Introduction to SQL

Dr. Shrutilipi Bhattacharjee, Assistant Professor, Dept. of IT, NIT Karnataka, India

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 as follows:

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

  - **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

Subqueries in the *Where* Clause

- A common use of subqueries is to perform tests:
  - For set membership
  - For set comparison
  - For set cardinality

Set Membership

- Find courses offered in Fall 2017 and in Spring 2018

```
select distinct course_id
from section
where semester = 'Fall' and year = 2017 and
        course_id in (select course_id
                      from section
                      where semester = 'Spring' and year = 2018);
```

- Find courses offered in Fall 2017 but not in Spring 2018

```
select distinct course_id
from section
where semester = 'Fall' and year = 2017 and
        course_id  not in (select course_id
                           from section
                          where semester = 'Spring' and year = 2018);
```

Set Membership

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

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

- 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);
```

- **Note:** Above query can be written in a much simpler manner
- The formulation above is simply to illustrate SQL features

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');

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

  (= **some**) ≡ **in**
  However, ($\neq$ **some**) $\not\equiv$ **not in**
- (5 $\neq$ **some** | 0 | 5 |) = true (since 0 $\neq$ 5)

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');
```

Definition of *All* Clause

- F <comp> **all** r $\Leftrightarrow \forall\ t \in r$ such that (F <comp> t)
  where <comp> can be: $<,\ \leq,\ >,\ =,\ \neq$
- (5 < **all** | 0 / 5 / 6 |) = false (read: 5 < some tuple in the relation)
- (5 < **all** | 6 / 10 |) = true
- (5 = **all** | 4 / 5 |) = false
- (5 ≠ **all** | 4 / 6 |) = true (since 4 ≠ 5 and 6 ≠ 5)

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

Test for Empty Relations

- 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 2017 semester and in the Spring 2018 semester”

```
select course_id
from section as S
where semester = 'Fall' and year = 2017 and
exists  (select * from section as T where semester = 'Spring'
            and year = 2018 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 $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
- **Note:** Cannot write this query using = all and its variants

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 2017

```
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 = 2017);
```

Next Lecture

Introduction to SQL

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6th Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in