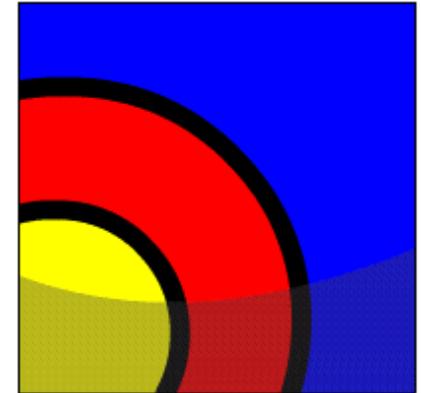


Conditional Control: CASE Statements

What Will I Learn?

In this lesson, you will learn to:

- Construct and use `CASE` statements in PL/SQL
- Construct and use `CASE` expressions in PL/SQL
- Include the correct syntax to handle null conditions in PL/SQL `CASE` statements
- Include the correct syntax to handle Boolean conditions in PL/SQL `IF` and `CASE` statements



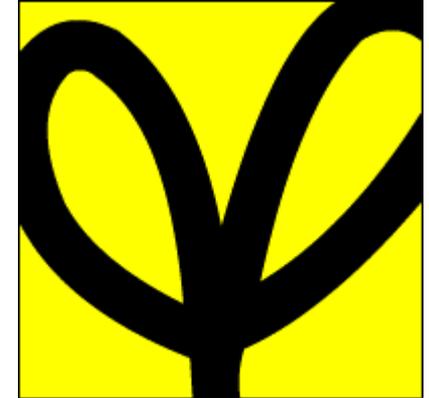


Why Learn It?

In this lesson, you learn how to use CASE statements and CASE expressions in a PL/SQL block.

CASE statements are similar to IF statements, but are often easier to write and easier to read.

CASE expressions are functions that return one of a number of values into a variable.





Tell Me/Show Me

Using a CASE Statement

Look at this IF statement. What do you notice?

```
DECLARE
    v_numvar      NUMBER;
BEGIN
    ...
    IF      v_numvar = 5  THEN statement_1; statement_2;
    ELSIF  v_numvar = 10 THEN statement_3;
    ELSIF  v_numvar = 12 THEN statement_4; statement_5;
    ELSIF  v_numvar = 27 THEN statement_6;
    ELSIF  v_numvar ... - and so on
    ELSE   statement_15;
    END IF;
    ...
END;
```

All the conditions test the same variable `v_numvar`. And the coding is very repetitive: `v_numvar` is coded many times.



Tell Me/Show Me

Using a CASE Statement (continued)

Here is the same logic, but using a CASE statement:

```
DECLARE
    v_numvar    NUMBER;
BEGIN
    ...
    CASE v_numvar
        WHEN 5 THEN statement_1; statement_2;
        WHEN 10 THEN statement_3;
        WHEN 12 THEN statement_4; statement_5;
        WHEN 27 THEN statement_6;
        WHEN ... - and so on
        ELSE statement_15;
    END CASE;
    ...
END;
```

It's much neater, isn't it? `v_numvar` is referenced only once. Easier to write, and easier to read.



Tell Me/Show Me

CASE Statements: A Second Example

```
DECLARE
  v_deptid    departments.department_id%TYPE;
  v_deptname  departments.department_name%TYPE;
  v_emps      NUMBER;
  v_mngid     departments.manager_id%TYPE := 108;
BEGIN
  CASE v_mngid
    WHEN 108 THEN
      SELECT department_id, department_name
         INTO v_deptid, v_deptname FROM departments
        WHERE manager_id=108;
      SELECT count(*) INTO v_emps FROM employees
        WHERE department_id=v_deptid;
    WHEN 200 THEN
      ...
  END CASE;
  DBMS_OUTPUT.PUT_LINE ('You are working in the ' || v_deptname |
  ' department. There are ' || v_emps || ' employees in this
  department');
END;
```

Tell Me/Show Me

Using a CASE Expression

You want to assign a value to one variable that depends on the value in another variable. Look at this IF statement:

```
DECLARE
  v_out_var  VARCHAR2(15);
  v_in_var   NUMBER;
BEGIN
  ...
  IF      v_in_var = 1  THEN v_out_var := 'Low value';
  ELSIF  v_in_var = 50 THEN v_out_var := 'Middle value';
  ELSIF  v_in_var = 99 THEN v_out_var := 'High value';
  ELSE
          v_out_var := 'Other value';
  END IF;
  ...
END;
```

Again, the coding is very repetitive.



