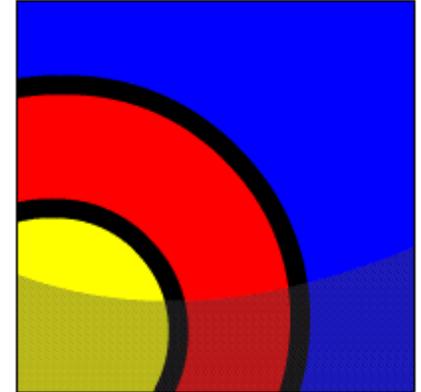


# Conditional Control: IF Statements

## What Will I Learn?

In this lesson, you will learn to:

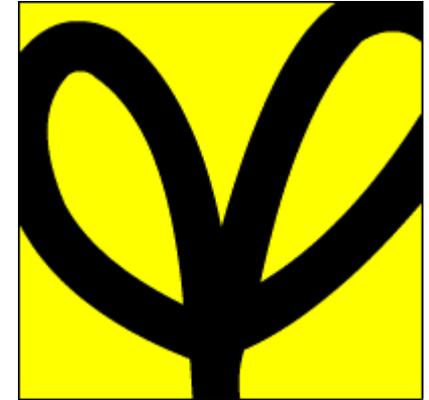
- Describe a use for conditional control structures
- List the types of conditional control structures
- Construct and use an `IF` statement
- Construct and use an `IF-THEN-ELSE` statement
- Create a PL/SQL to handle the null condition in `IF` statements





## Why Learn It?

In this section, you learn how to use the conditional logic in a PL/SQL block. Conditional processing extends the usefulness of programs by allowing the use of simple logical tests to determine which statements are executed.



## Tell Me/Show Me

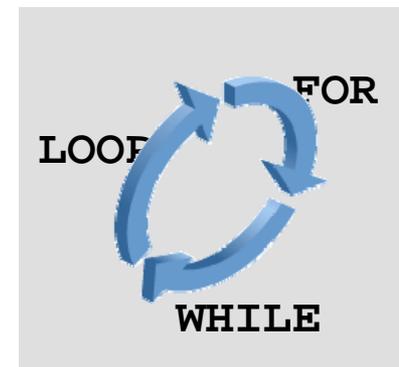
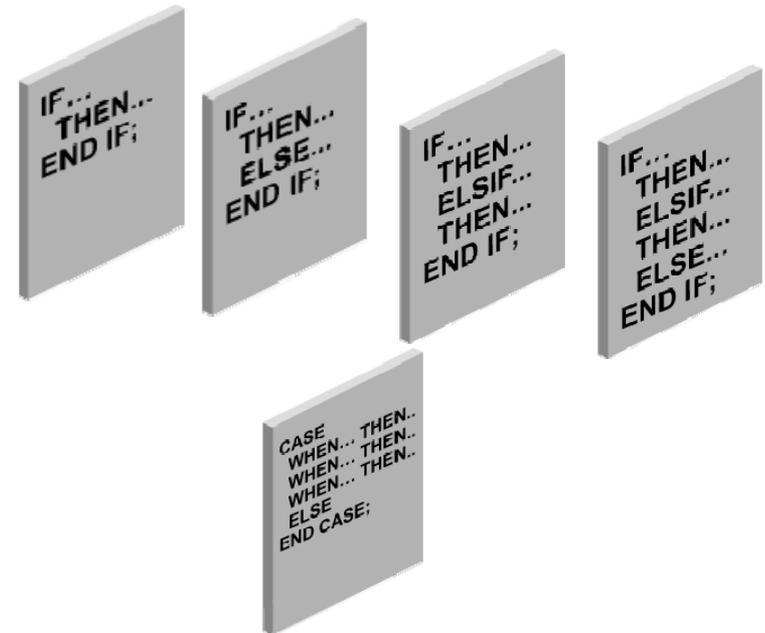
### Controlling the Flow of Execution

You can change the logical flow of statements within the PL/SQL block with a number of control structures.

This lesson introduces three types of PL/SQL control structures:

- Conditional constructs with the `IF` statement
- `CASE` expressions
- `LOOP` control structures

The `IF` statement is discussed in detail in this lesson. Later lessons discuss `CASE` and `LOOP` in detail.



