5

Using SQL Statements in PLSQL Blocks

Course Road Map

Lesson 1: Course Overview

Unit 1: Introducing PL/SQL

Unit 2: Programming with PL/SQL

Unit 3: Working with PL/SQL Code

Lesson 2: PL/SQL Overview

Lesson 3: Declaring PL/SQL Variables

Lesson 4: Writing Executable Statements

Lesson 5: Using SQL Statements in PLSQL Programs

You are here!

Objectives

After completing this lesson, you should be able to do the following:

- Determine the SQL statements that can be directly included in a PL/SQL executable block
- Make use of the `INTO` clause to hold the values returned by a SQL statement
- Manipulate data with DML statements in PL/SQL
- Use transaction control statements in PL/SQL
- Differentiate between implicit cursors and explicit cursors

Agenda

- SQL statements in PL/SQL blocks
- Manipulating data with PL/SQL
- Introducing SQL cursors

SQL Statements in PL/SQL

- SQL consists of three sub-languages:
  - DDL
  - DML
- DDL statements are generally not included in PL/SQL blocks.
- DML statements are included in PL/SQL blocks.

SQL Statements in PL/SQL

- `SELECT` command to retrieve data into the PL/SQL block.
- DML commands to make changes to the database.
- `COMMIT`, `ROLLBACK`, or `SAVEPOINT` commands for transaction control.

SELECT Statements in PL/SQL

- The INTO clause is required.
- Queries must return only one row.

```
DECLARE
 v_fname VARCHAR2(25);
BEGIN
 SELECT first_name INTO v_fname
 FROM employees WHERE employee_id=200;
 DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
END;
/
```

```
PL/SQL procedure successfully completed.

First name is :Jennifer
```

Retrieving Data in PL/SQL: Example

Retrieve `hire_date` and `salary` for the specified employee.

```
DECLARE
 v_emp_hiredate    employees.hire_date%TYPE;
 v_emp_salary      employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      v_emp_hiredate, v_emp_salary
  FROM      employees
  WHERE     employee_id = 100;
  DBMS_OUTPUT.PUT_LINE ('Hire date is :'|| v_emp_hiredate);
  DBMS_OUTPUT.PUT_LINE ('Salary is :'|| v_emp_salary);
END;
/
```

```
PL/SQL procedure successfully completed.

Hire date is :17-JUN-11
Salary is :24000
```

Retrieving Data in PL/SQL

Return the sum of salaries for all the employees in the specified department.

Example:

```
DECLARE
   v_sum_sal    NUMBER(10,2);
   v_deptno     NUMBER NOT NULL := 60;
BEGIN
  SELECT  SUM(salary)  -- group function
  INTO v_sum_sal  FROM employees
  WHERE      department_id = v_deptno;
  DBMS_OUTPUT.PUT_LINE ('The sum of salary is ' || v_sum_sal);
END;
```

```
PL/SQL procedure successfully completed.

The sum of salary is :28800
```

Naming Ambiguities

```
DECLARE
  hire_date      employees.hire_date%TYPE;
  sysdate        hire_date%TYPE;
  employee_id    employees.employee_id%TYPE := 176;
BEGIN
  SELECT      hire_date, sysdate
  INTO        hire_date, sysdate
  FROM        employees
  WHERE       employee_id = employee_id;
END;
/
```

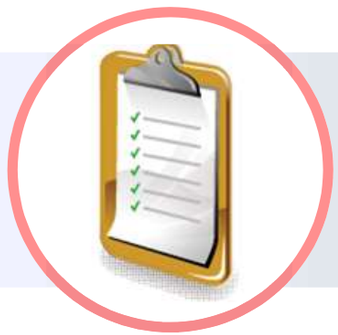
```
Error report -
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
01422. 00000 -  "exact fetch returns more than requested number of rows"
*Cause:    The number specified in exact fetch is less than the rows returned.
*Action:   Rewrite the query or change number of rows requested
```

Avoiding Naming Ambiguities

- Use a naming convention to avoid ambiguity in the `WHERE` clause.
- Avoid using database column names as identifiers.
- The names of local variables and formal parameters take precedence over the names of database tables.
- The names of database table columns take precedence over the names of local variables.
- The names of variables take precedence over the function names.

