Chapter 5

JOINING TABLES

Objectives

* Understand the concepts of primary and foreign keys
* Connect tables using WHERE, INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN
* Use UNION and UNION ALL in joining tables

What we have used so far is a single table in retrieving its data. On this chapter, we will connect tables to generate a more comprehensive or detailed information in which most of modern databases have been set up.

Established enterprises implement relational databases in such a way that will eliminate duplication of data or redundancy. Duplicate data are expensive because you need to acquire more storages that will lead additional tasks and manpower to maintain it.

Relational database is organized to group related data in one table. Repetition of data is almost eliminated because you do not include columns that are not necessary to a particular table.

On this chapter, we will see that significance of primary and foreign keys as we establish the connection of tables. We will combine also groupings and filtering of rows as we select particular conditions on columns.

**Retrive data using WHERE clause**

We discussed the role of primary and foreign key in tables. Primary key sets the uniqueness of row. This key guarantees that existence of duplicate values is not possible once this is set during the creation of the table.

Foreign key on the other hand serves as a link from one table to another table which is on most cases contains the primary key of the given set of rows or records. You may find the significance of primary and foreign key in one-to-many or many-to-one relationships that exist between tables.

The given ERD shows the many-to-one relationship between EMPLOYEE and DEPARTMENT tables. WORKDEPT is set to be the FOREIGN KEY of EMPLOYEE table; DEPTNO (underlined) is set as the PRIMARY KEY of DEPARTMENT table.

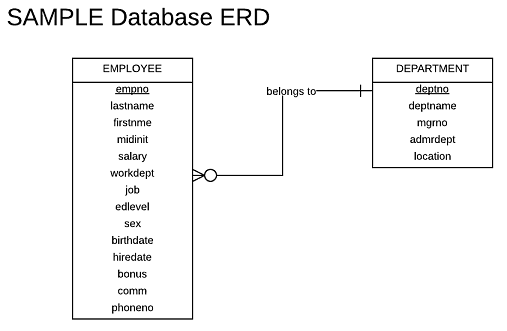


Figure 5-1. Primary and foreign keys many-to-one relationship

In other words, many WORKDEPTs can exist inside the EMPLOYEE table since many employees may work on one department as the WORKDEPT code suggests. DEPTNO is the same as WORKDEPT but its existence is LIMITED only once inside a DEPARTMENT table. DEPTNO guarantees that there will be only one existence of DEPTNO code and one can immediately identifies the name of that code represents (DEPTNAME). This is logical since no other departments should exist in an organization with the same name and purpose.

You can connect tables by simply using the WHERE clause. Let us say we want to get the department name in which an employee belong;

