Chapter 9

JOINING TABLES

- **Joining tables**
- **Self-join**
- **Outer Join**
- **SET Operators**

Joining Tables

In a relational database system, the total required data might not be available in a single table. Generally the data is scattered. Because if the total data is stored in a single table it will lead to a lot of ***redundancy.*** So, often we have to bring the data of two or more tables together to get the required information.

The act of combining two or more tables in such a way that you retrieve values from the columns of all the tables, to get the required data is called as ***joining***.

In order to join two tables, there must be a common column between those two tables. For example to join COURSES and BATCHES table, we use CCODE column as that is a common column between these two tables.

The following SELECT command is used to get information about batches from both COURSES and BATCHES table.

We take a row from BATCHES table and get the name of course from COURSES table using CCODE column of BATCHES. CCODE column of BATCHES is used to get the corresponding row from COURSES table. As we used equal to operator to join these two tables, the join is also called as **equi-join**.

```
select  bcode, batches.ccode, name, fcode, stdate
from  batches, courses
where  batches.ccode  = courses.ccode
```

```
BCODE CCODE NAME                           FCODE STDATE
----- ----- ------------------------------ ----- ---------
b1    ora   Oracle database                gk    12-JAN-01
b2    asp   ASP.NET                        da    15-JAN-01
b3    c     C programming                  hs    20-JAN-01
b4    xml   XML Programming                dh    02-MAR-01
b5    java  Java Language                  hs    05-APR-01
b6    vbnet VB.NET                         da    12-JUL-01
b7    ora   Oracle database                jc    15-AUG-01
```

In the above query the following are the important points.

1. Two tables are used in FROM clause. Tables are separated by comma.
2. WHERE clause contains the condition using which both the tables are to be joined.
3. Column CCODE is to be qualified using the table name as it exits in both the table that are used in FROM clause.

Table Alias

While joining the table we have to qualify the columns that are in more than one table using the table name. If table name is lengthy this process could be very tedious. Table alias is a short name that can be used to refer to the table name in the query. For example, the above query can be rewritten using table alias as follows:

```
select  bcode, b.ccode, name, fcode, stdate
from  batches b, courses c
where  b.ccode  = c.ccode
```

B is the alias to table BATCHES and C is the alias for COURSES. Throughout the query table BATCHES can be referred using the alias B and COURSES using alias C. As the purpose of using an alias is to shorten the reference to table, alias is generally very short. Also remember alias is available only in the query in which it is created.

Product of two tables

While tables are joined, if WHERE clause is not given then it results in PRODUCT of the table that are being joined. Product is the result in which each row of first table is joined with each row of the second table. This is also called as **Cartesian product**.

The following example will join

```
select bcode, b.ccode, name,  fcode, stdate, enddate
from  batches b, courses c;
```

The above command will result in 42 (6* 7) rows as we have 6 rows in COURSES and 7 rows in BATCHES table.

Product is generally the result of an error than the desired result.

Join condition and normal condition

While joining two tables the condition used to join the table is called as join condition. As we have above, without join condition, the result of joining will be product of the table.

Apart from the join condition, which is required for joining, normal conditions can also be given in WHERE clause. Then Oracle uses normal condition to filter rows and then joins the filtered rows based on the join condition.

The following query will get the details of batches that are completed along with name of the course. To get the name of the course it uses COURSES table and to select only those batches that are completed, it uses a normal condition as follows.

```
select bcode, b.ccode, name, fcode, stdate
from   batches b, courses c
where  b.ccode = c.ccode and enddate is not null

BCODE CCODE NAME                           FCODE STDATE
----- ----- ------------------------------ ----- ---------
b1    ora   Oracle database                gk    12-JAN-01
b2    asp   ASP.NET                        da    15-JAN-01
b3    c     C programming                  hs    20-JAN-01
b4    xml   XML Programming                dh    02-MAR-01
b5    java  Java Language                  hs    05-APR-01
```

It is also possible to order the result of join using ORDER BY clause as follows.

```
select bcode, b.ccode, name, fcode, stdate
from   batches b, courses c
where  b.ccode = c.ccode and enddate is not null
order by  stdate;
```

The above query displays the result of the join in the ascending order of STDATE column.

Joining more than two tables

Just like how two tables are joined, more than two tables can also be joined to get the required information. For example, in the above query we retrieved information about batches and course name. What if we want to get name of the faculty and not just the code? Then we have to use FACULTY table along with BATCHES and COURSES table.

