



# Database Programming

## 2. Condition Control (IF, CASE)

Sagara Samarawickrama  
CSD 4203

## 2. Condition Control

In almost every program that you write, you need to make decisions. For example, if it is the end of the fiscal year, bonuses must be distributed to the employees based on their salaries. To compute employee bonuses, a program needs to have a conditional control. In other words, it needs to employ a selection structure.

Conditional control allows you to control the flow of the execution of the program based on a condition. In programming terms, it means that the statements in the program are not executed sequentially. Rather, one group of statements or another will be executed depending on how the condition is evaluated.

In PL/SQL, there are three types of conditional control: IF, ELSIF, and CASE statements.



## 2. Condition Control

### IF Statements

An IF statement has two forms: IF-THEN and IF-THEN-ELSE. An IF-THEN statement allows you to specify only one group of actions to take. In other words, this group of actions is taken only when a condition evaluates to TRUE. An IF-THEN-ELSE statement allows you to specify two groups of actions, and the second group of actions is taken when a condition evaluates to FALSE or NULL.

### IF-THEN Statements

An IF-THEN statement is the most basic kind of a conditional control and has the structure shown below

```
IF CONDITION  
THEN  
STATEMENT 1;  
...  
STATEMENT N;  
END IF;
```

## 2. Condition Control

### IF Statements

An IF statement has two forms: IF-THEN and IF-THEN-ELSE. An IF-THEN statement allows you to specify only one group of actions to take. In other words, this group of actions is taken only when a condition evaluates to TRUE. An IF-THEN-ELSE statement allows you to specify two groups of actions, and the second group of actions is taken when a condition evaluates to FALSE or NULL.

### IF-THEN Statements

An IF-THEN statement is the most basic kind of a conditional control and has the structure shown below

```
IF CONDITION  
THEN  
STATEMENT 1;  
...  
STATEMENT N;  
END IF;
```



## 2. Condition Control

The reserved word IF marks the beginning of the IF statement. Statements 1 through N are a sequence of executable statements that consist of one or more of the standard programming structures. The CONDITION between the keywords IF and THEN determines whether these statements are executed. Consider the following example

```
DECLARE
  v_num1 NUMBER := 5;
  v_num2 NUMBER := 3;
  v_temp NUMBER;
BEGIN
  -- if v_num1 is greater than v_num2 rearrange their values
  IF v_num1 > v_num2
  THEN
    v_temp := v_num1;
    v_num1 := v_num2;
    v_num2 := v_temp;
  END IF;

  -- display the values of v_num1 and v_num2
  DBMS_OUTPUT.PUT_LINE ('v_num1 = ' || v_num1);
  DBMS_OUTPUT.PUT_LINE ('v_num2 = ' || v_num2);
END;
```

## 2.Condition Control

### IF-THEN-ELSE Statement

An IF-THEN statement specifies the sequence of statements to execute only if the condition evaluates to TRUE. When this condition evaluates to FALSE or NULL, there is no special action to take except to proceed with execution of the program

An IF-THEN-ELSE statement enables you to specify two groups of statements. One group of statements is executed when the condition evaluates to TRUE. Another group of statements is executed when the condition evaluates to FALSE or NULL

```
IF CONDITION
THEN
    STATEMENT 1;
ELSE
    STATEMENT 2;
END IF;
STATEMENT 3;
```




## 2. Condition Control

### IF-THEN-ELSE Statement

The IF-THEN-ELSE construct should be used when trying to choose between two mutually exclusive actions. Consider the following example:

```
DECLARE
  v_num NUMBER := &sv_user_num;
BEGIN
  -- test if the number provided by the user is even
  IF MOD(v_num,2) = 0
  THEN
    DBMS_OUTPUT.PUT_LINE (v_num||' is even number');
  ELSE
    DBMS_OUTPUT.PUT_LINE (v_num||' is odd number');
  END IF;
END;
```



```
IF CONDITION 1
THEN
  STATEMENT 1;
ELSIF CONDITION 2
THEN
  STATEMENT 2;
ELSIF CONDITION 3
THEN
  STATEMENT 3;
...
ELSE
  STATEMENT N;
END IF;
```

### ELSIF Statements

The reserved word IF marks the beginning of an ELSIF construct. The words CONDITION 1 through CONDITION N are a sequence of the conditions that evaluate to TRUE or FALSE. These conditions are mutually exclusive.

