PL/SQL IF THEN ELSE Statements | Conditional Control

PL/SQL IF THEN ELSE conditional control statements. PL/SQL Conditional Control two type: IF THEN ELSE statement and [CASE statement,](http://www.way2tutorial.com/plsql/plsql_case_statement.php)

PL/SQL IF statement check condition and transfer the execution flow on that matched block depending on a condition. IF statement execute or skip a sequence of one or more statements. PL/SQL IF statement four different type,

1. [IF THEN Statement](http://www.way2tutorial.com/plsql/plsql_if_then_else_statements_conditional_control.php#ifthen)
2. [IF THEN ELSE Statement](http://www.way2tutorial.com/plsql/plsql_if_then_else_statements_conditional_control.php#ifthenelse)
3. [IF THEN ELSIF Statement](http://www.way2tutorial.com/plsql/plsql_if_then_else_statements_conditional_control.php#ifthenelsif)
4. [Nested IF THEN ELSE Statement](http://www.way2tutorial.com/plsql/plsql_if_then_else_statements_conditional_control.php#nestedifthenelse)

### IF THEN Statement

IF THEN Statement write in following syntax format:

IF ( condition ) THEN

statement

END IF;

#### **Example**

We declare one number with initialize 14 value is equal of condition value, Comparing 2 values by using IF THEN statement,

DECLARE

no INTEGER(2) := 14;

BEGIN

IF ( no = 14 ) THEN

DBMS\_OUTPUT.PUT\_LINE('condition true');

END IF;

END;

/

#### **Result**

condition true  
  
PL/SQL procedure successfully completed.

### IF THEN ELSE Statement

IF THEN ELSE Statement write in following syntax format:

IF ( condition ) THEN

statement;

ELSE

statement;

END IF;

#### **Example**

Same as above example if condition not true then else part will execute.

DECLARE

no INTEGER(2) := 14;

BEGIN

IF ( no = 11 ) THEN

DBMS\_OUTPUT.PUT\_LINE(no || ' is same');

ELSE

DBMS\_OUTPUT.PUT\_LINE(no || ' is not same');

END IF;

END;

/

#### **Result**

14 is not same  
  
PL/SQL procedure successfully completed.

### IF THEN ELSIF Statement

IF THEN ELSIF Statement write in following syntax format:

IF ( condition-1 ) THEN

statement-1;

ELSIF ( condition-2 ) THEN

statement-2;

ELSIF ( condition-3 ) THEN

statement-3;

ELSE

statement;

END IF;

Above syntax same as below syntax both are logically same

IF ( condition-1 ) THEN

statement-1;

ELSE

IF ( condition-2 ) THEN

statement-2;

ELSE

IF ( condition-3 ) THEN

statements-3;

END IF;

END IF;

END IF;

#### **Example**

Here one student result example for archiving grade.

DECLARE

result CHAR(20) := 'second';

BEGIN

IF ( result = 'distinction' ) THEN

DBMS\_OUTPUT.PUT\_LINE('First Class with Distinction');

ELSIF ( result = 'first' ) THEN

DBMS\_OUTPUT.PUT\_LINE('First Class');

ELSIF ( result = 'second' ) THEN

DBMS\_OUTPUT.PUT\_LINE('Second Class');

ELSIF ( result = 'third' ) THEN

DBMS\_OUTPUT.PUT\_LINE('Third Class');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Fail');

END IF;

END;

/

#### **Result**

Second Class  
  
PL/SQL procedure successfully completed.

### Nested IF THEN ELSE Statement

Logically IF THEN ELSIF Statement and Nested IF THEN ELSE Statement both are same. Nested IF THEN ELSE Statement write in following syntax format:

IF ( condition-1 ) THEN

statement-1;

ELSE

IF ( condition-2 ) THEN

statement-2;

ELSE

IF ( condition-3 ) THEN

statements-3;

END IF;

END IF;

END IF;

#### **Example**

Here check condition students gender male, if not male then finding the result using nested IF THEN ELSE statement.

DECLARE

gender CHAR(20) := 'female';

result CHAR(20) := 'second';

BEGIN

IF ( gender = 'male' ) THEN

DBMS\_OUTPUT.PUT\_LINE('Gender Male Record Skip!');

ELSE

IF ( result = 'distinction' ) THEN

DBMS\_OUTPUT.PUT\_LINE('First Class with Distinction');

ELSIF ( result = 'first' ) THEN

DBMS\_OUTPUT.PUT\_LINE('First Class');

ELSIF ( result = 'second' ) THEN

DBMS\_OUTPUT.PUT\_LINE('Second Class');

ELSIF ( result = 'third' ) THEN

DBMS\_OUTPUT.PUT\_LINE('Third Class');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Fail');

END IF;

END IF;

END;

/

#### **Result**

Second Class  
  
PL/SQL procedure successfully completed.

