Handout 8  
Multi Table Queries. Joins and Nested Subqueries.

Joins

Many queries need to access data from more than one table. To combine (or join) two or more tables in a query:

**equi-join** - connects tuples based on the equality of the attributes of rows in two tables. Most often it involves the equality between a **foreign key** of one table to a **primary key** of another.

**Example:** Display the names of all employees with their department.

```
SELECT emp.fname || ' ' || emp.lname
FROM EMPLOYEE, DEPT
WHERE emp.dept = dept_nbr;
```

*emp.dept* is the foreign key in the **EMPLOYEE** table that references the *dept_nbr* primary key of the **DEPT** table.

In the query above *emp.dept = dept_nbr* is the *join condition*, it "ties together" the rows of the two tables.

Note that the same query can be written in the following way, where each column name is prefixed with the name of the table from which the column is taken:

```
SELECT EMPLOYEE.emp.fname || ' ' || EMPLOYEE.emp.lname
FROM EMPLOYEE, DEPT
WHERE EMPLOYEE.emp.dept = DEPT.dept_nbr;
```

Such prefixing is **necessary if two tables have columns with the same name** (textbook examples). Access always uses the prefixed notation.

When the WHERE clause is missing or does not contain a join condition, the result is a **cross join (a.k.a. Cartesian product)** of the two tables: a set of rows that includes every possible row combination from the two tables. In other words, each row from one table is conjoined with each row of the other.

**Example:** what will be produced by the following query? How many rows will be selected given that the **EMPLOYEE** table has 16 rows the **DEPT** table has 7 rows?

```
SELECT emp.fname || ' ' || emp.lname, dept_name
FROM EMPLOYEE, DEPT
;
```

Note that the WHERE clause may contain conditions other than the join condition.

**Example:** Display the names of those employees that work in building 'G'

```
SELECT emp.fname || ' ' || emp.lname
FROM EMPLOYEE, DEPT
WHERE emp.dept = dept.nbr and dept.building = 'G';

More than two tables can be joined together.

**Example:** Display the names of those customer representatives who work with customers from California, and the building in which they work.

```
SELECT emp.fname || ' ' || emp.lname, dept.building
FROM EMPLOYEE, DEPT, CUSTOMER
WHERE cust.state = 'CA' and cust.rep = emp.nbr and emp.dept = dept.nbr ;
```

Here is a more complex example that demonstrates multitable grouping: **Example:** Display the number and name of each customer and the total amount of last year’s sales to that customer.

```
SELECT sales_cust_nbr, cust.name, SUM(sales.last_year)
FROM SALES, CUSTOMER
WHERE cust_nbr = sales_cust_nbr
GROUP BY sales_cust_nbr, cust.name;
```

**Practice problems:**

1. What result is generated by the following queries?

```
select Emp_nbr, Emp_lname, Emp_fname
from EMPLOYEE, CUSTOMER, SALES
where Emp_nbr = Cust_rep and
  Cust_nbr = Sales_cust_nbr and
  Sales_prod_nbr = 60;

select Cust_name, Cust_phone, sum(Sales_last_year)
from CUSTOMER, SALES, PRODUCT
where Cust_nbr = Sales_cust_nbr and Sales_prod_nbr = Prod_nbr and
  Prod_prodline between 10 and 19
group by Cust_name, Cust_phone
having count(Sales_last_year) > 3
order by count(Sales_last_year) desc;
```

2. Print department name, department manager name and department phone number for all departments.

3. Display the customer number and phone of those customers who bought products from product lines 21, 22 and 23.
4. For each customer representative, display the name and the total number of their customers (HINT: requires grouping).

5. For each customer representative who has more than 1 customer, display the name and the total number of their customers (HINT: requires grouping).

A table can be joined with itself (sometimes necessary). Then, the FROM clause must list the same table twice, giving each table instance a separate name, so that two different rows in the same table could be referenced:

**Example:** Display all pairs of customers that have the same customer representative:

```
SELECT C1.cust_name, C2.cust_name
FROM CUSTOMER C1, CUSTOMER C2
WHERE C1.cust_rep = C2.cust_rep;
```

**Practice problem:** What would be the result if the CUSTOMER table was not aliased, i.e.