SELECT empNo, lastName, firstNme, workDept, deptName

FROM employee, department

WHERE workdept=deptNo

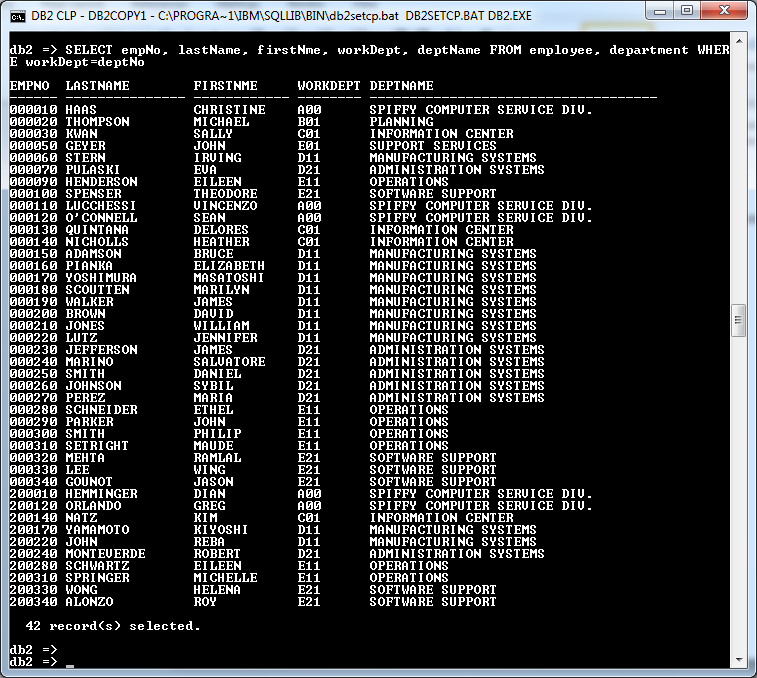


Figure 5-2. Connecting two tables using WHERE clause

If you check the structure of the EMPLOYEE’s table you will see the WORKDEPT that serves as the code in which the department employees work at. Another table DEPARTMENT consists of DEPTNO and DEPTNAME that lists all the departments of the organization.

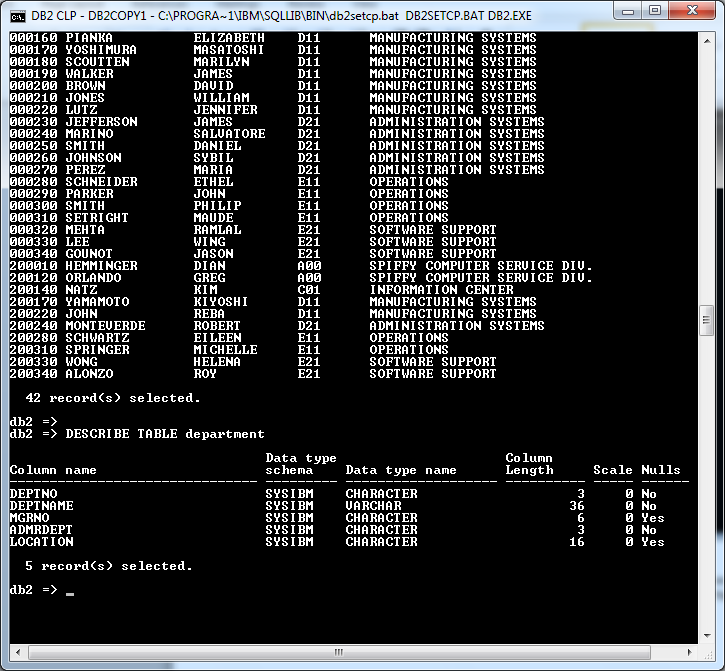


Figure 5-3. Structure of DEPARTMENT table

Our DEPARTMENT table contains 14 departments as we check its content.

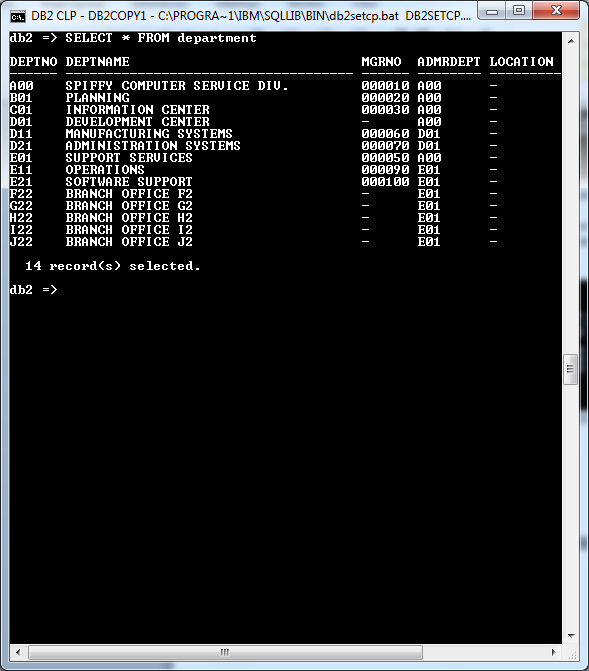


Figure 5-4. Content of DEPARTMENT table

We separate the two tables with a comma and equate WORKDEPT and DEPTNO after WHERE to establish the connection of the two tables. You will notice here that records which are present on both tables are included on the result. Department codes from F22 to J22 were not included on the result since these are not seen inside WORKDEPT of employee’s table.

