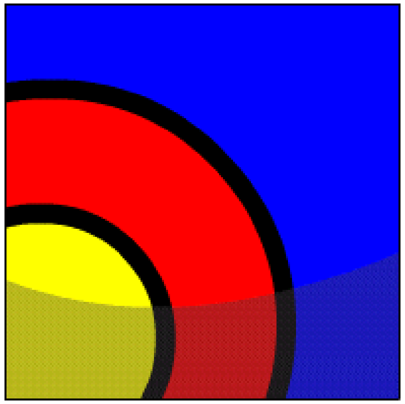
Retrieving Data in PL/SQL

What Will I Learn?

In this lesson, you will learn 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



Why Learn It?

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.

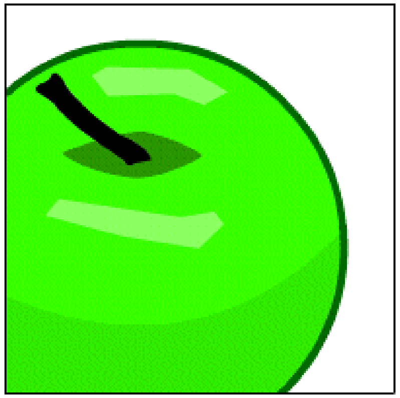
Tell Me/Show Me

SQL Statements in PL/SQL

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

- `SELECT` to retrieve data from the database.
- DML statements, such as `INSERT`, `UPDATE`, and `DELETE` to make changes to rows in the database.
- Transaction control statements, such as `COMMIT`, `ROLLBACK`, or `SAVEPOINT`. You use transaction control statements to make the changes to the database permanent or to discard them. Transaction control statements are covered later in the course.

This lesson covers `SELECT` statements.



Tell Me/Show Me

SQL Statements in PL/SQL (continued)

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

Statement Type	Examples
DDL	`CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`
DCL	`GRANT`, `REVOKE`

PL/SQL does not directly support data definition language (DDL) statements, such as `CREATE TABLE`, `ALTER TABLE`, or `DROP TABLE` and DCL statements such as `GRANT` and `REVOKE`.

You cannot directly execute DDL and DCL statements because they are constructed and executed at run time. That is, they are dynamic. Static SQL statements are statements that are fixed at the time a program is compiled.

Tell Me/Show Me

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];
```

Tell Me/Show Me

SELECT Statements in PL/SQL (continued)

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

```
DECLARE
  v_country_name wf_countries.country_name%TYPE;
BEGIN
  SELECT country_name INTO v_country_name
    FROM wf_countries  WHERE country_id= 359;
  DBMS_OUTPUT.PUT_LINE(' The country name is :
                     '||v_country_name);
END;
```

Tell Me/Show Me

Retrieving Data in PL/SQL

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

Example:

```
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 is: ' || v_emp_hiredate
                       || ' and Salary is: '
                       || v_emp_salary);
END;
```

Tell Me/Show Me

Retrieving Data in PL/SQL (continued)

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

```
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
```

Tell Me/Show Me

Retrieving Data in PL/SQL (continued)

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 ('The sum of salary is '
                        || v_sum_sal);
END;
```

Tell Me/Show Me

Guidelines for Retrieving Data in PL/SQL

- 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.
- Specify the same number of variables in the `INTO` clause as database columns in the `SELECT` clause. Be sure that they correspond positionally and that their data types are compatible.
- Declare the receiving variables using `%TYPE`.

Tell Me/Show Me

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 num
```

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.

Tell Me/Show Me

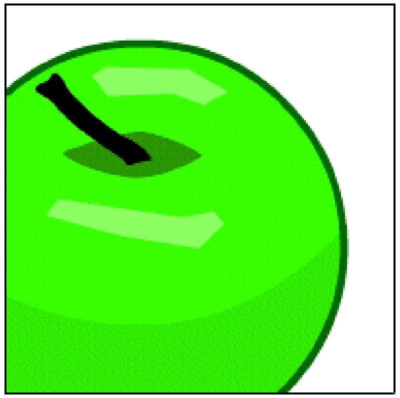
Guidelines for Naming Conventions (continued)

What is deleted in the following PL/SQL block?

```
DECLARE
  last_name VARCHAR2(25) := 'King';
BEGIN
  DELETE FROM emp_dup WHERE last_name = last_name;
END;
```

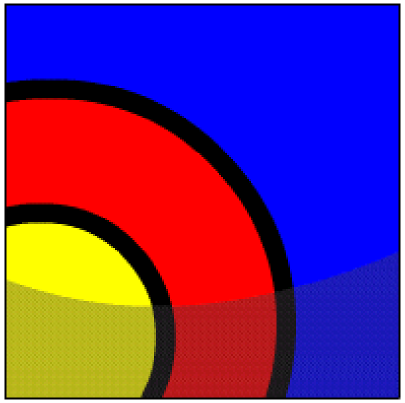
Tell Me/Show Me

Guidelines for Naming Conventions (continued)



- 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 learned 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

Try It/Solve It

The exercises in this lesson cover the following topics:

- Recognizing SQL statements that can be directly included in a PL/SQL executable block
- Using the `INTO` clause to hold the values returned by a single-row SQL SELECT statement
- Following guidelines for retrieving data
- Following guidelines for naming variables