```
SELECT cust_name, cust_name
FROM CUSTOMER
WHERE cust_rep = cust_rep;
```

Here’s another example where aliasing is necessary: **Example:** Display all employees who are married to another employee, together with the name of their spouse:

```
SELECT E1.emp_fname || E1.emp_lname, E2.emp_fname || E2.emp_lname
FROM EMPLOYEE E1, EMPLOYEE E2, MARRIAGE M
WHERE M.emp1_nbr = E1.emp_nbr and M.emp2_nbr = E2.emp_nbr;
```

**Practice problem:**

- How to eliminate duplicate pairs in the previous query?
- Display all pairs of employees that have the same number of dependents.

**Nested Subqueries**

Some SELECT statements include nested SELECT subqueries in the WHERE, or HAVING clause. The result of the subquery is either

- a number, generated by an aggregate fn. (COUNT, SUM, AVE, MIN, MAX).
  
  **Example:** Display the names of those employees that have the highest number of dependents. Hint: an employee has the highest number of dependents if his number of dependents equals the maximum number of dependents across the entire company.
SELECT E1.emp_fname, E1.emp_lname
FROM EMPLOYEE E1
WHERE E1.nbr_of_dependents =
    (SELECT MAX(E2.emp_nbr_of_dependents)
     FROM EMPLOYEE E2
    )

• a set of rows.

Example: Display the names of those customers who bought the most expensive product(s). Hint: generate the product number(s) of the most expensive product(s); select those customers who have sales records that refer to that product(s):

SELECT DISTINCT cust_name
FROM CUSTOMER, SALES
WHERE cust_nbr = sales_cust_nbr and sales_prod_nbr IN
    (SELECT P1.prod_nbr
     FROM PRODUCT P1
     WHERE P1.prod_price = (SELECT MAX(P2.prod_price)
                            FROM PRODUCT P2 )
    )

Both of the above examples are examples in which the inner query does not relate to any column values from the outer query - these are called non-correlated subqueries.

There are also correlated subqueries - those that refer to a column value from the outer query:

Example: Display the names of those employees that have the highest number of dependents in their department. Hint: an employee has the highest number of dependents in the department if his number of dependents equals the maximum number of dependents across other employees in his/her department.

SELECT E1.emp_fname, E1.emp_lname
FROM EMPLOYEE E1
WHERE E1.nbr_of_dependents =
    ( SELECT MAX(E2.emp_nbr_of_dependents)
     FROM EMPLOYEE E2
     WHERE E2.emp_dept = E1.emp_dept
    )

Correlated subqueries are often use with the EXIST (or NOT EXISTS) operator:

Example: Display the names of those employees that are not representing any customers.

SELECT E1.emp_fname, E1.emp_lname
FROM EMPLOYEE E1
WHERE NOT EXISTS
    (SELECT C.cust_nbr
     FROM CUSTOMER C
     WHERE E1.emp_nbr = C.cust_rep
    )

Practice problems:
1. What is produced by the following queries:

```sql
select Cust_nbr, Cust_name
from CUSTOMER
where Cust_nbr not in (select sales_cust_nbr
                         from SALES);
```

```sql
select Prod_nbr, Prod_name, Prodline_name
from PRODUCT P1, PRODLINE
where P1.Prod_prodline = Prodline_nbr and
  P1.Prod_price < (select avg(P2.Prod_price)
                   from PRODUCT P2
                   where P2.prod_prodline = P1.prod_prodline);
```

2. Display the names of all employees that are not department managers.

3. Display the names and phone numbers of those customers who had the least number of items purchased in their customer group.