**Use INNER JOIN to connect tables**

INNER JOIN or simply JOIN can be used to achieve the same result on what the WHERE clause is used. The syntax in using an INNER JOIN is :

SELECT <column name 1>, … <column name n> FROM <table name 1>

INNER JOIN <related table name 2>

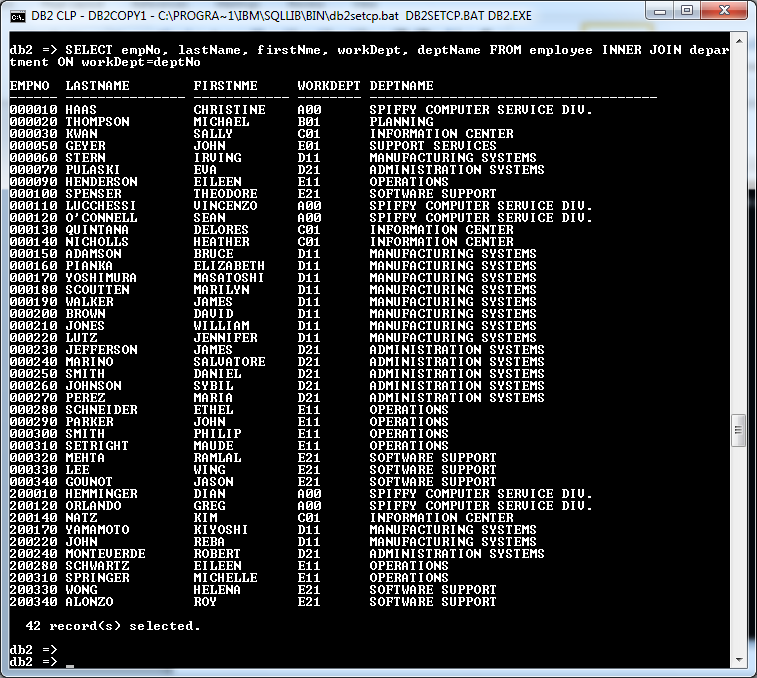
ON <column name of table 1> = <related column name of table 2>

Thus, to achieve the same result , we have:

SELECT empNo, lastName, firstNme, workDept, deptName FROM employee

INNER JOIN department

ON workDept=deptNo

Figure 5-5. Result of using INNER JOIN

Here, instead of simply separating the two tables by a comma, we have used INNER JOIN clause after the name of the first table and indicated the related table after INNER JOIN. Equate the two related columns that will serve as the connection for the two tables using ON before equality test.

We could achive the same result by dropping the word INNER. Simple JOIN is the same as INNER JOIN.

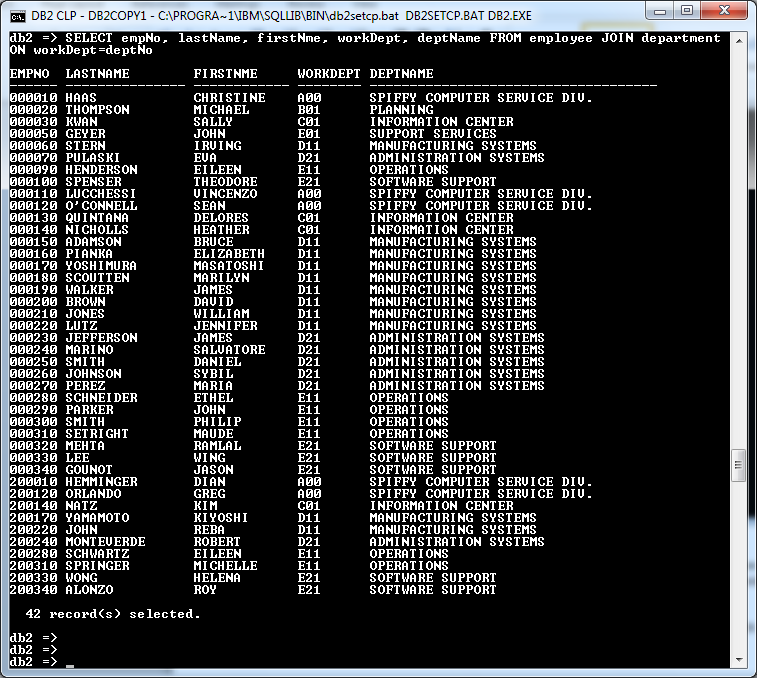


Figure 5-6. JOIN as equivalent of INNER JOIN

**Use LEFT JOIN to include all rows of the left table**

On most cases, not all rows or records have equivalence with the related table. What if the rows are not present on the left table but still we want to include these rows on our list? Here, we will employ LEFT JOIN in our statement.

