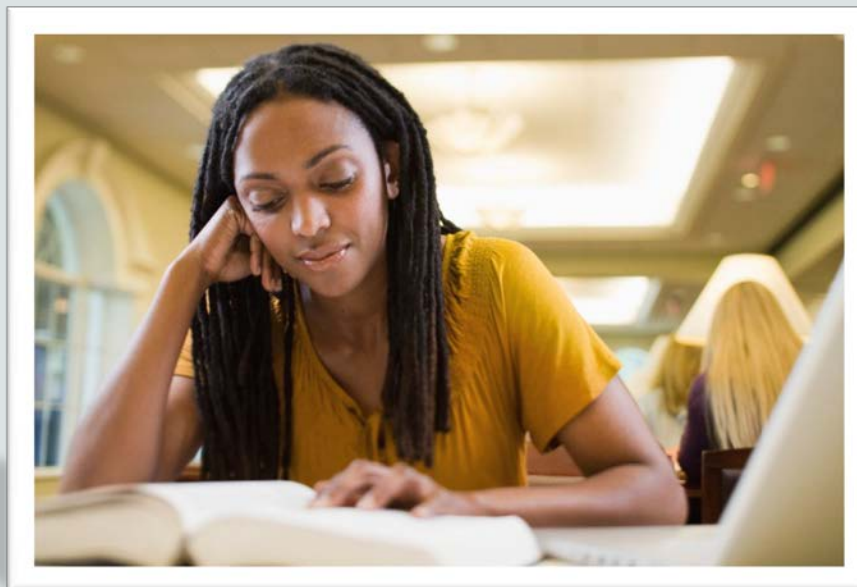




Database Programming with PL/SQL

3-3

Manipulating Data in PL/SQL



Objectives

This lesson covers the following objectives:

- Construct and execute PL/SQL statements that manipulate data with DML statements
- Describe when to use implicit or explicit cursors in PL/SQL
- Create PL/SQL code to use SQL implicit cursor attributes to evaluate cursor activity

Purpose

- You have learned that you can include `SELECT` statements that return a single row in a PL/SQL block.
- The data retrieved by the `SELECT` statement must be held in variables using the `INTO` clause.
- In this lesson, you learn how to include data manipulation language (DML) statements, such as `INSERT`, `UPDATE`, `DELETE`, and `MERGE` in PL/SQL blocks.
- DML statements will help you perform a task on more than a single row.