Once if condition find TRUE then execute that true block and skip other block that block never execute.

### PL/SQL Loop - Basic Loop, FOR Loop, WHILE Loop

PL/SQL Loop Basic Loop, FOR Loop, WHILE Loop repeat a number of block statements in your PL/SQL program. Loop use when we have a block of statements for required to repeatedly certain number of times. PL/SQL loop statements 3 different forms:

1. [Basic LOOP](http://www.way2tutorial.com/plsql/plsql_loop_basic_loop_while_loop_for_loop.php#basicloop)
2. [WHILE LOOP](http://www.way2tutorial.com/plsql/plsql_loop_basic_loop_while_loop_for_loop.php#whileloop)
3. [FOR LOOP](http://www.way2tutorial.com/plsql/plsql_loop_basic_loop_while_loop_for_loop.php#forloop)

Oracle recommended to write a label when use loop statement. It's benefit to improve readability. label is not compulsory for execute loop. compiler does not check to label defined or not. Define label before LOOP keyword and after END LOOP keyword.

### Basic LOOP

Basic LOOP write in following syntax format:

[ label\_name ] LOOP

statement(s);

END LOOP [ label\_name ];

#### **Example Code**

DECLARE

no NUMBER := 5;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE ('Inside value: no = ' || no);

no := no -1;

IF no = 0 THEN

EXIT;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Outside loop end');

END;

/

#### **Example Result**

Inside value: no = 5  
Inside value: no = 4  
Inside value: no = 3  
Inside value: no = 2  
Inside value: no = 1  
Outside loop end  
  
PL/SQL procedure successfully completed.

### WHILE LOOP

WHILE LOOP write in following syntax format:

[ label\_name ] WHILE condition LOOP

statement(s);

END LOOP [ label\_name ];

#### **Example Code**

DECLARE

no NUMBER := 0;

BEGIN

WHILE no < 10 LOOP

no := no + 1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Sum :' || no);

END;

/

#### **Example Result**

Sum : 10  
  
PL/SQL procedure successfully completed.

### FOR LOOP

FOR LOOP write in following syntax format:

[ label\_name ] FOR current\_value IN [ REVERSE ] lower\_value..upper\_value LOOP

statement(s);

END LOOP [ label\_name ];

#### **Example Code**

BEGIN

FOR no IN 1 .. 5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Iteration : ' || no);

END LOOP;

END;

/

#### **Example Result**

Iteration : 1  
Iteration : 2  
Iteration : 3  
Iteration : 4  
Iteration : 5  
  
PL/SQL procedure successfully completed.

#### **REVERSE FOR Loop**

Optional REVERSE keyword introduce to iteration is proceed from upper\_value to lower\_value range.

Example Code :

BEGIN

FOR no IN REVERSE 5 .. 1 LOOP

DBMS\_OUTPUT.PUT\_LINE('Iteration : ' || no);

END LOOP;

END;

/

Example Result :

Iteration : 5  
Iteration : 4  
Iteration : 3  
Iteration : 2  
Iteration : 1  
  
PL/SQL procedure successfully completed.

### PL/SQL EXIT CONTINUE GOTO Statements

PL/SQL EXIT, CONTINUE, GOTO Statements (Sequential Control Statements) are control your iteration loop, This three statement are sequential control statement,

1. [EXIT Statement](http://www.way2tutorial.com/plsql/plsql_exit_continue_goto_statements_sequential_control.php#exitstatement) : This statement to exit the loop.  
   [EXIT WHEN Statement](http://www.way2tutorial.com/plsql/plsql_exit_continue_goto_statements_sequential_control.php#exitwhenstatement) : This statement to exit, when WHEN clauses condition true.
2. [CONTINUE Statement](http://www.way2tutorial.com/plsql/plsql_exit_continue_goto_statements_sequential_control.php#continuestatement) : to skip the current iteration with in loop.  
   [CONTINUE WHEN Statement](http://www.way2tutorial.com/plsql/plsql_exit_continue_goto_statements_sequential_control.php#continuewhenstatement) : to skip the current iteration with in loop when WHEN clauses condition true.
3. [GOTO Statement](http://www.way2tutorial.com/plsql/plsql_exit_continue_goto_statements_sequential_control.php#gotostatement) : Transfers the program execution flow unconditionally.

### EXIT Statement

EXIT statement unconditionally exit the current loop iteration and transfer control to end of current loop. EXIT statement writing syntax,

Syntax

[ label\_name ] LOOP

statement(s);

EXIT;

END LOOP [ label\_name ];

#### **Example Code**

DECLARE

no NUMBER := 5;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE ('Inside value: no = ' || no);

no := no -1;

IF no = 0 THEN

EXIT;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Outside loop end'); -- After EXIT control transfer this statement

END;

/

#### **Example Result**

Inside value: no = 5  
Inside value: no = 4  
Inside value: no = 3  
Inside value: no = 2  
Inside value: no = 1  
Outside loop end  
  
PL/SQL procedure successfully completed.

