Oracle PL/SQL – Conditional Statements

Controlling Program Flow

- PL/SQL programs run sequentially from the top of the block to the bottom, unless acted upon by a control structure.

- PL/SQL provides the following functionalities with help of control structures

    - Ability to divert code execution in accordance with certain conditions.
    - Execute certain code repeatedly until a condition is met.
    - Jump to various sections of the block as needed.

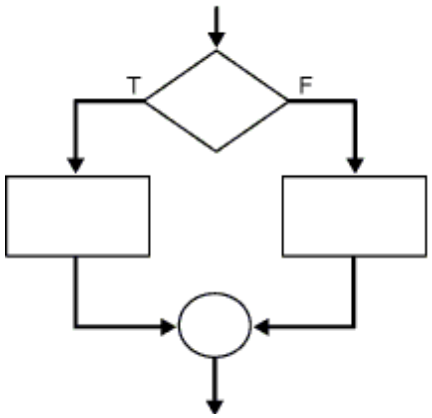
- The control structures are divided into three categories.

    - Conditional Statements
    - Iterative Statements
    - Sequential Statements

Conditional Statements

- It allows processing a portion of code depending on whether certain criteria are met or not.

- For example, IF the attained result for a module is pass THEN proceed with the next module ELSE revise the same module again.



Main types of conditional control structures are

- IF-THEN
- IF-THEN-ELSE
- IF-THEN-ELSIF
- CASE

1. IF-THEN

- It is the most basic conditional evaluation.
- If a condition is met, then do something and if not met, skip the code between Then and End IF keywords and continue with the rest of the program.
- The condition must evaluate to TRUE, FALSE or NULL.

Syntax:

```
IF condition
THEN
    action statements;
END IF;
```

Example: 1

The below example shows the PL/SQL code to update the salary of the employee with ID 101 by 10%, if the designation of that employee is 'ASE'

```
set serveroutput on;
DECLARE
       -- declare variables
       v_salary employee.salary%TYPE;
       v_designation employee.designation%TYPE;
BEGIN
     select designation into v_designation FROM employee where emp_id = 101;
     IF (v_designation = 'ASE') THEN
        UPDATE employee SET salary=salary+0.10*salary WHERE emp_id = 101;
     END IF;
END;
/
```

2. IF-THEN-ELSE

- It is an extension to IF-THEN statements.
- Allows specifying what to do if the condition evaluates to FALSE or NULL.

Syntax:

```
IF condition
THEN
     action statements;
ELSE
     action statements;
END IF;
```

Example: 2

This is an extension to Example 1. Here, if the designation of employee with ID 101 is not equal to 'ASE', then the salary is updated by 20%

```
set serveroutput on;
DECLARE
       -- declare variables
       v_salary employee.salary%TYPE;
       v_designation employee.designation%TYPE;
BEGIN
     select designation into v_designation FROM employee where emp_id = 101;
     IF (v_designation = 'ASE') THEN
        UPDATE employee SET salary=salary+0.10*salary WHERE emp_id = 101;
     ELSE
        UPDATE employee SET salary=salary+0.20*salary WHERE emp_id = 101;
     END IF;
END;
/
```

3. IF-THEN-ELSIF

- It allows choosing between several alternatives.
- It provides a way to chain IF conditions together and if one is met, the rest are skipped.
- It can have an optional ELSE after ELSIF which would get executed if all the rest of the conditions are not met.

Syntax:

```
IF condition1
THEN
     action statements1;
ELSIF condition2
THEN
     action statements2;
ELSE
     action statements3;
END IF;
```

Note:

- The term is ELSIF and not ELSEIF
- An IF-THEN statement can have zero or one ELSE and it must come after all ELSIF.
- An IF-THEN statement can have zero to many ELSIF and they must come before the ELSE.
- Once an ELSIF succeeds, none of the remaining ELSIF or ELSE will be tested.

Example: 3

In the below example, a condition is added to Example:2. If the designation of employee with ID 101 is 'ASE' then the salary is incremented by 10%.If not 'ASE' and if it is 'ITA' then the increment is 15% and if both conditions are not true then the increment is 20%. Here, if the designation is 'ASE' then the other two conditions are not evaluated

```
set serveroutput on;
DECLARE
       -- declare variables
       v_salary employee.salary%TYPE;
       v_designation employee.designation%TYPE;
BEGIN
     select designation into v_designation FROM employee where emp_id = 101;
     IF (v_designation = 'ASE') THEN
        UPDATE employee SET salary=salary+0.10*salary WHERE emp_id = 101;
     ELSIF (v_designation = 'ITA') THEN
        UPDATE employee SET salary=salary+0.15*salary WHERE emp_id = 101;
     ELSE
        UPDATE employee SET salary=salary+0.20*salary WHERE emp_id = 101;
     END IF;
END;
/
```

4. CASE statement

- Similar to IF-THEN-ELSE statements, CASE expression selects a result from one or more alternatives.
- The CASE expression uses a selector, an expression whose value determines which alternative to return.
- The selector is followed by one or more WHEN clauses, which are checked

sequentially.

- If the first WHEN clause matches the value of the selector, then the remaining WHEN clauses are not evaluated.
- Optional ELSE clause is executed if the value of the selector does not match with any of the WHEN clauses.

Syntax:

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

Example: 4

In the given example, the grade value is checked with each WHEN clause value and the value returned by the matched WHEN clause will be assigned to the 'appraisal' variable.

```
DECLARE
   grade CHAR(1) := 'B';
   appraisal VARCHAR2(20);
BEGIN
   appraisal :=
      CASE grade
         WHEN 'A' THEN 'Excellent'
         WHEN 'B' THEN 'Very Good'
         WHEN 'C' THEN 'Good'
         WHEN 'D' THEN 'Fair'
         WHEN 'F' THEN 'Poor'
         ELSE 'No such grade'
         END;
    dbms_output.put_line('Appraisal:' ||appraisal);
END;
/
```

Output:

```
Appraisal:Very Good

PL/SQL procedure successfully completed
```

5. Searched CASE

- Searched CASE statement is similar to CASE statement but has no selector
- WHEN clauses contain search conditions that yield Boolean values.

Syntax:

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

Example: 5

In the given example, the appraisal is derived based on the total_marks value. Each WHEN clause expression is evaluated one by one. Once the expression returns TRUE, it skips the evaluation of rest of the WHEN clauses. Here, the total_marks is assigned the value 78 and it satisfies the third WHEN clause expression (between 70 and 80).Therefore, ‘Good’ value is returned and assigned to appraisal variable.

```
DECLARE
   total_marks NUMBER(5):=78;
   appraisal VARCHAR2(20);
BEGIN
   appraisal :=
      CASE
         WHEN total_marks >90 THEN 'Excellent'
         WHEN total_marks BETWEEN 80 AND 90 THEN 'Very Good'
         WHEN total_marks BETWEEN 70 AND 80 THEN 'Good'
         WHEN total_marks BETWEEN 60 AND 70 THEN 'Fair'
         WHEN total_marks <60 THEN 'Poor'
         ELSE 'No such grade'
      END;
   dbms_output.put_line('Appraisal:' ||appraisal);
END;
/
```

Output:

```
Appraisal:Good

PL/SQL procedure successfully completed
```