The syntax of LEFT JOIN is somewhat similar to INNER JOIN:

SELECT <column name 1> , … <column name n> FROM <table name 1>

LEFT JOIN <related table name 2>

ON <column name of table 1> = <related column name of table 2>

Our problem requires to list all departments with or without projects. Sort the report according to department name.

From the given ERD, we connect DEPARTMENT with PROJECT table by DEPTNO column.

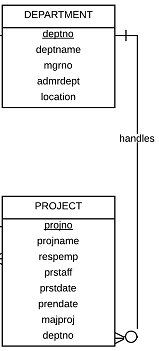


Figure 5-7. DEPARTMENT and PROJECT tables connection

We issue a statement:

SELECT d.deptNo, d.deptName, p.projNo, p.projName FROM department d

LEFT JOIN project p

ON d.deptNo=p.deptNo

ORDER BY 2

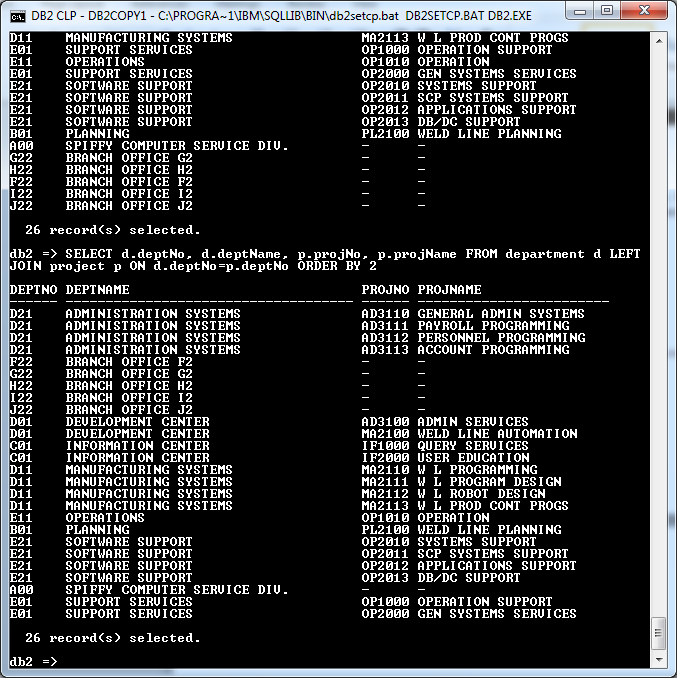


Figure 5-8. Using LEFT JOIN

We have used an alias name for department and project to prevent ambiguity on our statement. Remember that DEPTNO is present on both tables. The 2 in ORDER BY represent the second column of our statement which is the department name. The result indicates that branch offices and Spiffy Computer Service Division have no projects recorded because of the NULL values on PROJNO AND PROJNAME.

What you need to remember when using LEFT JOIN is that the first table after FROM serves as the master or parent and the second right adjoining table is the slave or child of the statement. Meaning, all columns that belong to the first table which are included in the SELECT statement will be populated (or displayed) regardless if the adjoining table has no equivalence on the second table --- on this case, there are no PROJNO present for department BRANCH OFFICES and SPIFFY COMPUTER SERVICE DIVISION.