To get the details of batches along with name of the course and name of the faculty, give the following:

```
select bcode,c.name course, f.name faculty, stdate
from  batches b, courses c, faculty f
where  b.ccode = c.ccode and
b.fcode = f.fcode
```

```
BCODE COURSE                         FACULTY                        STDATE
----- ------------------------------ ------------------------------ ---------
b1    Oracle database                George Koch                    12-JAN-01
b2    ASP.NET                        Dan Appleman                   15-JAN-01
b3    C programming                  Herbert Schildt                20-JAN-01
b4    XML Programming                David Hunter                   02-MAR-01
b5    Java Language                  Herbert Schildt                05-APR-01
b6    VB.NET                         Dan Appleman                   12-JUL-01
b7    Oracle database                Jason Couchman                 15-AUG-01
```

The query takes data from three tables – BATCHES, COURSES and FACULTY. It uses CCODE to join BATCHES table with COURSES table. It uses FCODE to join BATCHES table with FACULTY table.

When two or more tables are joined then the minimum number of conditions to be given is derived from the following formula:

Number of join conditions = Number of tables being joined - 1

If more than one column is used to join tables then the number of join conditions may be even more.

Self Join

When a table is joined with itself it is called as self-join. This is a variant of join. In this two copies of the same table are taken and joined as if they are two different types.

The following example demonstrates self-join. It displays the details of batches that started after batch B3. We take two copies of BATCHES table. In first copy we anchor at the row where batch code is B3. Then in the second copy of the table we take row where STDATE of the batch is more than STDATE of first copy. Since we already anchored at row with batch code B3 the STDATE of first copy will be the starting date of batch B3.

```
select  b2.*
from  batches b1, batches b2
where  b1.bcode = 'b3'
     and  b2.stdate > b1.stdate;


BCODE CCODE FCODE STDATE    ENDDATE      TIMING
----- ----- ----- --------- --------- ---------
b3    c     hs    20-JAN-01 27-FEB-01         3
b3    c     hs    20-JAN-01 27-FEB-01         3
b3    c     hs    20-JAN-01 27-FEB-01         3
b3    c     hs    20-JAN-01 27-FEB-01         3
```

In the above example b1 and b2 are two alias to the table BATCHES. Selection b2.* would mean we want to select all columns from first table.

Outer Join

When two tables are joined, only the rows that contain the common keys are selected. That means if BATCHES and COURSES tables are joined then you get the details of courses that have corresponding values in BATCHES table. If a course has no corresponding row (batch) in BATCHES table then it is not included in the output. Let us have a look at the following data we have in BATCHES and COURSES tables.

```
SQL> select ccode,name from courses;

CCODE NAME
----- ------------------------------
ora   Oracle database
vbnet VB.NET
c     C programming
asp   ASP.NET
java  Java Language
xml   XML Programming
cs    C Sharp
```

```
SQL> select bcode,ccode,stdate from batches;

BCODE CCODE STDATE
----- ----- ---------
b1    ora   12-JAN-01
```

```
b2    asp   15-JAN-01
b3    c     20-JAN-01
b4    xml   02-MAR-01
b5    java  05-APR-01
b6    vbnet 12-JUL-01
b7    ora   15-AUG-01
```

The following SELECT command will retrieve information from both the tables.

```
select  c.ccode, name,fee, bcode, stdate, enddate
from   courses c, batches b
where  c.ccode =b.ccode
order by c.ccode

CCODE NAME                                 FEE BCODE STDATE    ENDDATE
----- ------------------------------ --------- ----- --------- ---------
asp   ASP.NET                             5000 b2    15-JAN-01 05-MAR-01
c     C programming                       3500 b3    20-JAN-01 27-FEB-01
java  Java Language                       4500 b5    05-APR-01 10-MAY-01
ora   Oracle database                     4500 b1    12-JAN-01 20-FEB-01
ora   Oracle database                     4500 b7    15-AUG-01
vbnet VB.NET                              5500 b6    12-JUL-01
xml   XML Programming                     4000 b4    02-MAR-01 30-MAR-01
```

In the above join, we got the information about all courses except “C Sharp” because there is no corresponding row in BATCHES table for course code CS.

That means if parent table contains rows that do not have corresponding rows in child table, then Oracle will not retrieve the values of those rows of parent table.

Outer join allows you to retrieve the rows from parent table even though there is no corresponding rows for the parent table in the child table. In the following example, we get details of course CS though it doesn’t have any row in BATCHES table.

```
select  c.ccode, name,fee, bcode, stdate, enddate
from   courses c, batches b
where  c.ccode =b.ccode (+)
order by c.ccode;


CCODE NAME                                 FEE BCODE STDATE    ENDDATE
----- ------------------------------ --------- ----- --------- ---------
asp   ASP.NET                             5000 b2    15-JAN-01 05-MAR-01
c     C programming                       3500 b3    20-JAN-01 27-FEB-01
cs    C Sharp                             7000
java  Java Language                       4500 b5    05-APR-01 10-MAY-01
ora   Oracle database                     4500 b1    12-JAN-01 20-FEB-01
ora   Oracle database                        4500 b7    15-AUG-01
vbnet VB.NET                                 5500 b6    12-JUL-01
xml   XML Programming                        4000 b4    02-MAR-01 30-MAR-01
```

Compare this output with earlier output. Here as we used outer join by using (+) on the right of BATCHES table.

