Database Programming with SQL

7-1
Oracle Equijoin and Cartesian Product
Objectives

In this lesson, you will learn to:

• Name the Oracle proprietary joins and their ANSI/ISO SQL: 99 counterparts

• 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 table aliases in a join statement
Purpose

• The previous section looked at querying and returning data from more than one table in a relational database using ANSI/ISO SQL: 99 syntax.

• Legacy versions of Oracle databases required joins to use Oracle Proprietary join syntax, and many of these older databases are still in use.

• This lesson introduces Oracle Proprietary join syntax for Equijoins and Cartesian Product, and their ANSI/ISO SQL: 99 counterparts.
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
Join Comparison


<table>
<thead>
<tr>
<th>Oracle Proprietary Join</th>
<th>ANSI/ISO SQL: 1999 Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartesian Product</td>
<td>Cross Join</td>
</tr>
<tr>
<td>Equijoin</td>
<td>NATURAL JOIN</td>
</tr>
<tr>
<td></td>
<td>JOIN USING clause</td>
</tr>
<tr>
<td></td>
<td>JOIN ON clause (if the equality operator is used)</td>
</tr>
<tr>
<td>Non-equijoin</td>
<td>ON clause</td>
</tr>
</tbody>
</table>
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:

```sql
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```
ORACLE Proprietary Joins

• 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

• 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.

• When the same column name appears in both tables, the column name must be prefaced with the name of the table.
Join Syntax Example

• To qualify the columns, you use the syntax `tablename.columnname` as shown in the example below.

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

• An equijion is equivalent to ANSI:
  – NATURAL JOIN
  – JOIN USING
  – JOIN ON (when the join condition uses "=")

• The next slide demonstrates the what, where and how required to join the tables.
EQUIJOIN

• What? The SELECT clause specifies the column names to display.

• Where? The FROM clause specifies the tables that the database must access, separated by commas.

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

• An Equijoin uses the equals operator to specify the join condition.
EQUIJOIN

```
SELECT employees.last_name, employees.job_id, jobs.job_title
FROM employees, jobs
WHERE employees.job_id = jobs.job_id;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>JOB_TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gietz</td>
<td>AC_ACCOUNT</td>
<td>Public Accountant</td>
</tr>
<tr>
<td>Higgins</td>
<td>AC_MGR</td>
<td>Accounting Manager</td>
</tr>
<tr>
<td>Whalen</td>
<td>AD_ASST</td>
<td>Administration Assistant</td>
</tr>
<tr>
<td>King</td>
<td>AD_PRES</td>
<td>President</td>
</tr>
<tr>
<td>Kochhar</td>
<td>AD_VP</td>
<td>Administration Vice President</td>
</tr>
<tr>
<td>De Haan</td>
<td>AD_VP</td>
<td>Administration Vice President</td>
</tr>
<tr>
<td>Ernst</td>
<td>IT_PROG</td>
<td>Programmer</td>
</tr>
<tr>
<td>Hunold</td>
<td>IT_PROG</td>
<td>Programmer</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
EQUIJOIN

• Another example:

```
SELECT employees.last_name, departments.department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPARTMENT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whalen</td>
<td>Administration</td>
</tr>
<tr>
<td>Hartstein</td>
<td>Marketing</td>
</tr>
<tr>
<td>Fay</td>
<td>Marketing</td>
</tr>
<tr>
<td>Davies</td>
<td>Shipping</td>
</tr>
<tr>
<td>Vargas</td>
<td>Shipping</td>
</tr>
<tr>
<td>Rajs</td>
<td>Shipping</td>
</tr>
<tr>
<td>Mourgos</td>
<td>Shipping</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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 is equivalent to an ANSI CROSS JOIN.

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

• In this query, the join condition has been omitted:

```sql
SELECT employees.last_name, departments.department_name
FROM employees, departments;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPARTMENT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abel</td>
<td>Administration</td>
</tr>
<tr>
<td>Davies</td>
<td>Administration</td>
</tr>
<tr>
<td>De Haan</td>
<td>Administration</td>
</tr>
<tr>
<td>Ernst</td>
<td>Administration</td>
</tr>
<tr>
<td>Fay</td>
<td>Administration</td>
</tr>
<tr>
<td>Gietz</td>
<td>Administration</td>
</tr>
<tr>
<td>Grant</td>
<td>Administration</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

160 rows returned in 0.01 seconds
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.

```
SELECT employees.last_name, employees.job_id, jobs.job_title
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
AND employees.department_id = 80;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>JOB_TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zlotkey</td>
<td>SA_MAN</td>
<td>Sales Manager</td>
</tr>
<tr>
<td>Taylor</td>
<td>SA_REP</td>
<td>Sales Representative</td>
</tr>
<tr>
<td>Abel</td>
<td>SA_REP</td>
<td>Sales Representative</td>
</tr>
</tbody>
</table>
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 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.
Table Aliases

• Table aliases are used in the query below.

```sql
SELECT last_name, e.job_id, job_title
FROM employees e, jobs j
WHERE e.job_id = j.job_id
AND department_id = 80;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>JOB_TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zlotkey</td>
<td>SA_MAN</td>
<td>Sales Manager</td>
</tr>
<tr>
<td>Taylor</td>
<td>SA_REP</td>
<td>Sales Representative</td>
</tr>
<tr>
<td>Abel</td>
<td>SA_REP</td>
<td>Sales Representative</td>
</tr>
</tbody>
</table>

• When column names are not duplicated between two tables, you do not need to add the table name or alias to the column name.
Table Aliases

• 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.

• Using the name of a table in the SELECT clause that has been given an alias in the FROM clause will result in an error.

```
SELECT last_name, employees.job_id, job_title
FROM employees e, jobs j
WHERE e.job_id = j.job_id
AND department_id = 80;
```

ORA-00904: "EMPLOYEES"."JOB_ID": invalid identifier
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?

• To join three tables, you need to add another join condition to the WHERE clause using the AND operator

Table 1  Table 2  Table 3
Join Syntax Example

• Suppose we need a report of our employees and the city where their department is located?

• We need to join three tables: employees, departments and locations.

```sql
SELECT last_name, city
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
  AND d.location_id = l.location_id
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abel</td>
<td>Oxford</td>
</tr>
<tr>
<td>Davies</td>
<td>South San Francisco</td>
</tr>
<tr>
<td>De Haan</td>
<td>Seattle</td>
</tr>
<tr>
<td>Ernst</td>
<td>Southlake</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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: 99 counterparts

• 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 table aliases in a join statement