**Populate all rows from the right table using RIGHT JOIN**

The second table with a RIGHT JOIN clause serves as the master and the first table is the slave. This is the opposite of the LEFT JOIN. You may use the following syntax:

SELECT <column name 1> , … <column name n> FROM <table name 1>

RIGHT JOIN <related table name 2>

ON <column name of table 1> = <related column name of table 2>

How do we achieve the same result produced by the LEFT JOIN statement that requires the to list all departments with or without projects using RIGHT JOIN? Only the placement of the tables will be changed to generate the same result:

SELECT d.deptNo, d.deptName, p.projNo, p.projName FROM project p

RIGHT JOIN department d

ON d.deptNo=p.deptNo

ORDER BY 2

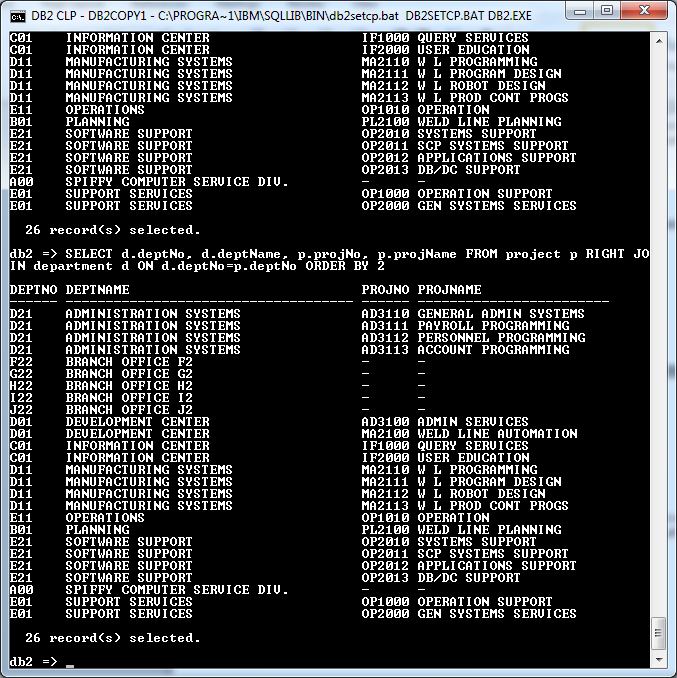


Figure 5-9. Using RIGHT JOIN

**Use FULL OUTER JOIN to include all rows of both tables**

The next requirement is to produce list of employees and the same time include all the departments of the organization regardless if it has no employees on it. Sort the list according to department name.

Our challenge here is how you are going to display all the departments even there are no employees recorded that work on that department. Here, we will use FULL OUTER JOIN (or simply FULL JOIN):

The syntax is:

SELECT <column name 1> , … <column name n> FROM <table name 1>

FULL JOIN <related table name 2>

ON <column name of table 1> = <related column name of table 2>

Our specific SELECT statement for the requirement will be:

SELECT e.empNo, e.lastName, e.firstNme, e.workDept, d.deptName FROM employee e

FULL JOIN department d

ON e.workDept = d.deptNo

ORDER BY 5

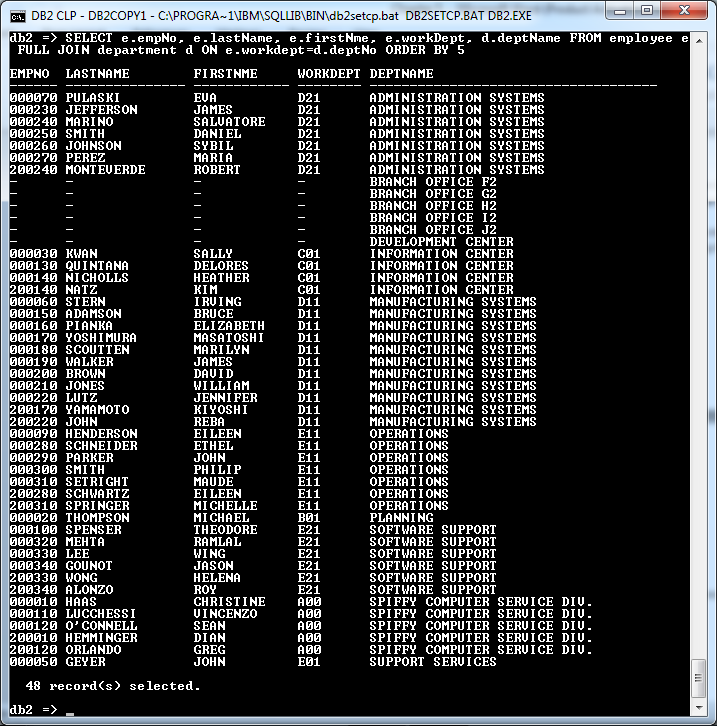


Figure 5-10. Using FULL JOIN

The results included branch offices from F2 to J2 and development center even it does not contain employee on that department. The EMPNO, LASTNAME, FIRST NAME, WORKDEPT contained NULL value in the list that shows no employee listed on these departments.

