## Multiway JOIN in the FROM clause

* FULL OUTER JOIN – combines (unions) result of LEFT and RIGHT OUTER JOIN.

/\* Mysql does not support full outer join, but you can simulate it with left outer join UNION right outer join\*/

* Nested JOIN for a multiway join:

SELECT pnumber, dnum, lname, address, bdate

FROM ( (project JOIN department ON dnum=dnumber) JOIN employee ON mgr\_ssn=ssn)

WHERE plocation='Stafford';

* What the query returns?

## Aggregate Functions in SQL

* 1. Built-in aggregate functions in SQL are **COUNT***,* **SUM**, **MAX**, **MIN**, and **AVG.**

Examples:

* Retrieve the total number of employees in the company:

**SELECT COUNT**(\*)

FROM employee;

* Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.

SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)

FROM (employee JOIN department ON dno = dnumber)

WHERE dname = 'Research';

* Application of aggregate functions and attribute renaming with “AS”.

SELECT SUM(salary) AS Total\_Sal, MAX(salary) AS Highest\_Sal, MIN(salary) AS Lowest\_Sal, AVG(Salary) AS Average\_Sal

FROM employee;

*[ In PHPMyAdmin,* ***do not leave space*** *between COUNT and (\*). Neither for other aggregation functions.]*

* What does the following query return?

SELECT lname, fname

FROM employee

WHERE (SELECT count(\*)

FROM dependent

WHERE ssn = essn) >= 2;

## The GROUP BY and HAVING Clauses

### GROUP BY

* 1. Creates subgroups of tuples ***before*** applying aggregation functions.
  2. Aggregation function(s) will be applied on each group.
* Given definition of GROUP BY, can you guess what the following query will return?

SELECT dno, count(\*), avg(salary)

FROM employee

GROUP BY dno;

* What if there are some employees which dno is NULL for them?
* Note: the grouping attribute must appear in the SELECT clause:

SELECT **dno**, count(\*), avg(salary)

FROM employee

GROUP BY **dno**;

* GROUP BY may be applied to the result of a JOIN:

SELECT pnumber, pname, COUNT(\*)

FROM (project JOIN works\_on ON pnumber=pno )

GROUP BY pnumber, pname;

* What does the query return?

### HAVING clause

* 1. Provides a condition to choose or reject a ***group***. Similar to WHERE clause, but for a group, not a single tuple.
  2. Example:
  3. SELECT pnumber, pname, COUNT(\*)
  4. FROM (project JOIN works\_on ON pnumber=pno )
  5. GROUP BY pnumber, pname
  6. HAVING COUNT(\*) > 2;
* What this query returns?
* ***Can you write this query?*** *for each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.*

/\* Next session: lab 2 from chapter 7 \*/

## Combining the WHERE and the HAVING Clause

* Consider the query Q28: we want to count the *total* number of employees whose salaries exceed $40,000 in each department, but only for departments where more than five employees work.
* INCORRECT QUERY:

**SELECT** dno, **COUNT**(\*)

**FROM** employee

**WHERE** salary>40000

**GROUP BY** dno

**HAVING** **COUNT**(\*) > 5;

* Why is it incorrect? Or what does it return instead?
* Split the query into the following parts:

1. Departments which have more than five employees.

**SELECT** dno

**FROM** employee

**GROUP BY** dno

**HAVING COUNT**(\*) > 5;

Employees working IN these departments AND have a salary > 4000:

**SELECT** \*

**FROM** employee

**WHERE** salary>40000 **AND** dno **IN (SELECT** dno

**FROM** employee

**GROUP BY** dno

**HAVING COUNT**(\*) > 5) ;

1. Total number of such employees in each of these departments.

**SELECT** dno, count(\*)

**FROM** employee

**WHERE** salary>40000 **AND** dno **IN (SELECT** dno

**FROM** employee

**GROUP BY** dno

**HAVING COUNT**(\*) > 5)

**GROUP BY** dno;

## CASE clause

* It is similar to switch-case in programming languages.
* Used when a value or statement can be different based on certain conditions.

Example:

**UPDATE** EMPLOYEE

**SET** Salary =

**CASE WHEN** Dno = 5 **THEN** Salary + 2000

**WHEN** Dno = 4 **THEN** Salary + 1500

**WHEN** Dno = 1 **THEN** Salary + 3000

**END;**

**/\* Please do not run this query, it will change the state of our database. \*/**

## WITH clause

* WITH clause allows you to define a temporary table (**with** results of a query) that will only be used in the current query (not available in all SQL implementations).
* Allows you to break down a complex query to some shorter queries and then creating the main query in an step-by-step way.

An alternate approach to Q28:

* **Q28’:** **WITH BIGDEPTS (Dno) AS**

**( SELECT Dno**

**FROM EMPLOYEE**

**GROUP BY Dno**

**HAVING COUNT (\*) > 5)**

**SELECT** Dno, **COUNT** (\*)

**FROM** EMPLOYEE

**WHERE** Salary>40000 **AND** Dno **IN** BIGDEPTS

**GROUP BY** Dno;

**/\* we can not run this query since mysql does not support WITH \*/**

## EXPANDED Block Structure of SELECT SQL Queries