## Tell Me/Show Me

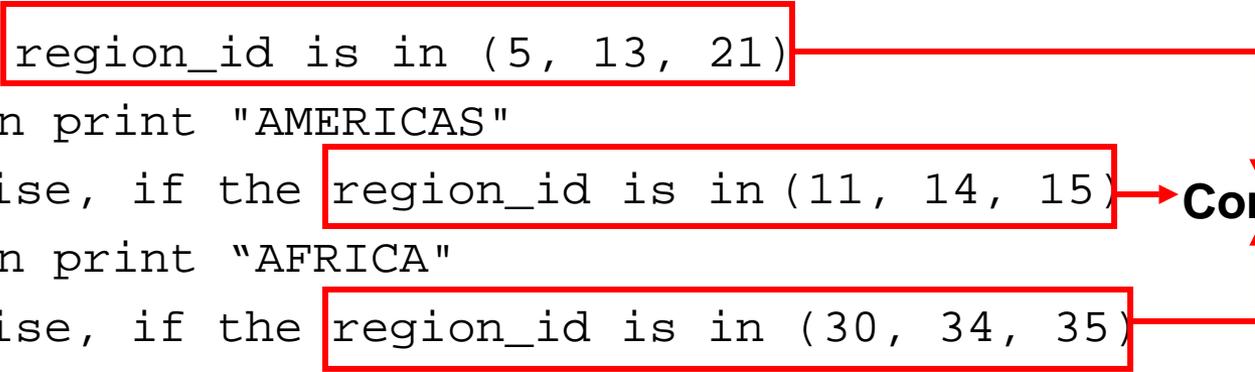
### IF statement

The IF statement contains alternative courses of action in a block based on conditions. A condition is an expression with a TRUE or FALSE value that is used to make a decision.

Consider the following example:

```
if the region_id is in (5, 13, 21)  
    then print "AMERICAS"  
otherwise, if the region_id is in (11, 14, 15)  
    then print "AFRICA"  
otherwise, if the region_id is in (30, 34, 35)  
    then print "ASIA"
```

**Conditions**



## Tell Me/Show Me

### CASE Expressions

CASE expressions are similar to IF statements in that they also determine a course of action based on conditions. They are different in that they can be used outside of a PLSQL block in an SQL statement. Consider the following example:

```
If the region_id is
    5 then print "AMERICAS"
    13 then print "AMERICAS"
    21 then print "AMERICAS"
    11 then print "AFRICA"
    14 then print "AFRICA"
    15 then print "AFRICA"
    30 then print "ASIA" ...
```



CASE expressions are discussed in the next lesson.

## Tell Me/Show Me

### LOOP control structures

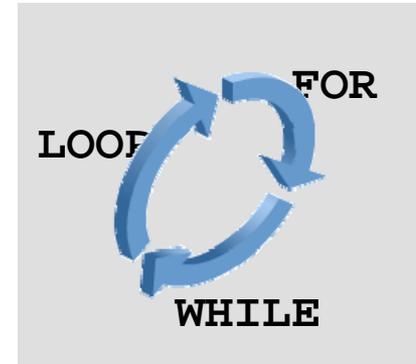
Loop control structures are repetition statements that enable you to execute statements in a PLSQL block repeatedly. There are three types of loop control structures supported by PL/SQL, **BASIC**, **FOR**, and **WHILE**. Consider the following example:

Print a list of numbers 1–5 by using a loop and a counter.

```
Loop Counter equals: 1
Loop Counter equals: 2
Loop Counter equals: 3
Loop Counter equals: 4
Loop Counter equals: 5
```

```
Statement processed.
```

Loops are discussed in later lessons.



## Tell Me/Show Me

### IF Statements

The structure of the PL/SQL `IF` statement is similar to the structure of `IF` statements in other procedural languages. It enables PL/SQL to perform actions selectively based on conditions.

Syntax:

```
IF condition THEN
    statements;
[ELSIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```



## Tell Me/Show Me

### IF Statements (continued)

- *condition* is a Boolean variable or expression that returns TRUE, FALSE, or NULL.
- THEN introduces a clause that associates the Boolean expression with the sequence of statements that follows it.
- *statements* can be one or more PL/SQL or SQL statements. (They can include further IF statements containing several nested IF, ELSE, and ELSIF statements.) The statements in the THEN clause are executed only if the condition in the associated IF clause evaluates to TRUE.