You may notice that we used an alias name even there is no possible ambiguity in the statement. It is an important practice to indicate an alias name for each column so you may not be lost in determining in which table does a column belong with.

**Work with Multiple JOINs**

You may work with nested joins if the problem requires it. SELECT statement is not limited with two tables. You may adjoin more than two tables if the requirement calls to include all columns in a single SELECT statement. Consider the following problem:

Create a list of departments with its employees that displays the name of the boss’s last name , first name and middle initial. Include in the list employee’s number, last name, first name, work dept, department name. Sort the list according to deparment name then by employee’s last name.

By inspection in the structure of the DEPARTMENT table, you will see that there is a column MGRNO that contains the manager’s employee number for that particular department. Remember, that a manager is also an employee. On this case, you can derive the name of the manager by simply referring it from manager’s number.

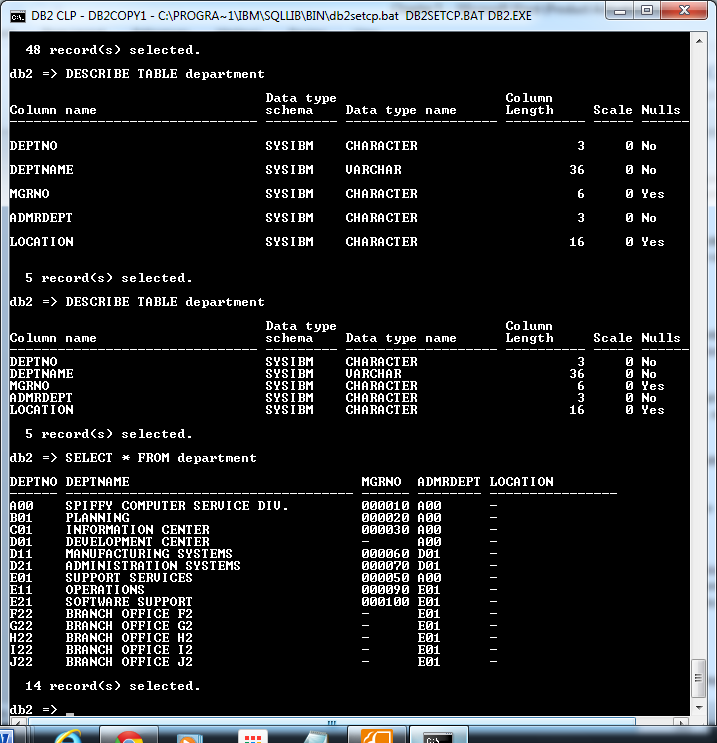


Figure 5-11. Structure and content of DEPARTMENT table

From our previous examples, we do not have problem getting the list of employees with its department. The following statement generates a partial list that closely match our requirement:

SELECT e.workDept, d.deptName, e.empNo, e.lastName, e.firstNme, d.mgrNo FROM department d