Create Copy of Original Table

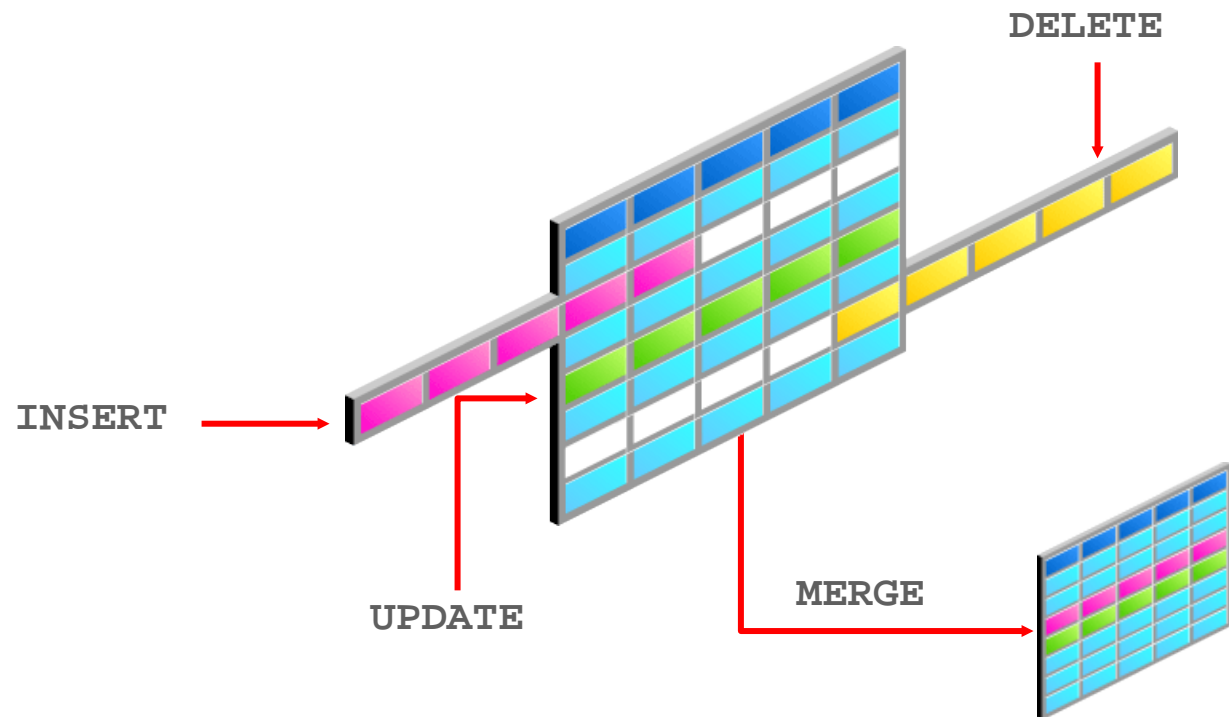
- It is very important that you do NOT modify the existing tables (such as EMPLOYEES and DEPARTMENTS), because they will be needed later in the course.
- The examples in this lesson use the COPY_EMP table.
- If you haven't already created the COPY_EMP table, do so now by executing this SQL statement:

```
CREATE TABLE copy_emp  
  AS SELECT *  
  FROM employees;
```

Manipulating Data Using PL/SQL

Make changes to data by using DML commands within your PLSQL block:

- INSERT
- UPDATE
- DELETE
- MERGE



Manipulating Data Using PL/SQL

- You manipulate data in the database by using the DML commands.
- You can issue the DML commands—`INSERT`, `UPDATE`, `DELETE`, and `MERGE`—without restriction in PL/SQL.
 - The `INSERT` statement adds new rows to the table.
 - The `UPDATE` statement modifies existing rows in the table.
 - The `DELETE` statement removes rows from the table.

Manipulating Data Using PL/SQL

- The `MERGE` statement selects rows from one table to update and/or insert into another table.
- The decision whether to update or insert into the target table is based on a condition in the `ON` clause.
 - Note: `MERGE` is a deterministic statement—that is, you cannot update the same row of the target table multiple times in the same `MERGE` statement.
 - You must have `INSERT` and `UPDATE` object privileges in the target table and the `SELECT` privilege in the source table.

Inserting Data

- The `INSERT` statement adds new row(s) to a table.
- Example: Add new employee information to the `COPY_EMP` table.

```
BEGIN
  INSERT INTO copy_emp
    (employee_id, first_name, last_name,
    email,
    hire_date, job_id, salary)
  VALUES (99, 'Ruth', 'Cores',
    'RCORES', SYSDATE, 'AD_ASST', 4000);
END;
```

- One new row is added to the `COPY_EMP` table.

Updating Data

- The UPDATE statement modifies existing row(s) in a table.
- Example: Increase the salary of all employees who are stock clerks.

```
DECLARE
  v_sal_increase employees.salary%TYPE := 800;
BEGIN
  UPDATE copy_emp
    SET salary = salary + v_sal_increase
    WHERE job_id = 'ST_CLERK';
END;
```



Deleting Data

- The DELETE statement removes row(s) from a table.
- Example: Delete rows that belong to department 10 from the COPY_EMP table.

```
DECLARE
    v_deptno    employees.department_id%TYPE := 10;
BEGIN
    DELETE FROM    copy_emp
        WHERE    department_id = v_deptno;
END;
```