```
IF condition THEN
    statements;
[ELSIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```



## Tell Me/Show Me

### IF Statements (continued)

- ELSIF is a keyword that introduces a Boolean expression. (If the first condition yields FALSE or NULL, then the ELSIF keyword introduces additional conditions.)
- ELSE introduces the default clause that is executed if and only if none of the earlier conditions (introduced by IF and ELSIF) are TRUE. The tests are executed in sequence so that a later condition that might be true is pre-empted by an earlier condition that is true.
- END IF; marks the end of an IF statement.

```
IF condition THEN
    statements;
[ELSIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```



## Tell Me/Show Me

### IF Statements (continued)

**Note:** ELSIF and ELSE are optional in an IF statement. You can have any number of ELSIF keywords but only one ELSE keyword in your IF statement. END IF marks the end of an IF statement and must be terminated by a semicolon.

```
IF condition THEN
    statements;
[ELSIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```



## Tell Me/Show Me

### Simple IF Statement

The following is an example of a simple IF statement with a THEN clause. The `v_myage` variable is initialized to 31. The condition for the IF statement returns FALSE because `v_myage` is not less than 11. Therefore, the control never reaches the THEN clause and nothing is printed to the screen.

```
DECLARE
  v_myage NUMBER:=31;
BEGIN
  IF v_myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  END IF;
END;
```



## Tell Me/Show Me

### IF THEN ELSE Statement

The `ELSE` clause is added to the code in the previous slide. The condition has not changed and, therefore, it still evaluates to `FALSE`. Remember that the statements in the `THEN` clause are only executed if the condition returns `TRUE`. In this case, the condition returns `FALSE` and, therefore, the control moves to the `ELSE` statement.

```
DECLARE
  v_myage NUMBER:=31;
BEGIN
  IF v_myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  ELSE
    DBMS_OUTPUT.PUT_LINE(' I am not a child ');
  END IF;
END;
```



# Tell Me/Show Me

## IF ELSIF ELSE Clause

The IF statement now contains multiple ELSIF clauses and an ELSE clause. Notice that the ELSIF clauses add additional conditions. As with the IF, each ELSIF condition is followed by a THEN clause, which is executed if the condition returns TRUE.

```
DECLARE
    v_myage NUMBER:=31;
BEGIN
    IF v_myage < 11
        THEN
            DBMS_OUTPUT.PUT_LINE('I am a child');
    ELSIF v_myage < 20
        THEN
            DBMS_OUTPUT.PUT_LINE('I am young');
    ELSIF v_myage < 30
        THEN
            DBMS_OUTPUT.PUT_LINE('I am in my twenties');
    ELSIF v_myage < 40
        THEN
            DBMS_OUTPUT.PUT_LINE('I am in my thirties');
    ELSE
        DBMS_OUTPUT.PUT_LINE('I am always young ');
    END IF;
END;
```



## Tell Me/Show Me

### IF ELSIF ELSE Clause (continued)

When you have multiple clauses in the `IF` statement and a condition is `FALSE` or `NULL`, control then shifts to the next clause. Conditions are evaluated one by one from the top. `IF` all conditions are `FALSE` or `NULL`, then the statements in the `ELSE` clause are executed. The final `ELSE` clause is optional.

```
...IF      v_myage < 11 THEN
          DBMS_OUTPUT.PUT_LINE(' I am a child ');
ELSIF v_myage < 20 THEN
          DBMS_OUTPUT.PUT_LINE(' I am young ');
ELSIF v_myage < 30 THEN
          DBMS_OUTPUT.PUT_LINE(' I am in my twenties ');
ELSIF v_myage < 40 THEN
          DBMS_OUTPUT.PUT_LINE(' I am in my thirties ');
ELSE
          DBMS_OUTPUT.PUT_LINE(' I am always young ');
END IF;...
```



## Tell Me/Show Me

### IF Statement With Multiple Expressions

An IF statement can have multiple conditional expressions related with logical operators, such as AND, OR, and NOT.

For example:

```
DECLARE
  v_myage          NUMBER          := 31;
  v_myfirstname    VARCHAR2(11)    := 'Christopher';
BEGIN
  IF v_myfirstname = 'Christopher' AND v_myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child named Christopher');
  END IF;
END;
```

The condition uses the AND operator and, therefore, it evaluates to TRUE only if both the above conditions are evaluated as TRUE. There is no limitation on the number of conditional expressions; however, these statements must be connected with the appropriate logical operators.