Tell Me/Show Me

Using a CASE Expression (continued)

Here is the same logic, but using a CASE expression:

```
DECLARE
  v_out_var   VARCHAR2(15);
  v_in_var    NUMBER;
BEGIN
  ...
  v_out_var :=
    CASE v_in_var
      WHEN 1   THEN 'Low value'
      WHEN 50  THEN 'Middle value'
      WHEN 99  THEN 'High value'
      ELSE     'Other value'
    END;
  ...
END;
```

Again, it is much neater than the equivalent IF statement.



Tell Me/Show Me

CASE Expressions

A `CASE` expression selects one of a number of results and returns it into a variable.

In the syntax, *expressionN* can be a literal value, such as 50, or an expression, such as $(27+23)$ or (v_other_var*2) .

```
variable_name :=  
  CASE selector  
    WHEN expression1 THEN result1  
    WHEN expression2 THEN result2  
    ...  
    WHEN expressionN THEN resultN  
  [ELSE resultN+1]  
END;
```



Tell Me/Show Me

CASE Expressions: A Second Example

```
DECLARE
    v_grade      CHAR(1) := 'A';
    v_appraisal  VARCHAR2(20);
BEGIN
    v_appraisal :=
        CASE v_grade
            WHEN 'A' THEN 'Excellent'
            WHEN 'B' THEN 'Very Good'
            WHEN 'C' THEN 'Good'
            ELSE 'No such grade'
        END;
    DBMS_OUTPUT.PUT_LINE ('Grade: ' || v_grade ||
                          ' Appraisal ' || v_appraisal);
END;
```

Grade: A

Appraisal Excellent

Statement processed.



Tell Me/Show Me

CASE Expressions: A Third Example

What do you think will be displayed when this block is executed?

```
DECLARE
  v_out_var  VARCHAR2(15);
  v_in_var   NUMBER := 20;
BEGIN
  v_out_var :=
    CASE v_in_var
      WHEN 1          THEN 'Low value'
      WHEN v_in_var  THEN 'Same value'
      WHEN 20         THEN 'Middle value'
      ELSE            'Other value'
    END;
  DBMS_OUTPUT.PUT_LINE(v_out_var);
END;
```

Tell Me/Show Me

Searched CASE Expressions

PL/SQL also provides a searched CASE expression, which has the following form:

```
CASE
  WHEN search_condition1 THEN result1
  WHEN search_condition2 THEN result2
  ...
  WHEN search_conditionN THEN resultN
  [ELSE resultN+1]
END;
```

A searched CASE expression has no selector. Also, its WHEN clauses contain search conditions that yield a Boolean value, not expressions that can yield a value of any type.



Tell Me/Show Me

Searched CASE Expressions: An Example

```
DECLARE
  v_grade      CHAR(1) := 'A';
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal :=
    CASE -- no selector here
      WHEN v_grade = 'A' THEN 'Excellent'
      WHEN v_grade IN ('B','C') THEN 'Good'
      ELSE 'No such grade'
    END;
  DBMS_OUTPUT.PUT_LINE ('Grade: ' || v_grade ||
    ' Appraisal ' || v_appraisal);
END;
```



Tell Me/Show Me

How are CASE Expressions Different From CASE Statements?