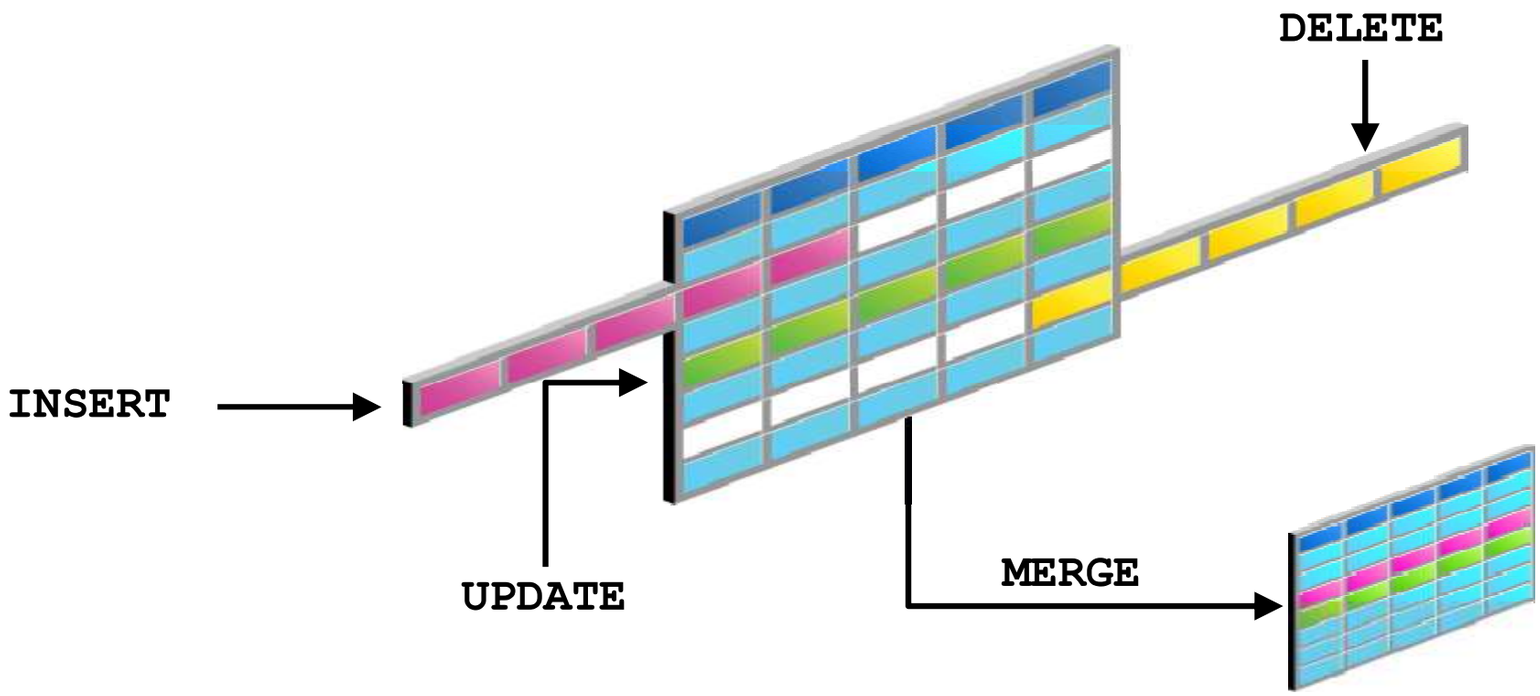
Agenda

- SQL statements in PL/SQL blocks
- Manipulating data with PL/SQL
- Introducing SQL cursors



Make changes to database tables by using DML commands:

- INSERT
- UPDATE
- DELETE
- MERGE



Insert Data: Example

Add new employee information to the EMPLOYEES table.

```
BEGIN
 INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
   VALUES(employees_seq.NEXTVAL, 'Ruth', 'Cores',
   'RCORES',CURRENT_DATE, 'AD_ASST', 4000);
END;
/
```

