Conditional Control: Case Statements
Objectives

This lesson covers the following objectives:

• 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
Purpose

• 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 work like functions to return one value from a number of values into a variable.
Using a **CASE** Statement

• Look at this `IF` statement. What do you notice?
• All the conditions test the same variable `v_numvar`.
• And the coding is very repetitive: `v_numvar` is coded many times.

```sql
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;
```
Using a **CASE** Statement

- Here is the same logic, but using a **CASE** statement.
- It is much easier to read. `v_numvar` is written only once.

```plsql
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;
```
CASE Statements: An Example

A simple example to demonstrate the CASE logic.

```sql
DECLARE
  v_num NUMBER := 15;
  v_txt VARCHAR2(50);
BEGIN
  CASE v_num
    WHEN 20 THEN v_txt := 'number equals 20';
    WHEN 17 THEN v_txt := 'number equals 17';
    WHEN 15 THEN v_txt := 'number equals 15';
    WHEN 13 THEN v_txt := 'number equals 13';
    WHEN 10 THEN v_txt := 'number equals 10';
    ELSE v_txt := 'some other number';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_txt);
END;
```
Searched `CASE` Statements

• You can use `CASE` statements to test for non-equality conditions such as `<`, `>`, `>=`, etc.
• These are called searched `CASE` statements.
• The syntax is virtually identical to an equivalent `IF` statement.

```plsql
DECLARE
  v_num  NUMBER := 15;
  v_txt  VARCHAR2(50);
BEGIN
  CASE
    WHEN v_num > 20 THEN v_txt := 'greater than 20';
    WHEN v_num > 15 THEN v_txt := 'greater than 15';
    ELSE v_txt := 'less than 16';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_txt);
END;
```
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.
- Again, the coding is very repetitive.

```
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;
```
Using a `CASE` Expression

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

```sql
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;
```
## CASE Expression Syntax

- A **CASE** expression selects one of a number of results and assigns it to a variable.

- In the syntax, \( expressionN \) can be a literal value, such as 50, or an expression, such as \((27+23)\) or \((v\_other\_var*2)\).

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

What would be the result of this code if v_grade was initialized as "C" instead of "A."

```
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;
```

RESULT:
Grade: A
Appraisal: Excellent
Statement processed.
CASE Expression: A Second Example

Determine what will be displayed when this block is executed:

```plsql
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;
```
Searched `CASE` Expression Syntax

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

```plsql
variable_name := 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.
Searched \textbf{CASE} Expressions: An Example

Searched CASE expressions allow non-equality conditions, compound conditions, and different variables to be used in different WHEN clauses.

```sql
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;
```
How are **CASE** Expressions Different From **CASE** Statements?

They are different because:

- **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;
```
How are `CASE` Expressions Different From `CASE` Statements?

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

```plsql
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;
```
Logic Tables

• When using **IF** and **CASE** statements you often need to combine conditions using **AND**, **OR**, and **NOT**.

• The following Logic Table displays the results of all possible combinations of two conditions.

• Example: TRUE and FALSE is FALSE.

<table>
<thead>
<tr>
<th>AND</th>
<th>TRUE</th>
<th>FALSE</th>
<th>NULL</th>
<th>OR</th>
<th>TRUE</th>
<th>FALSE</th>
<th>NULL</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>NULL</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>NULL</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>FALSE</td>
<td>NULL</td>
<td>NULL</td>
<td>TRUE</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Boolean Conditions

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

\[ v\_flag := v\_reorder\_flag \text{ AND } v\_available\_flag; \]

<table>
<thead>
<tr>
<th>V_REORDER_FLAG</th>
<th>V_AVAILABLE_FLAG</th>
<th>V_FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>1. _____</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>2. _____</td>
</tr>
<tr>
<td>NULL</td>
<td>TRUE</td>
<td>3. _____</td>
</tr>
<tr>
<td>NULL</td>
<td>FALSE</td>
<td>4. _____</td>
</tr>
</tbody>
</table>
Terminology

Key terms used in this lesson included:

- **CASE** expression
- **CASE** statement
- Logic tables
Summary

In this lesson, you should have learned how 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