- **CASE expressions** return a value into a variable.
- CASE expressions end with `END;`
- A CASE expression is a single PL/SQL statement.

```
DECLARE
  v_grade      CHAR(1) := 'A';
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal :=
    CASE
      WHEN v_grade = 'A' THEN 'Excellent'
      WHEN v_grade IN ('B','C') THEN 'Good'
      ELSE 'No such grade'
    END;
  DBMS_OUTPUT.PUT_LINE ('Grade: ' || v_grade ||
                        ' Appraisal ' || v_appraisal);
END;
```

Tell Me/Show Me

How are CASE Expressions Different From CASE Statements? (continued)

- CASE statements evaluate conditions and perform actions.
- A CASE statement can contain many PL/SQL statements.
- CASE statements end with `END CASE ;`.

```
DECLARE
    v_grade CHAR(1) := 'A';
BEGIN
    CASE
        WHEN v_grade = 'A' THEN
            DBMS_OUTPUT.PUT_LINE ('Excellent');
        WHEN v_grade IN ('B','C') THEN
            DBMS_OUTPUT.PUT_LINE ('Good');
        ELSE
            DBMS_OUTPUT.PUT_LINE('No such grade');
    END CASE;
END;
```

Tell Me/Show Me

Logic Tables

When using IF and CASE statements you often need to combine conditions using AND, OR, and NOT. The following Logic Tables show the results of all possible combinations of two conditions.

AND	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	OR	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	NOT	
<i>TRUE</i>	TRUE	(1) FALSE	NULL	<i>TRUE</i>	TRUE	TRUE	TRUE	<i>TRUE</i>	FALSE
<i>FALSE</i>	FALSE	FALSE	FALSE	<i>FALSE</i>	TRUE	FALSE	NULL	<i>FALSE</i>	TRUE
<i>NULL</i>	NULL	FALSE	NULL	<i>NULL</i>	TRUE	NULL	NULL	<i>NULL</i>	NULL

Example: (1) TRUE AND FALSE is FALSE

 **Tell Me/Show Me****Boolean Conditions**

What is the value of `v_flag` in each case?

```
v_flag := v_reorder_flag AND v_available_flag;
```

V_REORDER_FLAG	V_AVAILABLE_FLAG	V_FLAG
TRUE	TRUE	?
TRUE	FALSE	?
NULL	TRUE	?
NULL	FALSE	?

Tell Me/Show Me

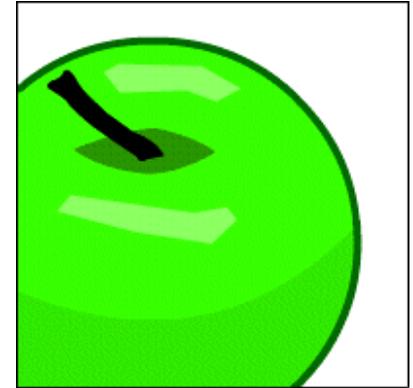
Terminology

Key terms used in this lesson include:

CASE expression

CASE statement

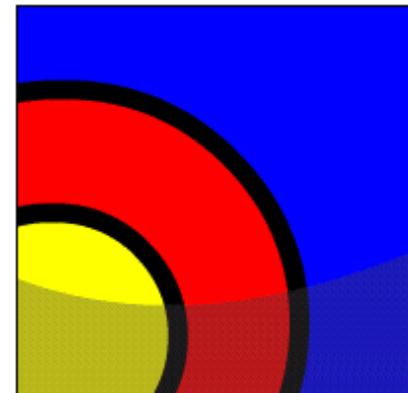
Logic Tables



Summary

In this lesson, you learned to:

- Construct and use `CASE` statements in PL/SQL
- Construct and use `CASE` expressions in PL/SQL
- Include the correct syntax to handle null conditions in PL/SQL `CASE` statements
- Include the correct syntax to handle Boolean conditions in PL/SQL `IF` and `CASE` statements



Try It/Solve It

The exercises in this lesson cover the following topics:

- Constructing and using `CASE` statements
- Constructing and using `CASE` expressions
- Handling null conditions in `CASE` statements
- Handling Boolean conditions in `IF` and `CASE` statements

