Joins Between Tables

Thus far, our queries have only accessed one table at a time. Queries can access multiple tables at once, or access the same table in such a way that multiple rows of the table are being processed at the same time. A query that accesses multiple rows of the same or different tables at one time is called a *join* query.

For example, say you wish to list all the employee records together with the name and location of the associated department. To do that, we need to compare the `deptno` column of each row of the `emp` table with the `deptno` column of all rows in the `dept` table, and select the pairs of rows where these values match. This would be accomplished by the following query:

```
SELECT emp.ename, emp.sal, dept.deptno, dept.dname, dept.loc FROM emp, dept WHERE emp.deptno = dept.deptno;

 ename  |   sal   | deptno |   dname    |   loc
--------+---------+--------+------------+----------
 MILLER | 1300.00 |     10 | ACCOUNTING | NEW YORK
 CLARK  | 2450.00 |     10 | ACCOUNTING | NEW YORK
 KING   | 5000.00 |     10 | ACCOUNTING | NEW YORK
 SCOTT  | 3000.00 |     20 | RESEARCH   | DALLAS
 JONES  | 2975.00 |     20 | RESEARCH   | DALLAS
 SMITH  |  800.00 |     20 | RESEARCH   | DALLAS
 ADAMS  | 1100.00 |     20 | RESEARCH   | DALLAS
 FORD   | 3000.00 |     20 | RESEARCH   | DALLAS
 WARD   | 1250.00 |     30 | SALES      | CHICAGO
 TURNER | 1500.00 |     30 | SALES      | CHICAGO
 ALLEN  | 1600.00 |     30 | SALES      | CHICAGO
 BLAKE  | 2850.00 |     30 | SALES      | CHICAGO
 MARTIN | 1250.00 |     30 | SALES      | CHICAGO
 JAMES  |  950.00 |     30 | SALES      | CHICAGO
(14 rows)
```

Observe two things about the result set:

There is no result row for department 40. This is because there is no matching entry in the `emp` table for department 40, so the join ignores the unmatched rows in the `dept` table.

Shortly we will see how this can be fixed.

It is more desirable to list the output columns qualified by table name rather than using * or leaving out the qualification as follows:

```
SELECT ename, sal, dept.deptno, dname, loc FROM emp, dept WHERE emp.deptno = dept.deptno;
```

Since all the columns had different names (except for `deptno` which therefore must be qualified), the parser automatically found out which table they belong to, but it is good style to fully qualify column names in join queries:

Join queries of the kind seen thus far can also be written in this alternative form:

```
SELECT emp.ename, emp.sal, dept.deptno, dept.dname, dept.loc FROM emp INNER JOIN dept ON emp.deptno = dept.deptno;
```

This syntax is not as commonly used as the one above, but we show it here to help you understand the following topics.

You will notice that in all the above results for joins no employees were returned that belonged to department 40 and as a consequence, the record for department 40 never appears. Now we will figure out how we can get the department 40 record in the results despite the fact that there are no matching employees. What we want the query to do is to scan the `dept` table and for each row to find the matching `emp` row. If no matching row is found we want some "empty" values to be substituted for the `emp` table's columns. This kind of query is called an *outer join*. (The joins we have seen so far are *inner joins*.) The command looks like this:

```
SELECT emp.ename, emp.sal, dept.deptno, dept.dname, dept.loc FROM dept LEFT OUTER JOIN
emp ON emp.deptno = dept.deptno;

 ename  |   sal   | deptno |   dname    |   loc
--------+---------+--------+------------+----------
 MILLER | 1300.00 |     10 | ACCOUNTING | NEW YORK
 CLARK  | 2450.00 |     10 | ACCOUNTING | NEW YORK
 KING   | 5000.00 |     10 | ACCOUNTING | NEW YORK
 SCOTT  | 3000.00 |     20 | RESEARCH   | DALLAS
 JONES  | 2975.00 |     20 | RESEARCH   | DALLAS
 SMITH  |  800.00 |     20 | RESEARCH   | DALLAS
 ADAMS  | 1100.00 |     20 | RESEARCH   | DALLAS
 FORD   | 3000.00 |     20 | RESEARCH   | DALLAS
 WARD   | 1250.00 |     30 | SALES      | CHICAGO
 TURNER | 1500.00 |     30 | SALES      | CHICAGO
 ALLEN  | 1600.00 |     30 | SALES      | CHICAGO
 BLAKE  | 2850.00 |     30 | SALES      | CHICAGO
 MARTIN | 1250.00 |     30 | SALES      | CHICAGO
 JAMES  |  950.00 |     30 | SALES      | CHICAGO
        |         |     40 | OPERATIONS | BOSTON
(15 rows)
```

This query is called a *left outer join* because the table mentioned on the left of the join operator will have each of its rows in the output at least once, whereas the table on the right will only have those rows output that match some row of the left table. When a left-table row is selected for which there is no right-table match, empty (`NULL`) values are substituted for the right-table columns.