## 2.Condition Control

Consider the following example.

```
DECLARE
  v_num NUMBER := &sv_num;
BEGIN
  DBMS_OUTPUT.PUT_LINE ('Before IF statement...');
  IF v_num < 0
  THEN
    DBMS_OUTPUT.PUT_LINE (v_num||' is a negative number');
  ELSIF v_num = 0
  THEN
    DBMS_OUTPUT.PUT_LINE (v_num||' is equal to zero');
  ELSE
    DBMS_OUTPUT.PUT_LINE (v_num||' is a positive number');
  END IF;
  DBMS_OUTPUT.PUT_LINE ('After IF statement...');
END;
```

### Nested IF Statements

You have encountered different types of conditional controls: IF-THEN statement, IFTHEN-ELSE statement, and ELSIF statement. These types of conditional controls can be nested inside of one another—for example, an IF statement can be nested inside an ELSIF, and vice versa. Consider the example in the next slide



## 2.Condition Control

```
DECLARE
  v_num1  NUMBER := &sv_num1;
  v_num2  NUMBER := &sv_num2;
  v_total NUMBER;
BEGIN
  IF v_num1 > v_num2
  THEN
    DBMS_OUTPUT.PUT_LINE ('IF part of the outer IF');
    v_total := v_num1 - v_num2;
  ELSE
    DBMS_OUTPUT.PUT_LINE ('ELSE part of the outer IF');
    v_total := v_num1 + v_num2;

    IF v_total < 0
    THEN
      DBMS_OUTPUT.PUT_LINE ('Inner IF');
      v_total := v_total * (-1);
    END IF;
  END IF;
  DBMS_OUTPUT.PUT_LINE ('v_total = '||v_total);
END;
```

The IF-THEN-ELSE statement is called an outer IF statement because it encompasses the IF-THEN statement (shown in bold). The IF-THEN statement is called an inner IF statement because it is enclosed by the body of the IF-THEN-ELSE statement.

## 2. Condition Control

### CASE Statements

A CASE statement allows you to specify a selector that determines which group of actions to take. The structure of CASE statement is shown below.

```
CASE SELECTOR
  WHEN EXPRESSION 1 THEN STATEMENT 1;
  WHEN EXPRESSION 2 THEN STATEMENT 2;
  ...
  WHEN EXPRESSION N THEN STATEMENT N;
  ELSE STATEMENT N+1;
END CASE;
```

The reserved word CASE marks the beginning of the CASE statement. A selector is a value that determines which WHEN clause should be executed. Each WHEN clause contains an EXPRESSION and one or more executable statements associated with it. The ELSE clause is optional and works similar to the ELSE clause used in the IF-THEN-ELSE statement



## 2. Condition Control

Consider the following example.

```
DECLARE
  v_final_grade  NUMBER := &sv_final_grade;
  v_letter_grade CHAR(1);
BEGIN
  CASE
    WHEN v_final_grade >= 90
    THEN
      v_letter_grade := 'A';
    WHEN v_final_grade >= 80
    THEN
      v_letter_grade := 'B';
    WHEN v_final_grade >= 70
    THEN
      v_letter_grade := 'C';
    WHEN v_final_grade >= 60
    THEN
      v_letter_grade := 'D';
    ELSE
      v_letter_grade := 'F';
  END CASE;
  -- control resumes here
  DBMS_OUTPUT.PUT_LINE ('Final grade is: '||v_final_grade);
  DBMS_OUTPUT.PUT_LINE ('Letter grade is: '||v_letter_grade);
END;
```

If you change the order of the statements it may produce incorrect results.

## 2. Condition Control

### Case Expressions.

Consider the following example.

```
DECLARE
  v_num      NUMBER := &sv_user_num;
  v_num_flag NUMBER;
  v_result   VARCHAR2(30);
BEGIN
  v_num_flag := MOD(v_num,2);

  -- test if the number provided by the user is even
  v_result := CASE v_num_flag
                WHEN 0
                THEN
                  v_num||' is even number'
                ELSE
                  v_num||' is odd number'
                END;
  DBMS_OUTPUT.PUT_LINE (v_result);
END;
```

In this example, a new variable, v\_result, is used to hold the value returned by the CASE expression. If the variable v\_num is assigned the value of 8, this example produces the following output: *8 is even number*



## 2. Condition Control

### NULL IF Function.

The NULLIF function compares two expressions. If they are equal, then the function returns NULL; otherwise, it returns the value of the first expression.

#### **NULLIF (EXPRESSION 1, EXPRESSION 2)**

If EXPRESSION 1 is equal to EXPRESSION 2, the NULLIF function returns NULL. If EXPRESSION 1 does not equal EXPRESSION 2, the NULLIF function returns EXPRESSION 1.

```
DECLARE
  v_num      NUMBER := &sv_user_num;
  v_remainder NUMBER;
BEGIN
  -- calculate the remainder and if it is zero return NULL
  v_remainder := NULLIF(MOD(v_num, 2), 0);
  DBMS_OUTPUT.PUT_LINE ('v_remainder: ' || v_remainder);
END;
```

Here a value is assigned to the variable v\_num at run time. Next, this value is divided by 2, and its remainder is compared to 0 via the NULLIF function. If the remainder equals 0, the NULLIF function returns NULL; otherwise, it returns the remainder

