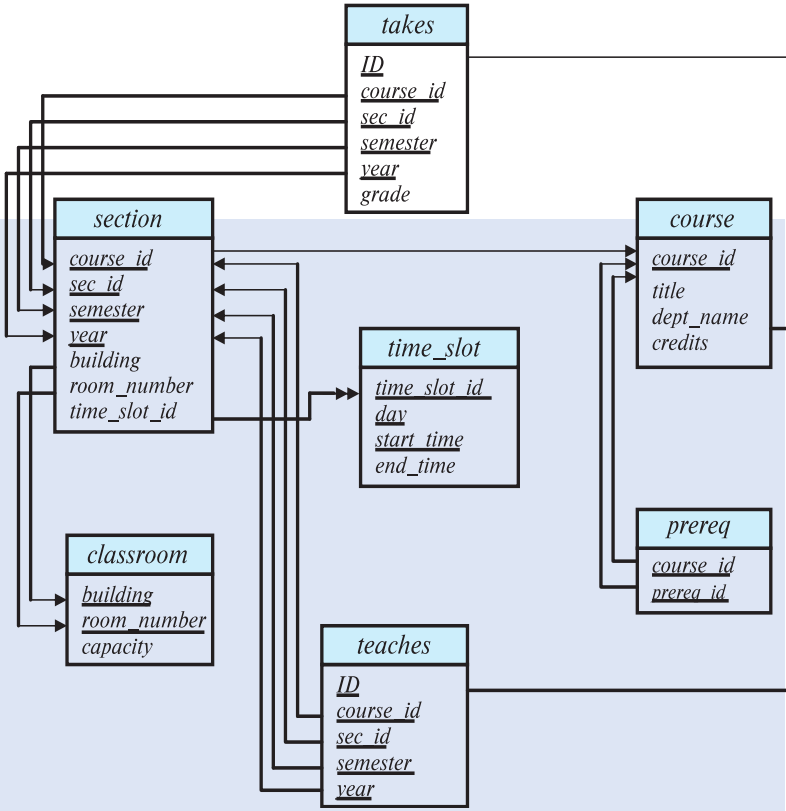
Set Membership

- Name all instructors whose name is neither “Mozart” nor Einstein”

```
select distinct name
 from instructor
 where  name not in ('Mozart', 'Einstein')
```

Set Membership

On arbitrary Relation



- Find the total number of (distinct) students who have taken course sections taught by the instructor with *ID* 10101

```
select count (distinct ID)
from takes
where (course_id, sec_id, semester, year) in
                    (select course_id, sec_id, semester, year
                     from teaches where teaches.ID= 10101);
```

Set Comparison – "some" Clause

- Find names of instructors with salary greater than that of some (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';
```

- Same query using > **some** clause

```
select name
from instructor
where salary > some (select salary
                     from instructor
                     where dept name = 'Biology');
```

> Some ≡ "greater than atleast one"

Definition of "some" Clause

- F <comp> **some** r $\Leftrightarrow$ $\exists\, t \in r$ such that (F <comp> t)
  Where <comp> can be: $<,\ \leq,\ >,\ =,\ \neq,\ <>$

(5 < **some** | 0 / 5 / 6 |) = true (read: 5 < some tuple in the relation)

(5 < **some** | 0 / 5 |) = false

(5 = **some** | 0 / 5 |) = true

(5 ≠ **some** | 0 / 5 |) = true (since 0 ≠ 5)

(= **some**) ≡ **in** (**< >** **some**) ≠ **not in**

However, (≠ **some**) ≠ **not in,**

Set Comparison – “all” Clause

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

```
select name
from instructor
where salary > all (select salary
                    from instructor
                    where dept name = 'Biology');
```

> all ≡ “greater than all”

Definition of “all” Clause

- F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ such that (F <comp> t)

(5 < **all** | 0, 5, 6 |) = false

(5 < **all** | 6, 10 |) = true

(5 = **all** | 4, 5 |) = false

(5 ≠ **all** | 4, 6 |) = true (since 5 ≠ 4 and 5 ≠ 6)

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

Set Comparison – “all” Clause

- Find the department that have the highest average salary

```
select dept_name
from instructor
group by dept_name
having avg(salary)  >= all
                        (select avg(salary)
                        from instructor
                        group by dept_name);
```

Test for Empty Relations

- Existence of a tuple
- The **exists** construct returns the value **true** if the argument subquery is nonempty.
- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

Use of “exists” Clause

- Yet another way of specifying the query “Find all courses taught in both the Fall 2022 semester and in the Spring 2023 semester”

```
select course_id
from section as S
where semester = 'Fall' and year = 2022 and
        exists  (select *
                 from section as T
                 where semester = 'Spring' and year= 2023
                       and S.course_id = T.course_id);
```

- **Correlation name** – variable S in the outer query
- **Correlated subquery** – the inner query

Use of “not exists” Clause

- Find all students who have taken all courses offered in the Biology department.

