ORACLE®
ACADEMY

Database Programming with PL/SQL

3-2
Retrieving Data in PL/SQL

Objectives

This lesson covers the following objectives:

- Recognize the SQL statements that can be directly included in a PL/SQL executable block
- Construct and execute an `INTO` clause to hold the values returned by a single-row SQL SELECT statement
- Construct statements to retrieve data that follow good practice guidelines
- Construct statements that apply good practice guidelines for naming variables

Purpose

- In this lesson, you learn to embed standard SQL SELECT statements in PL/SQL blocks.
- You also learn the importance of following usage guidelines and naming convention guidelines when retrieving data.
- Blocks can be a good method for organizing your code.
- When you review code written by someone else, it is easier to read chunks of a program than it is to read one long continuous program.

SQL Statements in PL/SQL

You can use the following kinds of SQL statements in PL/SQL:

- `SELECT` statements to retrieve data from a database.
- DML statements, such as `INSERT`, `UPDATE`, and `DELETE`, to make changes to the database.
- Transaction control statements, such as `COMMIT`, `ROLLBACK`, or `SAVEPOINT`, to make changes to the database permanent or to discard them.
- Transaction control statements will be covered later and are not available in the iAcademy-hosted APEX environment.

DDL/DCL Limitations in PL/SQL

- You cannot use DDL and DCL directly in PL/SQL.

Handle Style	Description
DDL	**`CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`**
DCL	**`GRANT`, `REVOKE`**

- PL/SQL does not directly support DDL statements, such as `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE`, or DCL statements such as `GRANT` and `REVOKE`.

DDL/DCL Limitations in PL/SQL

- You cannot directly execute DDL and DCL statements because they are constructed and executed at run time—that is, they are dynamic.
- There are times when you may need to run DDL or DCL within PL/SQL.
- The recommended way of working with DDL and DCL within PL/SQL is to use Dynamic SQL with the EXECUTE IMMEDIATE statement.
- This will be discussed later in the course.

SELECT Statements in PL/SQL

Retrieve data from a database into a PL/SQL variable with a SELECT statement so you can work with the data within PL/SQL.

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

Using the INTO Clause

- The INTO clause is mandatory and occurs between the SELECT and FROM clauses.
- It is used to specify the names of PL/SQL variables that hold the values that SQL returns from the SELECT clause.

```
DECLARE
  v_emp_lname employees.last_name%TYPE;
BEGIN
  SELECT last_name
    INTO v_emp_lname
    FROM employees
    WHERE employee_id = 100;
  DBMS_OUTPUT.PUT_LINE('His last name is ' || v_emp_lname);
END;
```

Retrieving Data in PL/SQL Example

You must specify one variable for each item selected, and the order of the variables must correspond with the order of the items selected.

```
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('Hiredate: ' || v_emp_hiredate);
  DBMS_OUTPUT.PUT_LINE('Salary: '|| v_emp_salary);
END;
```

Retrieving Data in PL/SQL Embedded Rule

- SELECT statements within a PL/SQL block fall into the ANSI classification of embedded SQL for which the following rule applies: embedded queries must return exactly one row.
- A query that returns more than one row or no rows generates an error.

```
DECLARE
  v_salary employees.salary%TYPE;
BEGIN
  SELECT salary INTO v_salary
    FROM employees;
  DBMS_OUTPUT.PUT_LINE(' Salary is : ' || v_salary);
END;
```

```
ORA-01422:  exact fetch returns more than requested number of rows
```

Retrieving Data in PL/SQL Example

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

```
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('Dep #60 Salary Total: ' || v_sum_sal);
END;
```

Guidelines for Retrieving Data in PL/SQL

- The guidelines for retrieving data in PL/SQL are:
  - Terminate each SQL statement with a semicolon (`;`).
  - Every value retrieved must be stored in a variable using the `INTO` clause.
  - The WHERE clause is optional and can contain input variables, constants, literals, or PL/SQL expressions.
- However, you should fetch only one row and the usage of the WHERE clause is therefore needed in nearly all cases.
- Can you think of a case where it isn't needed?

Guidelines for Retrieving Data in PL/SQL

- Specify the same number of variables in the `INTO` clause as database columns in the `SELECT` clause.
- Be sure the columns and variables are in the same positional order and their data types are compatible.
- To insure data type compatibility between columns and variables, declare the receiving variables using %TYPE.

Guidelines for Naming Conventions

In potentially ambiguous SQL statements, the names of database columns take precedence over the names of local variables.

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

```
ORA-01422:  exact fetch returns more than requested
number of rows
```

This example raises an unhandled run-time exception because in the WHERE clause, the PL/SQL variable name is the same as that of the database column name in the employees table.

Guidelines for Naming Conventions Example

- What is deleted by the following PL/SQL block?

```
DECLARE
  last_name employees.last_name%TYPE := 'King';
BEGIN
  DELETE FROM emp_dup WHERE last_name = last_name;
END;
```

- Does it remove the row where the employee's last name is King?

Guidelines for Naming Conventions Details

Guidelines for naming conventions:

- Use a naming convention to avoid ambiguity in the WHERE clause.
- Avoid using database column names as identifiers.
- Errors can occur during execution 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* (in a PL/SQL statement).
- The names of database table *columns* take precedence over the names of local variables.

Summary

In this lesson, you should have learned how to:

- Recognize the SQL statements that can be directly included in a PL/SQL executable block
- Construct and execute an `INTO` clause to hold the values returned by a single-row SQL SELECT statement
- Construct statements to retrieve data that follow good practice guidelines
- Construct statements that apply good practice guidelines for naming variables

ORACLE® ACADEMY