JOIN employee e

ON d.deptNo = e.workDept

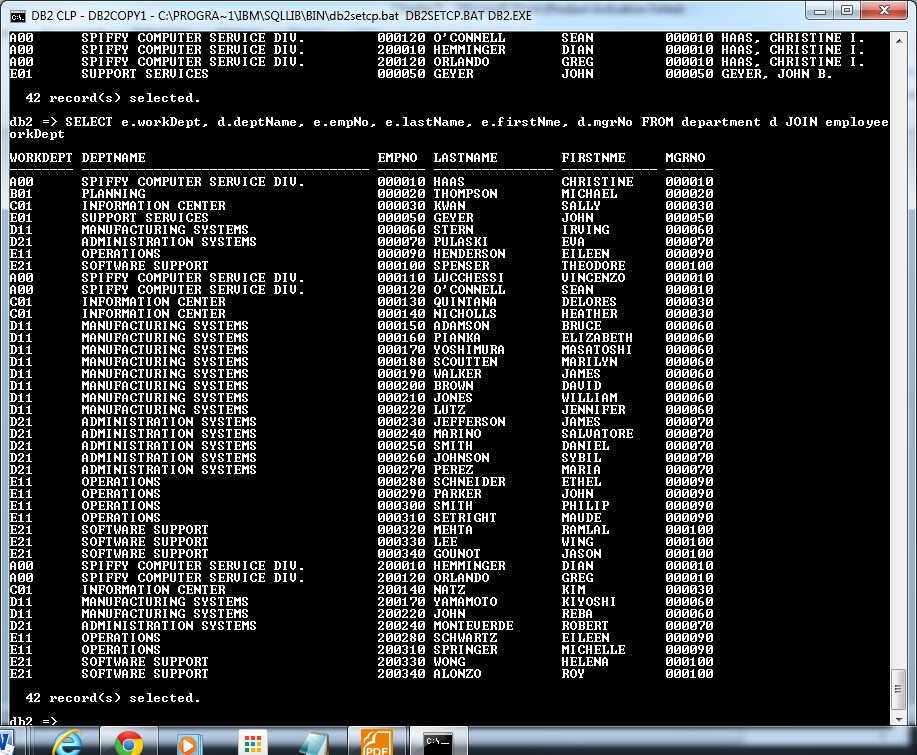


Figure 5-12. Partial result of the query that will included boss’s name

How come JOIN was used instead of LEFT JOIN or RIGHT JOIN? We used JOIN to include only rows that are present in EMPLOYEE and DEPARTMENT table. So, those departments that has no employee will not be included in the list.

You may considerthe result as one table that needs to connect to another table. The employee table will be the next table that you need to connect in order to refer at manager’s name.

But how do you connect another employee’s table if you already used it as the second table in the statement? Here you will use another set of EMPLOYEE table with an alias name of b. As we revised the previous statement we have:

SELECT e.workDept, d.deptName, e.empNo, e.lastName, e.firstNme, d.mgrNo, b.lastName || ‘, ‘ || b.firstNme || ‘ ‘ || b.midInit || ‘.’ “Boss Name “ , FROM department d