Update Data: Example

Increase the salary of all employees who are stock clerks.

```
SELECT first_name,salary
FROM employees
WHERE job_id = 'ST_CLERK';
```

Script Output | Query Result | All Rows Fetched: 20 in 0.001 seconds

	FIRST_NAME	SALARY
1	Julia	3200
2	Irene	2700
3	James	2400
4	Steven	2200
5	Laura	3300

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

```
PL/SQL procedure successfully completed.
```

```
SELECT first_name,salary
FROM employees
WHERE job_id = 'ST_CLERK';
```

Script Output | Query Result | All Rows Fetched: 20 in 0.003 seconds

	FIRST_NAME	SALARY
1	Julia	4000
2	Irene	3500
3	James	3200
4	Steven	3000
5	Laura	4100

Delete Data: Example

Delete rows that belong to department 10 from the `employees` table.

```
DECLARE
  v_emp   employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM   employees
  WHERE  employee_id = v_emp;
END;
/
```

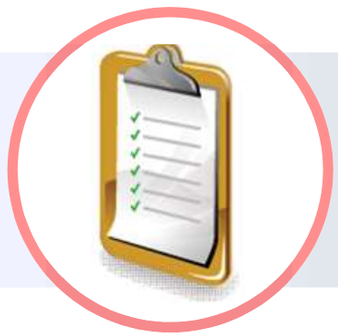
Merging Rows

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.empno)
   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.last_name,
          . . .,e.department_id);
END;
/
```

Agenda

- SQL statements in PL/SQL blocks
- Manipulating data with PL/SQL
- Introducing SQL cursors



SQL Cursor

- A cursor is a pointer to the private memory area that stores information about processing a specific `SELECT` or DML statement.
- There are two types of cursors:
  - **Implicit:** Created and managed by PL/SQL
  - **Explicit:** Created and managed explicitly by the programmer

Implicit Cursor

Explicit Cursor

SQL Cursor Attributes for Implicit Cursors

Using SQL cursor attributes, you can test the outcome of your SQL statements.

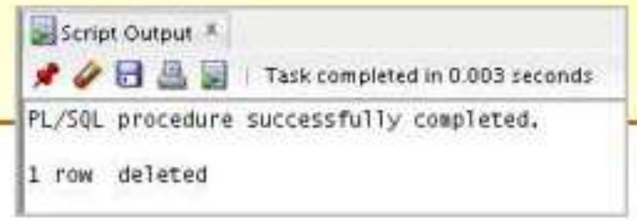
SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement affected at least one row
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement did not affect even one row
SQL%ROWCOUNT	An integer value that represents the number of rows affected by the most recent SQL statement

SQL Cursor Attributes for Implicit Cursors

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

Example:

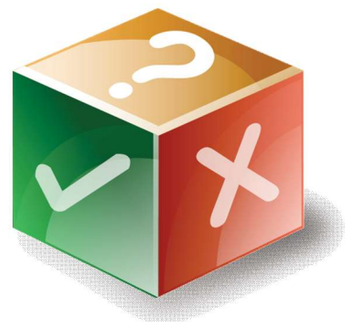
```
DECLARE
  v_rows_deleted VARCHAR2(30);
  v_empno employees.employee_id%TYPE := 165;
BEGIN
  DELETE FROM  employees
  WHERE employee_id = v_empno;
  v_rows_deleted := (SQL%ROWCOUNT ||
                     ' row deleted.');
  DBMS_OUTPUT.PUT_LINE (v_rows_deleted);

END;
```



When using the `SELECT` statement in PL/SQL, the `INTO` clause is required and queries can return one or more rows.

a. True

b. False



Summary

In this lesson, you should have learned how to:

- Embed DML statements, transaction control statements, and DDL statements in PL/SQL
- Use the `INTO` clause, which is mandatory for all `SELECT` statements in PL/SQL
- Differentiate between implicit cursors and explicit cursors
- Use SQL cursor attributes to determine the outcome of SQL statements

Practice 5: Overview

This practice covers the following topics:

- Selecting data from a table
- Inserting data into a table
- Updating data in a table
- Deleting a record from a table