An alternative syntax for an outer join is to use the outer join operator, "(+)", in the join condition within the `WHERE` clause. The outer join operator is placed after the column name of the table for which null values should be substituted for unmatched rows. So for all the rows in the `dept` table that have no matching rows in the `emp` table, Postgres Plus Advanced Server returns null for any select list expressions containing columns of `emp`. Hence the above example could be rewritten as:

```
SELECT emp.ename, emp.sal, dept.deptno, dept.dname, dept.loc FROM dept, emp WHERE
emp.deptno(+) = dept.deptno;

 ename  |   sal   | deptno |   dname    |   loc
--------+---------+--------+------------+----------
 MILLER | 1300.00 |     10 | ACCOUNTING | NEW YORK
 CLARK  | 2450.00 |     10 | ACCOUNTING | NEW YORK
 KING   | 5000.00 |     10 | ACCOUNTING | NEW YORK
 SCOTT  | 3000.00 |     20 | RESEARCH   | DALLAS
 JONES  | 2975.00 |     20 | RESEARCH   | DALLAS
 SMITH  |  800.00 |     20 | RESEARCH   | DALLAS
 ADAMS  | 1100.00 |     20 | RESEARCH   | DALLAS
 FORD   | 3000.00 |     20 | RESEARCH   | DALLAS
 WARD   | 1250.00 |     30 | SALES      | CHICAGO
 TURNER | 1500.00 |     30 | SALES      | CHICAGO
```

```
 ALLEN  | 1600.00 |     30 | SALES      | CHICAGO
 BLAKE  | 2850.00 |     30 | SALES      | CHICAGO
 MARTIN | 1250.00 |     30 | SALES      | CHICAGO
 JAMES  |  950.00 |     30 | SALES      | CHICAGO
        |         |     40 | OPERATIONS | BOSTON
(15 rows)
```

We can also join a table against itself. This is called a *self join*. As an example, suppose we wish to find the name of each employee along with the name of that employee's manager. So we need to compare the `mgr` column of each `emp` row to the `empno` column of all other `emp` rows.

```
SELECT e1.ename || ' works for ' || e2.ename AS "Employees and their Managers" FROM emp e1, emp e2 WHERE e1.mgr = e2.empno;

 Employees and their Managers
------------------------------
 FORD works for JONES
 SCOTT works for JONES
 WARD works for BLAKE
 TURNER works for BLAKE
 MARTIN works for BLAKE
 JAMES works for BLAKE
 ALLEN works for BLAKE
 MILLER works for CLARK
 ADAMS works for SCOTT
 CLARK works for KING
 BLAKE works for KING
 JONES works for KING
 SMITH works for FORD
(13 rows)
```

Here, the `emp` table has been re-labeled as `e1` to represent the employee row in the select list and in the join condition, and also as `e2` to represent the matching employee row acting as manager in the select list and in the join condition. These kinds of aliases can be used in other queries to save some typing, for example:

```
SELECT e.ename, e.mgr, d.deptno, d.dname, d.loc FROM emp e, dept d WHERE e.deptno = d.deptno;

 ename  | mgr  | deptno |   dname    |   loc
--------+------+--------+------------+----------
 MILLER | 7782 |     10 | ACCOUNTING | NEW YORK
 CLARK  | 7839 |     10 | ACCOUNTING | NEW YORK
 KING   |      |     10 | ACCOUNTING | NEW YORK
 SCOTT  | 7566 |     20 | RESEARCH   | DALLAS
 JONES  | 7839 |     20 | RESEARCH   | DALLAS
 SMITH  | 7902 |     20 | RESEARCH   | DALLAS
 ADAMS  | 7788 |     20 | RESEARCH   | DALLAS
 FORD   | 7566 |     20 | RESEARCH   | DALLAS
 WARD   | 7698 |     30 | SALES      | CHICAGO
 TURNER | 7698 |     30 | SALES      | CHICAGO
 ALLEN  | 7698 |     30 | SALES      | CHICAGO
 BLAKE  | 7839 |     30 | SALES      | CHICAGO
 MARTIN | 7698 |     30 | SALES      | CHICAGO
 JAMES  | 7698 |     30 | SALES      | CHICAGO
(14 rows)
```

This style of abbreviating will be encountered quite frequently.