Database Programming

Cartesian Product and the Join Operations

Objectives

In this lesson, you will learn to:

- Name the Oracle proprietary joins and their ANSI/ISO SQL: 1999 counterparts
- Describe the purpose of join conditions
- Construct and execute a SELECT statement that results in a Cartesian product
- Construct and execute SELECT statements to access data from more than one table using an equijoin
- Construct and execute SELECT statements that add search conditions using the AND operator
- Apply the rule for using column aliases in a join statement

Purpose

Querying and returning information from one database table at a time would not be a problem if all data in the database were stored in only one table. But you know from data modeling that separating data into individual tables and being able to associate the tables with one another is the heart of relational database design.

Purpose (cont.)

Fortunately, SQL provides join conditions that enable information to be queried from separate tables and combined in one report.

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPT_ID
100	King	90
101	Kochhar	90
…		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

EMPLOYEE_ID	DEPT_ID	DEPT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
…		
102	90	Executive
205	110	Accounting
206	110	Accounting

Obtaining Data from Multiple Tables

Join Commands

The two sets of commands or syntax which can be used to make connections between tables in a database:

- Oracle proprietary joins
- ANSI/ISO SQL 99 compliant standard joins

ORACLE Proprietary Joins

To query data from more than one table using the Oracle proprietary syntax, use a join condition in the WHERE clause.

The basic format of a join statement is:

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

ORACLE Proprietary Joins (cont.)

Imagine the problem arising from having two students in the same class with the same last name. When needing to speak to "Jackson," the teacher clarifies which "Jackson" by prefacing the last name with the first name. To make it easier to read a Join statement and to speed up database access, it is good practice to preface the column name with the table name.

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

ORACLE Proprietary Joins (cont.)

This is called "qualifying your columns." The combination of table name and column name helps eliminate ambiguous names when two tables contain a column with the same column name. Note: When the same column name appears in both tables, the column name must be prefaced with the name of the table.

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

Join Syntax Example

In the following example, which two tables are being joined? Which identical columns do these tables share?

```
SELECT d_play_list_items.event_id, d_play_list_items.song_id,
               d_track_listings.cd_number
FROM    d_play_list_items, d_track_listings
WHERE d_play_list_items.song_id = d_track_listings.song_id;
```

Join Syntax Example

If you wanted to join three tables together, how many joins would it take? How many bridges are needed to join three islands?

EQUIJOIN

Sometimes called a "simple" or "inner" join, an equijoin is a table join that combines rows that have the same values for the specified columns. In the example shown on the next slide, the what, where and how are required for the join condition.

EQUIJOIN (cont.)

What? The SELECT clause specifies the column names to retrieve.

Where? The FROM clause specifies the two tables that the database must access.

How? The WHERE clause specifies how the tables are to be joined.

EQUIJOIN (cont.)

```
SELECT d_play_list_items.song_id, d_play_list_items.event_id,
       d_track_listings.cd_number
FROM   d_play_list_items, d_track_listings
WHERE  d_play_list_items.song_id = d_track_listings.song_id;
```

SONG_ID	EVENT_ID	CD_NUMBER
45	100	92
46	100	93
47	100	91
48	105	95
49	105	91
47	105	91

Cartesian Product Join

If two tables in a join query have no join condition specified in the WHERE clause or the join condition is invalid, the Oracle Server returns the Cartesian product of the two tables. This is a combination of each row of one table with each row of the other. A Cartesian product always generates many rows and is rarely useful. For example, the Cartesian product of two tables, each with 100 rows, has 10,000 rows! This may not be what you were trying to retrieve.

To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

Cartesian Product Join Example

EMPLOYEES (20 rows)

```
SELECT employee_id, last_name, department_id
FROM employees
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
201	Hartstein	20
202	Fay	20

20 rows returned in 0.01 seconds Download

DEPARTMENTS (8 rows)

```
SELECT department_id, department_name, location_id
FROM departments
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows returned in 0.15 seconds Download

Cartesian Product Join Example (cont.)

THE CARTESIAN RESULT SET (20 X 8 rows)

```
SELECT employee_id, departments.department_id, location_id
FROM employees, departments
```

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	10	1700
101	10	1700
102	10	1700
103	10	1700
104	10	1700

201	190	1700
202	190	1700
205	190	1700
206	190	1700

160 rows returned in 0.02 seconds Download

Restricting Rows In a Join

As with single-table queries, the WHERE clause can be used to restrict the rows considered in one or more tables of the join. The query shown uses the AND operator to restrict the rows returned. Compare this result with the previous query.

```
SELECT d_play_list_items.song_id, d_play_list_items.event_id,
       d_track_listings.cd_number
FROM   d_play_list_items, d_track_listings
WHERE  d_play_list_items.song_id = d_track_listings.song_id
  AND  d_play_list_items.event_id < 105;
```

SONG_ID	EVENT_ID	CD_NUMBER
45	100	92
46	100	93
47	100	91

Aliases

Working with lengthy column and table names can be cumbersome. Fortunately, there is a way to shorten the syntax using aliases. To distinguish columns that have identical names but reside in different tables, use column aliases.

Aliases (cont.)

Column aliases were used in the query below. When column names are not duplicated between two tables, you do not need to add the table name to it.

```
SELECT d_track_listings.song_id AS TRACK,
       d_play_list_items.song_id AS " PLAY LIST"
FROM   d_play_list_items,
       d_track_listings
WHERE  d_play_list_items.song_id = d_track_listings.song_id;
```

TRACK	PLAYLIST
45	45
46	46
47	47

Table Aliases

Another way to make statements easier to read is to use table aliases. A table alias is similar to a column alias; it renames an object within a statement. It is created by entering the new name for the table just after the table name in the from-clause. However, if a table alias is used in the FROM clause, then that table alias must be substituted for the table name throughout the SELECT statement.

```
SELECT p.song_id , t.song_id
FROM   d_play_list_items p, d_track_listings t
WHERE  p.song_id = t.song_id;
```

Terminology

Key terms used in this lesson included:

- Alias
- Cartesian Product
- Equijoin
- Join Conditions
- Proprietary Join

Summary

In this lesson you have learned to:

- Name the Oracle proprietary joins and their ANSI/ISO SQL: 1999 counterparts
- Describe the purpose of join conditions
- Construct and execute a SELECT statement that results in a Cartesian product

Summary (cont.)

In this lesson you have learned to:

- Construct and execute SELECT statements to access data from more than one table using an equijoin
- Construct and execute SELECT statements that add search conditions using the AND operator
- Apply the rule for using column aliases in a join statement