Merging Rows

- The MERGE statement selects rows from one table to update and/or insert into another table.
- Insert or update rows in the `copy_emp` table to match the `employees` table.

```
BEGIN
  MERGE INTO copy_emp c USING employees e
    ON (e.employee_id = c.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      c.email           = e.email,
      . . .
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, ...e.department_id);
END;
```

Getting Information From a Cursor

- Look again at the DELETE statement in this PL/SQL block.

```
DECLARE
  v_deptno    employees.department_id%TYPE := 10;
BEGIN
  DELETE FROM  copy_emp
    WHERE  department_id = v_deptno;
END;
```

- It would be useful to know how many COPY_EMP rows were deleted by this statement.
- To obtain this information, we need to understand cursors.

What is a Cursor?

- Every time an SQL statement is about to be executed, the Oracle server allocates a private memory area to store the SQL statement and the data that it uses.
- This memory area is called an implicit cursor.
- Because this memory area is automatically managed by the Oracle server, you have no direct control over it.
- However, you can use predefined PL/SQL variables, called implicit cursor attributes, to find out how many rows were processed by the SQL statement.

Implicit and Explicit Cursors

There are two types of cursors:

- Implicit cursors: Defined automatically by Oracle for all SQL data manipulation statements, and for queries that return only one row.
 - An implicit cursor is always automatically named “SQL.”
- Explicit cursors: Defined by the PL/SQL programmer for queries that return more than one row.



Cursor Attributes for Implicit Cursors

- Cursor attributes are automatically declared variables that allow you to evaluate what happened when a cursor was last used.
- Attributes for implicit cursors are prefaced with “SQL.”
- Use these attributes in PL/SQL statements, but not in SQL statements.
- Using cursor attributes, you can test the outcome of your SQL statements.



Cursor Attributes for Implicit Cursors

Attribute	Description
<code>SQL%FOUND</code>	Boolean attribute that evaluates to <code>TRUE</code> if the most recent SQL statement returned at least one row.
<code>SQL%NOTFOUND</code>	Boolean attribute that evaluates to <code>TRUE</code> if the most recent SQL statement did not return even one row.
<code>SQL%ROWCOUNT</code>	An integer value that represents the number of rows affected by the most recent SQL statement.

Using Implicit Cursor Attributes: Example 1

- Delete rows that have the specified employee ID from the `copy_emp` table.
- Print the number of rows deleted.

```
DECLARE
  v_deptno copy_emp.department_id%TYPE := 50;
BEGIN
  DELETE FROM copy_emp
    WHERE department_id = v_deptno;
  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' rows deleted.');
```

```
END;
```

Using Implicit Cursor Attributes: Example 2

- Update several rows in the COPY_EMP table.
- Print the number of rows updated.

```
DECLARE
  v_sal_increase    employees.salary%TYPE := 800;
BEGIN
  UPDATE            copy_emp
    SET             salary = salary + v_sal_increase
    WHERE           job_id = 'ST_CLERK';

  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' rows updated.');
```



```
END;
```

Using Implicit Cursor Attributes: Good Practice Guideline

- Look at this code which creates a table and then executes a PL/SQL block.
- Determine what value is inserted into RESULTS.

```
CREATE TABLE results (num_rows NUMBER(4));

BEGIN
  UPDATE      copy_emp
    SET       salary = salary + 100
    WHERE     job_id = 'ST_CLERK';
  INSERT INTO results (num_rows)
    VALUES  (SQL%ROWCOUNT);
END;
```

Terminology

Key terms used in this lesson included:

- INSERT
- UPDATE
- DELETE
- MERGE
- Explicit cursors
- Implicit cursors

Summary

In this lesson, you should have learned how to:

- Construct and execute PL/SQL statements that manipulate data with DML statements
- Describe when to use implicit or explicit cursors in PL/SQL
- Create PL/SQL code to use SQL implicit cursor attributes to evaluate cursor activity

