CMSC 508
Database Theory

Intermediate SQL (II)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003

- Nested subqueries
  - SQL provides mechanisms for nesting 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:

    select $A_1, A_2, ..., A_n$
    from $r_1, r_2, ..., r_n$
    where $C_1, C_2, ..., C_n$

    - A_i can be replaced be a subquery that generates a **single value**
    - r_i can be replaced by any valid subquery (**temporal table**)
    - C_i can be replaced with an expression of the form:

      B <operation> (subquery)

      where *B* is an attribute and <operation> to be defined later
    - Any **having** predicate can be also replaced with a subquery

- Subqueries in the **where** clause
  - Common uses of subqueries to perform tests for set membership, set comparisons, and set cardinality
  - **Single-register subqueries**: comparison of multiple rows vs a single tuple returned from the subquery

select *last_name* **from** *employees*
where
salary > (**select** *salary* **from** *employees* **where** *employee_id* = 111);

select *last_name* **from** *employees*
where
salary > (**select** *salary* **from** *employees* **where** *employee_id* = 111)
and
job_id = (**select** *job_id* **from** *employees* **where** *employee_id* = 109);

- Subqueries in the **having** clause
  - Common uses of subqueries to perform tests for set membership, set comparisons, and set cardinality
  - **Single-register subqueries**: comparison of multiple rows vs a single tuple returned from the subquery

```
select department_name, MAX(salary)
from employees join departments
on employees.department_id = departments.department_id
group by department_name
having MIN(salary) >
      (select MIN(salary) from employees where department_id = 30);
```

- Subqueries in the **where** clause
  - **Multi-register subqueries**: comparison of multiple rows vs multiple tuples returned from the subquery

```
select last_name, job_title
from employees natural join jobs
where salary <
ANY (select salary from employees where job_id LIKE 'SA%')
and job_title NOT LIKE 'Sales%';

select last_name, job_title
from employees natural join jobs
where salary >
ALL (select AVG(salary) from employees group by department_id);
```

- Subqueries in the **where** clause
  - **Multi-register subqueries**: comparison of multiple rows vs multiple tuples returned from the subquery

```
select last_name, job_title
from employees natural join jobs
where manager_id
IN (select employee_id from employees where department_id = 20);


select distinct department_name
from employees e1 join departments
on e1.department_id = departments.department_id
where exists
      (select * from employees e2
       where e1.department_id = e2.department_id
       and e1.employee_id <> e2.employee_id);
```

- Exercise
  - Select the employees whose salary is bigger than their department's average salary

- Exercise
  - Select the employees whose salary is bigger than their department's average salary


```
select e1.last_name, e1.salary
from employees e1
where e1.salary >
      (select AVG(salary) from employees e2
      where e2.department_id = e1.department_id);
```

- Multi-column subqueries in the **where** clause
  - A subquery may return multiple columns

```
select e.last_name employee, e.department_id,
        m.last_name manager, e.salary
from employees e join employees m
       on e.manager_id = m.employee_id
where (e.manager_id, e.salary)
     IN (select manager_id, salary from
         employees where department_id = 80)
order by department_id;
```

- Multi-column subqueries in the **where** clause

```
select e.last_name employee, e.department_id,
        m.last_name manager, e.salary
from employees e join employees m
        on e.manager_id = m.employee_id
where e.manager_id
      IN (select manager_id from
          employees where department_id = 80)
and e.salary
      IN (select salary from
          employees where department_id = 80)
order by department_id;
```

NOT EQUIVALENT

- NULL values in subqueries
  - NOT IN with a set containing NULL always return FALSE
  - Example: list the employees not supervising any other employees

select *last_name*
from *employees*
where *employee_id* **NOT IN** (**select** *manager_id* **from** *employees*);

vs

select *last_name*
from *employees*
where *employee_id* **NOT IN**
(**select** *manager_id* **from** *employees* **where** *manager_id* **is not null**);

- Subqueries in the **from** clause
  - Commonly employed as **temporary** tables, very helpful!
  - Exercise example:

```
select last_name, job_title, salary, averages.avgDept
from employees natural join jobs ,
      (select department_id, AVG(salary) avgDept
       from employees
       group by department_id) averages
where employees.department_id = averages.department_id
and employees.salary > averages.avgDept;
```

- Subqueries in the **select** clause
  - Scalar subquery is used where a **single** value is expected
  - Example: list all departments along with the number of employees

select *department_name*, (**select** count(*) **from** *employees* **where** *departments.department_id* = *employees.department_id*) **from** *departments*;

vs

select *department_name*, count(*)
from *employees* **join** *departments*
on *departments.department_id* = *employees.department_id*
group by *department_name*;

The join does not show departments with no employees, full join then?

- Modification of the database using subqueries
  - Deleting, inserting, updating tuples in a given relation
  - Some illustrative examples **that will not work on purpose** so that you do not modify the database:

```
delete from instructor
where dept_name in (select dept_name
                    from department
                    where building = 'Watson');

update student S
set tot_cred = (select sum(credits)
                from takes, course
                where takes.course_id = course.course_id and S.ID= takes.ID
                and takes.grade <> 'F' and takes.grade is not null);
```

- Exercises

Write a query to show the name of the employees, the department name, and the location (city) for all employees working for a department whose location is in UK.

Write a query to show the department number, department name, the average salary of all employees working for that department, and the number of employees of the department. Show only the departments having at least two employees.

CMSC 508
Database Theory

Intermediate SQL (II)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003