## Tell Me/Show Me

### NULL Values in IF Statements

In this example, the `v_myage` variable is declared but is not initialized. The condition in the `IF` statement returns `NULL`, and not `TRUE` or `FALSE`. In such a case, the control goes to the `ELSE` statement because just like `FALSE`, `NULL` is not `TRUE`.

```
DECLARE
  v_myage NUMBER;
BEGIN
  IF v_myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  ELSE
    DBMS_OUTPUT.PUT_LINE(' I am not a child ');
  END IF;
END;
```



# Tell Me/Show Me

## Handling Nulls

When working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

- Simple comparisons involving nulls always yield `NULL`.
- Applying the logical operator `NOT` to a null yields `NULL`.
- In conditional control statements, if a condition yields `NULL`, it behaves just like a `FALSE`, and the associated sequence of statements is not executed.

## Tell Me/Show Me

### Handling Nulls (continued)

- Consider the following example:

```
x := 5;
```

```
y := NULL;
```

```
...
```

```
IF x != y THEN ... -- yields NULL, not TRUE and the  
sequence of statements are not executed
```

```
END IF;
```

- You can expect the sequence of statements to execute because `x` and `y` seem unequal. But, nulls are indeterminate. Whether `x` is equal to `y` is unknown. Therefore, the `IF` condition yields `NULL` and the sequence of statements is bypassed.

## Tell Me/Show Me

### Handling Nulls (continued)

- Consider the following example:

```
a := NULL;
```

```
b := NULL;
```

```
...
```

```
IF a = b THEN ... -- yields NULL, not TRUE and the  
sequence of statements are not executed
```

```
END IF;
```

- In the example, you can expect the sequence of statements to execute because a and b seem equal. But, again, equality is unknown. So the IF condition yields NULL and the sequence of statements is bypassed.

# Tell Me/Show Me

## Guidelines for Using IF Statements

- You can perform actions selectively when a specific condition is being met.
- When writing code, remember the spelling of the keywords:
  - `ELSIF` is one word.
  - `END IF` is two words.
- If the controlling Boolean condition is `TRUE`, then the associated sequence of statements is executed; if the controlling Boolean condition is `FALSE` or `NULL`, then the associated sequence of statements is passed over. Any number of `ELSIF` clauses is permitted.
- Indent the conditionally executed statements for clarity.

# Tell Me/Show Me

## Terminology

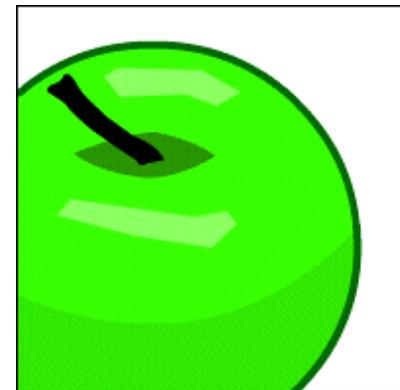
Key terms used in this lesson include:

IF

Condition

CASE

LOOP

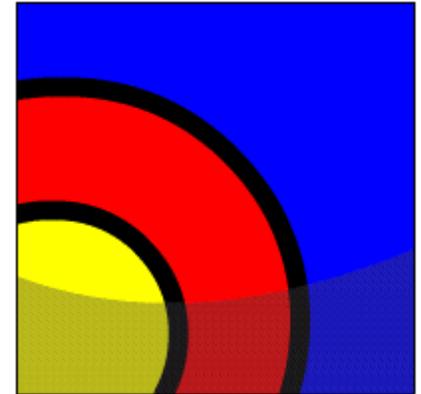




# Summary

In this lesson, you learned to:

- Describe a use for conditional control structures
- List the types of conditional control structures
- Construct and use an `IF` statement
- Construct and use an `IF-THEN-ELSE` statement
- Create a PL/SQL to handle the null condition in `IF` statements



## Try It/Solve It

The exercises in this lesson cover the following topics:

- Identifying the uses and types of conditional control structures
- Constructing and using an `IF` statement
- Constructing and using an `IF...THEN...ELSE` statement
- Handling the null condition in `IF` statements

