SQL – NESTED & CORRELATED SUBQUERIES (PART 2)

CIS-673, LECTURE#08

BY RAJ PATIL

2 CORRELATED SUBQUERY – SELECT CLAUSE

- **Query:** List all departments along with the number of instructors in each department
- **SQL:** select dept_name, (

```
select COUNT(*)
from instructor
WHERE instructor.dept_name = department.dept_name)
from department;
```

//OR

SQL: select dept_name, count(*) from instructor group by dept_name;

3 FROM CLAUSE – NESTED SUBQUERY

- **Query:** Find the average departments-salaries where the average salary is greater than $42,000

- **SQL:**
- SELECT dept_name, AVG(salary) as avg_salary FROM instructor

  GROUP BY dept_name

  HAVING avg_salary > 42000;

- **Nested Subquery:**

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

4 FROM CLAUSE – NESTED SUBQUERY

- **Query:** Find department(s) with the maximum budget

- **SQL:** Select department.dept_name

  from department,

  (select max(budget) as max_value

  from department) temp

  where department.budget = temp.max_value;

5 WITH – CREATES TEMPORARY RELATION

- **Query:** Find department(s) with the maximum budget

- **SQL:** WITH

```
temp(max_value) as
    (select max(budget)
    from department )
select department.dept_name
from department, temp
where department.budget = temp.max_value;
```

6 WITH – EXAMPLE#2

- **Query:** Find all departments where their total salary is greater than the average salary of all departments
- **SQL:**

```
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;
```

7 CORRELATED SUBQUERY – FROM CLAUSE

- LATERAL - Allows writing correlated subquery in FROM clause

- **Query:** print name, salary of each instructor along with the average salary in their dept.

- **SQL:**

```
select T1.name, T1.salary, avg_salary
from instructor T1, lateral (
select avg(salary) as avg_salary
from instructor T2
where T2.dept_name = T1.dept_name );
```

8 NESTED QUERY IN DELETE

- **Query:** Delete all instructors who work in the departments located in the Watson building.
- **SQL:**
```
delete from instructor
        where dept_name in (
        select dept_name from department where building = 'Watson');
```

- **Query:** Delete all instructors whose salary is less than the average salary of instructors.
- **SQL:**
```
delete from instructor
        where salary < (
                select avg (salary)
                from instructor);
```

CASE…(WHEN, THEN, ELSE)…END EXAMPLE

- **Query:** Give a 5% salary raise to instructors whose salary is less than average
- **SQL:** update instructor set salary = salary * 1.05

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

- **Query:** For instructors, if the salary is less than 50000 then give a 5% salary raise, else if it is less than 100000 give 3% raise, else give 1% raise.
- **SQL:** update instructor set salary = case when salary <= 50000 then salary * 1.05

  when salary <= 100000 then salary * 1.03

  else salary * 1.01 END;

11 CORRELATED QUERY IN UPDATE

Query: Recompute and update tot_creds value for all students

```
update student  as S
set tot_cred =
        (select sum(credits)
                from takes, course
                where       S.student_id= takes.student_id and
                             takes.course_id = course.course_id and
                            takes.grade != 'F' AND takes.grade is not null
        );
```

12 CORRELATED QUERY IN UPDATE

Query: Recompute and update tot_creds value for all students, and set tot_creds to zero for students who have not taken any courses

```
update student  as S
set tot_cred =
        (select     case
                        when sum(credits) is not null then sum(credits)
                        else 0
                    end
                from takes, course
                where       S.student_id= takes.student_id and
                             takes.course_id = course.course_id and
                            takes.grade != 'F' AND takes.grade is not null
        );
```

13 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**

- Similarly, the predicate **is not null** succeeds (is true) if the value on which it is applied is not null.

NULL VALUES (CONT.)

- SQL treats as **unknown** the result of any comparison involving a null value.
  - Example: *5* < **null** or **null** <> **null** or **null** = **null**

- Boolean operations
  - **and** : (*true* **and** *unknown*) = *unknown*,
    (*false* **and** *unknown*) = *false*,
    (*unknown* **and** *unknown*) = *unknown*
  - **or:** (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
- Result of **where** clause predicate is treated as *false* if it evaluates to *unknown*