JOIN employee e

ON d.deptNo = e.workDept

LEFT JOIN employee b

ON d.mgrNo=b.empNo

ORDER BY 2, 4

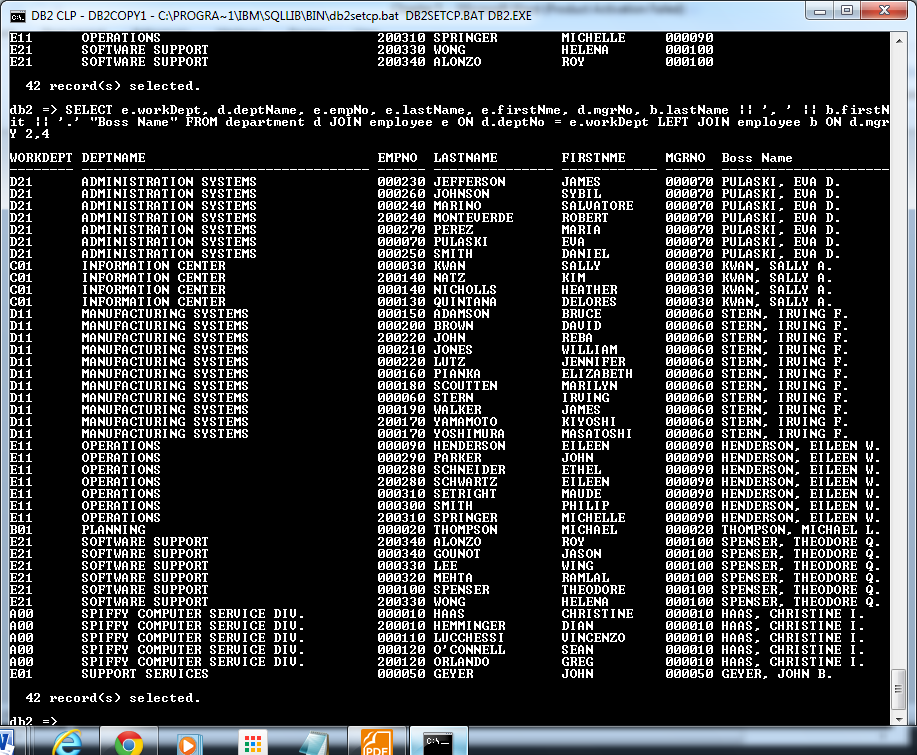


Figure 5-13. Result of the query that will included boss’s name

Take note that we concatenated boss name with double pipe symbols with b as its alias name. The result is sorted according to departments name (which is column 2) and then by employee’s last name (column 4).

**Apply filters, aggregate functions, and GROUP BYs in join statements**

The process is the same when you impose filters and groupings as long as the order of syntax is followed:

SELECT <column name 1>, … <column name n> FROM <table name 1>

WHERE <filter condition/s>

[JOIN/LEFT JOIN/RIGHT JOIN/FULL JOIN] <table name 2>

ON <column name of table 1> = <related column name of table 2>

GROUP BY <column name 1 or column number display, … column name n>

HAVING <filter condition/s>

ORDER BY <column name 1>, … <column name n>

Our next requirement needs to determine the number of employees per department excluding job positions of managers/president and department which has no employee. The list should include workdept, department name, last name of the boss and number of employees per department sorted according to department name. In addition, department with more than 4 employees should be included in the list.

The script that executes this is:

SELECT e.workDept, d.deptName, b.lastName "Boss Name", count(e.empno) "Number of Employees"

FROM department d JOIN employee e ON d.deptNo = e.workDept

LEFT JOIN employee b ON d.mgrNo = b.empNo

WHERE e.job NOT IN ('PRES','MANAGER')

GROUP BY e.workdept,d.deptName,b.lastName

HAVING COUNT(e.empno) > 4 ORDER BY 2



Figure 5-14. Result of applying filters, GROUP BY, and aggregate function

**Chapter 5 Exercises**

I. **Multiple Choice:**  Refer to SAMPLE database. Select the letter of the correct answer.

1. When many-to-zero is established between two tables, the minimum instance (or occurrence) with zero notation is:
2. NONE b. ONE C. MANY D. does not say
3. Which of the following instances (occurences) between SALESDETAIL and PRODUCT is NOT valid when zero-to-one or many-to-one relationships is established. The column that links between these tables have the same name of PRODCODE.
4. 3 PC0001 exists in SALESDETAIL table; only 1 PC0001 exists in PRODUCT table.
5. 1 PC0001 exists in SALESDETAIL table; only 1 PC0001 exists in PRODUCT table.
6. No existence of PC0001 in SALESDETAIL table; only 1 PC0001 exists in PRODUCT table.
7. 1 PC0001 exists in SALESDETAIL table; no existence of PC0001 in PRODUCT table.
8. How do you produce a report that lists employees’ department names, last names, first names, job postions sorted according to department name excluding managers and filtered only with workdept E21 and C01?
9. Which script lists the name of each department and the last name and first name of its manager. Sequence the list by department name?
10. What statement displays projno, projname and actno order by projno then by activity number using PROJECT and PROJACT tables. The list is filtered with project number beginning with AD. List identical rows once.
11. Which employees are assigned to project number AD3113. List by employee number, last name, and project number. Order the list by employee number and then by project number. List only one occurrence of duplicate result rows.
12. How do you list the department number, employee number, and salaries of all employees in A00 where the last line of the report displays the sum of all the salaries?
13. What script determines the total days employee consumed in a project? Subtract the start date from end date to determine the number of days. Display empno, last name, and total days in the project.

II. Programming Exercise

1. List the transaction number, sales date, product code, description, unit and quantity from hope database. Sort according to transaction number and product code.
2. Display the employee number, last name, first name, and job position from the job history of the employee.
3. Create a list of employees effective January 2011. List the employee number, last name, first name, position, deptcode, department name. Sort according from the highest to lowest salary.
4. Formulate a report that displays the total employees per department.