```
select distinct S.ID, S.name
from student as S
where not exists ( (select course_id
                    from course
                    where dept_name = 'Biology')
                          except
                    (select T.course_id
                     from takes as T
                      where S.ID = T.ID));
```

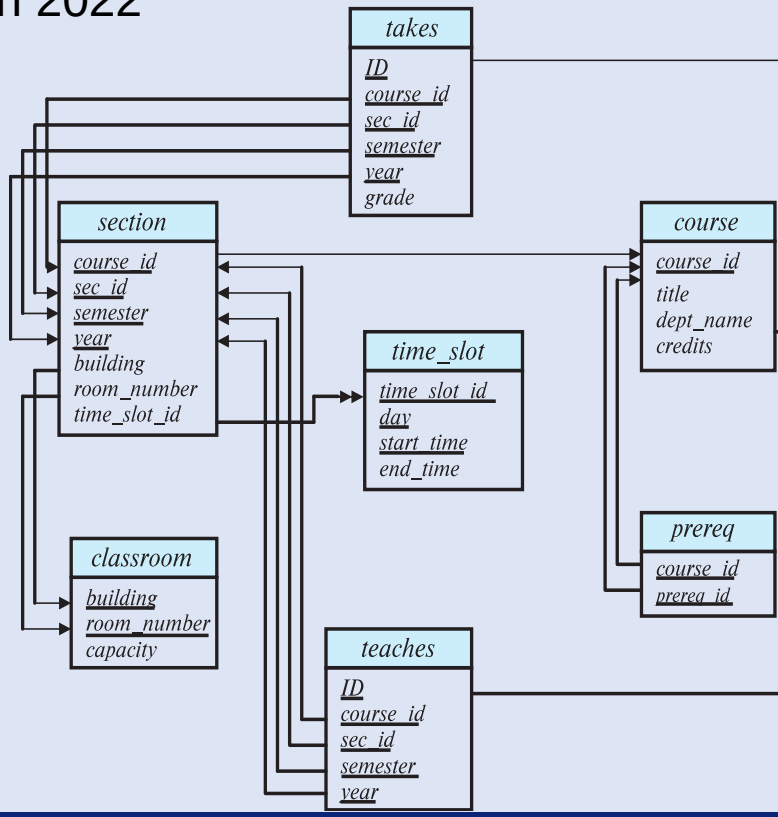
  - First nested query lists all courses offered in Biology
  - Second nested query lists all courses a particular student took

- Note that $X - Y = \emptyset \Leftrightarrow X \subseteq Y$

Test for Absence of Duplicate Tuples

- 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 2022

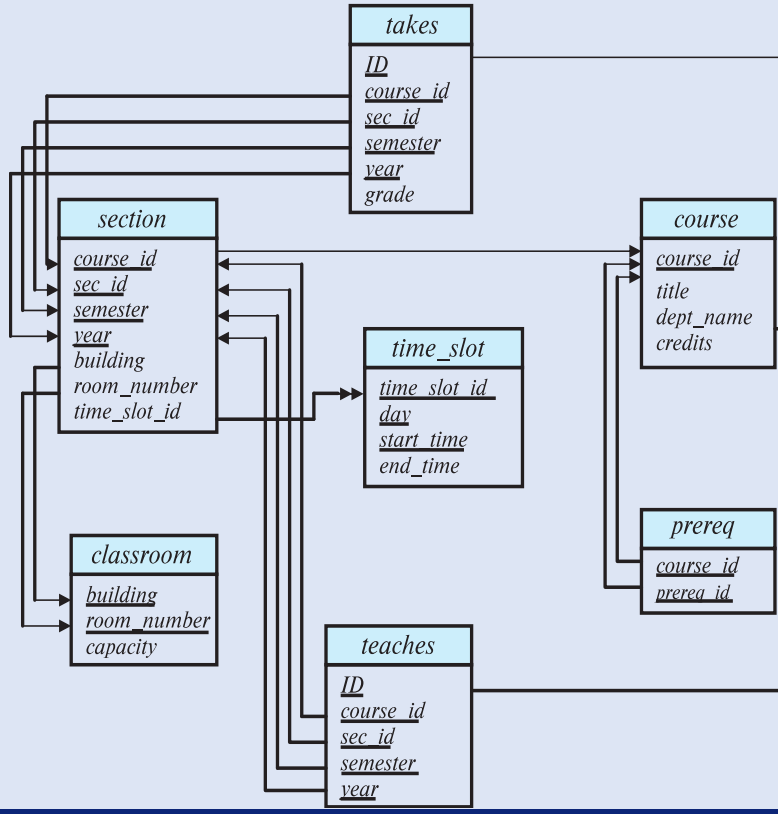
```
select T.course_id
from course as T
where unique ( select R.course_id
                 from section as R
                 where T.course_id = R.course_id
                       and R.year = 2022);
```



Equivalent Existence

- Find all courses that were offered at most once in 2022

```
select T.course_id
from course as T
where 1 <= ( select count (R.course_id)
                from section as R
                where T.course_id = R.course_id
                      and R.year = 2022);
```



Subqueries in the From Clause (new relation)

Subqueries in the From Clause

- SQL allows a subquery expression to be used in the **from** clause
- Find the average instructors' salaries of those departments where the average salary is greater than $42,000."

```
select dept_name, avg_salary
from ( select dept_name, avg (salary) as avg_salary
        from instructor
        group by dept_name)
       where avg_salary > 42000;
```

- Note that we do not need to use the **having** clause
- Another way to write above query

```
select dept_name, avg_salary
from ( select dept_name, avg (salary)
        from instructor
        group by dept_name)
        as dept_avg (dept_name, avg_salary)
       where avg_salary > 42000;
```

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;
```

Complex Queries using With Clause

- Find all departments where the total salary is greater than the average of the total salary at all departments

```
with dept _total (dept_name, value) as
        (select dept_name, sum(salary)
        from instructor
        group by dept_name),
dept_total_avg(value) as
        (select avg (value)
        from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

Thanks!