

4

Interacting with the Oracle Server

Objectives

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

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

SQL Statements in PL/SQL

- Retrieve a row from the database by using the `SELECT` command.
- Make changes to rows in the database by using DML commands.
- Control a transaction with the `COMMIT`, `ROLLBACK`, or `SAVEPOINT` command.

SELECT Statements in PL/SQL

Retrieve data from the database with a `SELECT` statement.

Syntax:

```
SELECT  select_list
INTO    {variable_name[, variable_name]...
        | record_name}
FROM    table
[WHERE  condition];
```

SELECT Statements in PL/SQL

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

Example

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

Retrieving Data in PL/SQL

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

Example:

```
DECLARE
  emp_hiredate    employees.hire_date%TYPE;
  emp_salary      employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      emp_hiredate, emp_salary
  FROM      employees
  WHERE     employee_id = 100;
END;
/
```

Retrieving Data in PL/SQL

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

Example:

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

Naming Conventions

```
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

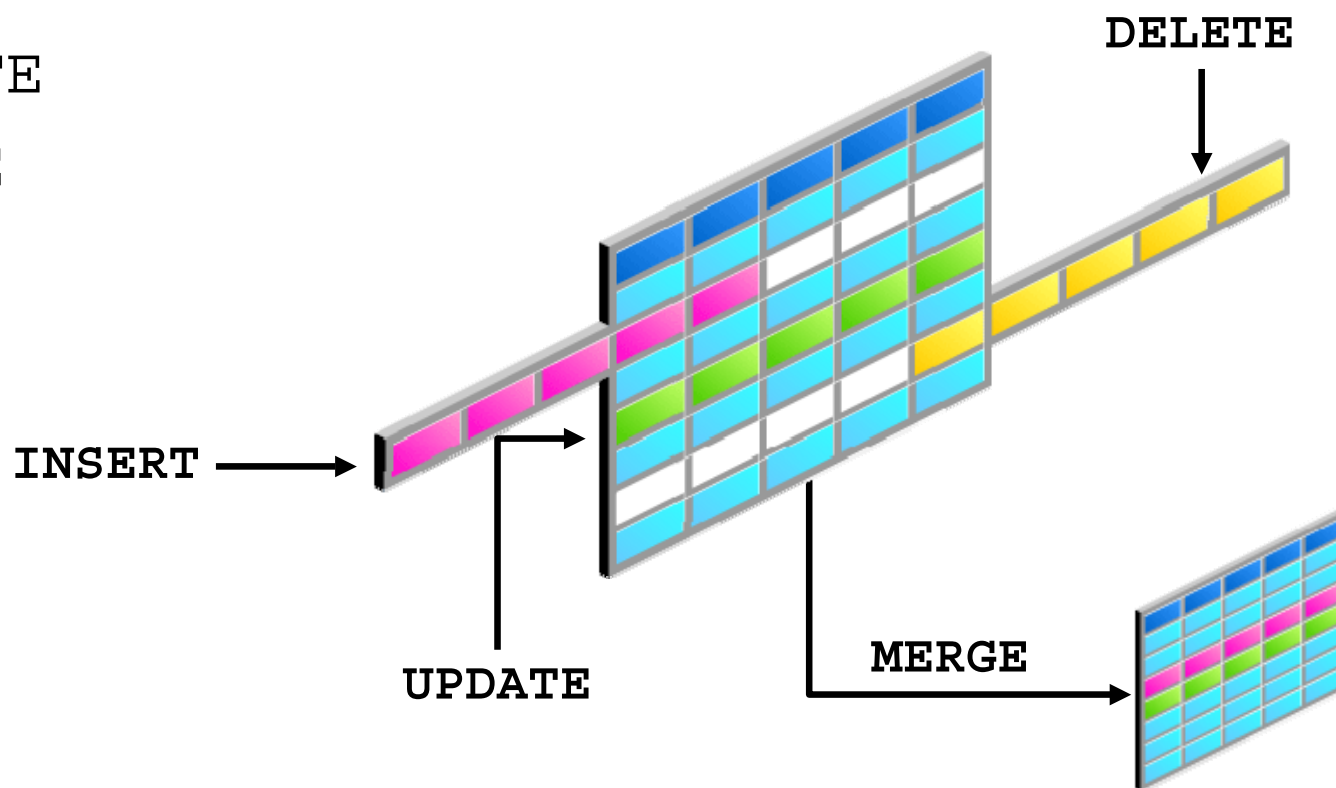
Naming Conventions

- Use a naming convention to avoid ambiguity in the `WHERE` clause.
- Avoid using database column names as identifiers.
- Syntax errors can arise because PL/SQL checks the database first for a column in the table.
- 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.

Manipulating Data Using PL/SQL

Make changes to database tables by using DML commands:

- INSERT
- UPDATE
- DELETE
- MERGE



Inserting Data

Add new employee information to the EMPLOYEES table.

Example:

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

Updating Data

Increase the salary of all employees who are stock clerks.

Example:

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

Deleting Data

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

Example:

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

Merging Rows

Insert or update rows in the `copy_emp` table to match the `employees` table.

```
DECLARE
    empno employees.employee_id%TYPE := 100;
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;
/
```

SQL Cursor

- A cursor is a pointer to the private memory area allocated by the Oracle server.
- There are two types of cursors:
 - Implicit: Created and managed internally by the Oracle server to process SQL statements
 - Explicit: Explicitly declared by the programmer

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 <code>TRUE</code> if the most recent SQL statement returned at least one row
SQL%NOTFOUND	Boolean attribute that evaluates to <code>TRUE</code> if the most recent SQL statement did not return 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

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
    empno employees.employee_id%TYPE := 176;
BEGIN
    DELETE FROM employees
    WHERE employee_id = empno;
    :rows_deleted := (SQL%ROWCOUNT ||
                     ' row deleted.');
```

```
END;
/
PRINT rows_deleted
```

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 4: 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