The plus (+) on the right of shorter table (taking into account unique value of CCODE column) is to add extra null rows with values in common column, that do exist in the left table, but not in the right table, and then join the rows.

Note*: When outer join is used, the order in which tables are listed in FROM clause is important. Also remember, (+) is to be used on the right of the table that is to be extended with null rows.*

As you can see in the above output, in the row CS there is no data for BCODE, STDATE and ENDDATE columns. This is understandable as there is no row in BATCHES table for course CS.

Also remember there is no possibility of having rows in BATCHES table that do not correspond to any course code in COURSES table. It is possible to have a parent row without child rows, but it is not possible to have a child row without a parent row. If we have defined foreign key constraint on the child table, this condition is implemented by Oracle whenever you insert or update rows in child table.

SET Operators

SET operators combine the results of two queries into one result. The following are the available SET operators. These operators are part of Relational Algebra operators.

Operator	**What it does?**
UNION	Returns unique rows of both the queries.
UNION ALL	Returns all rows selected by both the queries including duplicates.
INTERSECT	Returns common rows selected by both the queries.
MINUS	Returns rows that exist in the first query but not in the second query.

Table 1: Set Operators.

Assume there is a table called OLDCOURSES with the following rows:

```
select * from oldcourses;

CCODE NAME
----- --------------------
ora   Oracle8i
c     C Language
pas   Pascal Language
l123  Lotus 123
```

To get the list of all the courses from COURSES and OLDCOURSES tables by eliminating duplicates, give the following:

```
select ccode from oldcourses
union
select  ccode from courses;

CCODE
-----
asp
c
cs
java
l123
ora
pas
vbnet
xml
```

Though courses ORA and C exist in both the tables they are displayed only for once as UNION ignores duplicate entries.

UNION ALL is used to retrieve all rows from both the tables. It doesn’t ignore duplicates.

```
select ccode from oldcourses
union all
select ccode from courses;

CCODE
-----
ora
c
pas
l123
ora
```

```
vbnet
c
asp
java
xml
cs
```

The following query retrieves all courses that exists in both COURSES table and OLDCOURSES table using INTERSECT operator:

```
select ccode from courses
intersect
select ccode from oldcourses;

CCODE
-----
c
ora
```

To get the list of courses that exist in OLDCOURSES but not in COURSES table, give the following query with MINUS operator:

```
select ccode from oldcourses
minus
select ccode from courses;

CCODE
-----
l123
pas
```

RULES to be followed while using SET operators

1. Both queries should select the same number of columns.
2. Corresponding columns must be of the same data type. However the length and name of the columns may be different.
3. Column names of first query will be column headings.
4. Null values are treated as identical. Normally two null values are not equal. But when dealing with SET operators, if Oracle encounters two null values, one in first table and another in second table then Oracle will treat them as identical.

The following example will make the last point clear. Assume we have a row in first table with values 10 and null for columns C1 and C2 and another row in the second table with same values (10 and null) for columns C1 and C2. When you use UNION operator, Oracle will take null values in column C2 as equal and display the row with 10 and null only for once.

Summary

In relational databases, joining is a way of life. To join tables, all tables being joined must have a common column and it will be used in WHERE clause to establish join condition. When tables are joined, all non-unique columns are to be qualified using table name. A table can be joined to itself and it is called as self-join. Outer join is the operation of extending shorter table with null rows to match with values in join column of longer table. Plus sign (+) is used for outer join.

SET operators implement operators, UNION, INTERSECT and MINUS of relational algebra.

Exercises

1. What is required to join two tables?
2. What is meant by self-join?
3. How do you qualify a column that is existing in two or more tables that are being joined?
4. What is table alias? Is it stored anywhere?
5. What happens when you join two tables without any condition?
6. Display rollno, student name, pay date and amount paid.
7. Display rollno , student name, batch code , stdate of batch and faculty code.
8. Display rollno , student name, course name , stdate of batch and faculty code.
9. Display rollno , student name, course name , faculty code and enddate of all batches that were completed.
10. Display students who have got more number of characters in name than the student with roll number 10.
11. Display rollno, student name, email , pay date and amount paid.
12. In previous query include the details of student who haven’t paid anything so far.
13. Display the details of students who have paid nothing so far.