

Oracle Database 10g
SQL Fundamentals – lab 3

Joining Tables

- Sometimes you need to use data from more than one table.
- For example to produce a report that displays data from two separate tables: `EMPLOYEES` and `DEPARTMENTS` that includes `Employee_ID`, `Department_ID`, and `Department_Name`:
 - Employee IDs exist in the `EMPLOYEES` table.
 - Department IDs exist in both the `EMPLOYEES` and `DEPARTMENTS` tables.
 - Department names exist in the `DEPARTMENTS` table.
- To produce the report, you need to link (JOIN) the `EMPLOYEES` and `DEPARTMENTS` tables and access data from both of them.

Joining Tables

- Use a join to query data from more than one table.

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

- Write the join condition in the WHERE clause.
- Prefix the column name with the table name when the same column name appears in more than one table.
- To join n tables together, you need a minimum of $n-1$ join conditions.

Equijoins

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

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
144	Vargas	50	50	1500

■ ■ ■

19 rows selected.

Why Use Join Condition??

- If we do not the join condition, the Cartesian product of the two tables will be displayed.
- With no join condition, each employee will be displayed 8 times, once for each department!!

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

Additional Search Conditions

Using the AND Operator

In addition to the join, you may have criteria for your WHERE clause to restrict the rows under consideration for tables in the join

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

...

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

...

Example: To display employee Matos' information, you need an additional condition in the WHERE clause.

AND last_name = 'Matos';

Qualifying Ambiguous Column Names

- **Use table prefixes to qualify column names that are in multiple tables.**
- **Always improve performance by using table prefixes because you tell the Oracle Server exactly where to find the columns.**
- **Use column aliases in all clauses such as in the `SELECT` clause or the `ORDER BY` clause.**

Using Table Aliases

- Simplify queries by using table aliases.
- Table aliases help to keep SQL code smaller, therefore using less memory.
- If a table alias is used for a particular table name in the FROM clause, then that table alias must be substituted for the table name throughout the SELECT statement.

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id;
```


Outer Joins

Use an outer join to also see rows that do not meet the join condition (Records with No Direct Match).

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...
20 rows selected.

There are no employees in department 190.

Outer Joins Syntax

- The Outer join operator is the plus sign (+) which can be placed on either side of the WHERE clause condition, but not on both sides.
- Place the outer join symbol following the name of the column in the table without the matching rows.
- In other words, it is *placed on the “side” of the join that is deficient in information (will contain Null values)*

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column (+) = table2.column;
```

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column = table2.column (+);
```

Using Outer Joins

Display employee last names, department ID's and department names for all departments.

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id(+) = d.department_id ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
...		
Gietz	110	Accounting
		Contracting

20 rows selected.

Data Manipulation Language (DML)

A DML statement is executed when you:

- **INSERT**: insert new rows to a table
- **UPDATE**: update existing rows in a table
- **DELETE**: delete existing rows from a table

The INSERT Statement Syntax

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Only one row is inserted at a time.

Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally, list the columns in the INSERT clause.

```
INSERT INTO departments(department_id, department_name,  
                        manager_id, location_id)  
VALUES      (70, 'Public Relations', 100, 1700);  
1 row created.
```

- Enclose character and date values within single quotation marks.

Inserting Rows with Null Values

- **Implicit method: Omit the column from the column list.**

```
INSERT INTO departments (department_id,  
                          department_name  )  
VALUES (30, 'Purchasing');  
1 row created.
```

- **Explicit method: Specify the NULL keyword in the VALUES clause.**

```
INSERT INTO departments  
VALUES (100, 'Finance',  NULL,  NULL);  
1 row created.
```

The UPDATE Statement Syntax

- **Modify existing rows with the UPDATE statement.**

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE     condition];
```

- **Update more than one row at a time, if required.**
- **Note: In general, use the primary key to identify a single row.**
- **Using other columns can unexpectedly cause several rows to be updated**

Updating Rows in a Table

- Specific row or rows are modified if you specify the **WHERE** clause.

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- All rows in the table are modified if you omit the **WHERE** clause.

```
UPDATE    copy_emp
SET       department_id = 110;
22 rows updated.
```

Updating Two Columns

- Update employee 114's job and salary to sales man and 2000.

```
UPDATE    employees
SET       job_id   = 'sales_man',
          salary   = 2000
WHERE     employee_id = 114;
1 row updated.
```

The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM] table  
[WHERE condition];
```

Deleting Rows from a Table

- **Specific rows are deleted if you specify the WHERE clause.**

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 row deleted.
```

- **All rows in the table are deleted if you omit the WHERE clause.**

```
DELETE FROM copy_emp;
22 rows deleted.
```

Deleting Rows: Integrity Constraint Error

```
DELETE FROM departments
WHERE      department_id = 60;
```

```
DELETE FROM departments
      *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

You cannot delete a row that contains a primary key that is used as a foreign key in another table.

Data Definition Language Commands (DDL)

- **CREATE**
- **ALTER**
- **DROP**

Creating Tables

- Create the table.

```
CREATE TABLE dept
      (deptno NUMBER (2) ,
       dname  VARCHAR2 (14) ,
       loc    VARCHAR2 (13) ) ;
```

Table created.

You must specify:

Table name

Column name, column data type, and column size

Data Types

Data Type	Description
<code>VARCHAR2 (size)</code>	Variable-length character data, maximum size 4000
<code>CHAR (size)</code>	Fixed-length character data, maximum size 2000
<code>NUMBER (p, s)</code>	Variable-length numeric data (precision, scale)
<code>DATE</code>	Date and time values

The ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column
- Drop a column

Adding a Column

- You use the ADD clause to add columns.

```
ALTER TABLE dept80
ADD          (job_id VARCHAR2(9)) ;
Table altered.
```

- The new column becomes the last column.
- To add many columns, separate their definitions by commas.
- If the table already contains data, the new column will have a null value for all rows.

Modifying a Column Definition

- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
Table altered.
```

- You can increase the width of a numeric or character columns.

Dropping a Column

- Use the **DROP COLUMN** clause to drop columns you no longer need from the table.

```
ALTER TABLE dept80
DROP COLUMN job_id;
Table altered.
```

- Only one column can be dropped at a time.
- The table must have at least one column remaining in it after it is altered.
- It automatically drops all constraints on this column.
- The column may or may not contain data.
- Once a column is dropped, it cannot be recovered.

Dropping a Table

- All data and structure in the table is deleted.
- The **CASCADE CONSTRAINTS** option removes dependent integrity constraints.

```
DROP TABLE dept80;  
Table dropped.
```

Practice SQL Developer

Practice

- **Add one row of data to the yourname_employee table.**
11, 'Jon', Adams', 2000
12, 'Alan', 'Robin', 800
- **Confirm your addition to the table.**

```
create TABLE yourname_employee(  
    id number (2) NOT NULL,  
    first_name varchar2 (10),  
    last_name varchar2 (10),  
    salary number (6))
```

Results

Script Output

Explain

Autotrace

DBMS Output

OWA Output

results:

ORCLconnHR



```
INSERT INTO yourname_employee (id, first_name, last_name, salary )  
VALUES (11, 'Jon', 'Adams', 2000);
```

```
INSERT INTO yourname_employee (id, first_name, last_name, salary )  
VALUES (12, 'Alan', 'Robin', 800);
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

Results:

```
select *  
FROM yourname_employee
```

Results:

	ID	FIRST_NAME	LAST_NAME	SALARY
1	11	Jon	Adams	2000
2	12	Alan	Robin	800

Thank You