### EXIT WHEN Statement

EXIT WHEN statement unconditionally exit the current loop iteration when WHEN clause condition true. EXIT WHEN statement writing syntax,

Syntax

[ label\_name ] LOOP

statement(s);

EXIT WHEN condition;

END LOOP [ label\_name ];

#### **Example Code**

SQL>DECLARE

i number;

BEGIN

LOOP

dbms\_output.put\_line('Hello');

i:=i+1;

EXIT WHEN i>5;

END LOOP;

END;

/

#### **Example Result**

Hello  
Hello  
Hello  
Hello  
  
PL/SQL procedure successfully completed.

### CONTINUE Statement

CONTINUE Statement unconditionally skip the current loop iteration and next iteration iterate as normal, only skip matched condition.

Syntax

IF condition THEN

CONTINUE;

END IF;

#### **Example Code**

DECLARE

no NUMBER := 0;

BEGIN

FOR no IN 1 .. 5 LOOP

IF i = 4 THEN

CONTINUE;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Iteration : ' || no);

END LOOP;

END;

/

#### **Example Result**

Iteration # 1  
Iteration # 2  
Iteration # 3  
Iteration # 5  
  
PL/SQL procedure successfully completed.

### CONTINUE WHEN Statement

CONTINUE WHEN Statement unconditionally skip the current loop iteration when WHEN clauses condition true,

Syntax

CONTINUE WHEN condition;

statement(s);

#### **Example Code**

DECLARE

no NUMBER := 0;

BEGIN

FOR no IN 1 .. 5 LOOP

DBMS\_OUTPUT.PUT\_LINE('Iteration : ' || no);

CONTINUE WHEN no = 4

DBMS\_OUTPUT.PUT\_LINE('CONTINUE WHEN EXECUTE Iteration : ' || no);

END LOOP;

END;

/

#### **Example Result**

Iteration # 1  
Iteration # 2  
Iteration # 3  
CONTINUE WHEN EXECUTE Iteration : 4  
Iteration # 5  
  
PL/SQL procedure successfully completed.

### GOTO Statement

GOTO statement unconditionally transfer program control. GOTO statement writing syntax,

Syntax

GOTO code\_name

-----------

-----------

<<code\_name>>

-----------

-----------

#### **Example Code**

SQL>BEGIN

FOR i IN 1..5 LOOP

dbms\_output.put\_line(i);

IF i=4 THEN

GOTO label1;

END IF;

END LOOP;

<<label1>>

DBMS\_OUTPUT.PUT\_LINE('Row Filled');

END;

/

#### **Example Result**

1  
2  
3  
4  
Row Filled  
  
PL/SQL procedure successfully completed.

### PL/SQL Case Statement

PL/SQL CASE statement comparing one by one sequencing conditions. CASE statement attempt to match expression that is specified in one or more WHEN condition. CASE statement are following two forms,

1. [Simple CASE Statement](http://www.way2tutorial.com/plsql/plsql_case_statement.php#simplecase)
2. [Searched CASE Statement](http://www.way2tutorial.com/plsql/plsql_case_statement.php#searchedcase)

### Simple CASE Statement

PL/SQL simple CASE statement evaluates selector and attempt to match one or more WHEN condition.

#### **Syntax**

CASE selector

WHEN value-1

THEN statement-1;

WHEN value-2

THEN statement-2;

ELSE

statement-3;

END CASE

#### **Example Code**

SQL>DECLARE

a number := 7;

BEGIN

CASE a

WHEN 1 THEN

DBMS\_OUTPUT.PUT\_LINE('value 1');

WHEN 2 THEN

DBMS\_OUTPUT.PUT\_LINE('value 2');

WHEN 3 THEN

DBMS\_OUTPUT.PUT\_LINE('value 3');

ELSE

DBMS\_OUTPUT.PUT\_LINE('no matching CASE found');

END CASE;

END;

/

#### **Example Result**

no matching CASE found  
  
PL/SQL procedure successfully operation.

### Searched CASE Statement

PL/SQL searched CASE statement has not selector and attempt to match one or more WHEN clauses condition.

#### **Syntax**

CASE

WHEN condition-1 THEN

statement-1;

WHEN condition-2 THEN

statement-2;

ELSE

statement-3;

END CASE;

#### **Example Code**

SQL>DECLARE

a number := 3;

BEGIN

CASE

WHEN a = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('value 1');

WHEN a = 2 THEN

DBMS\_OUTPUT.PUT\_LINE('value 2');

WHEN a = 3 THEN

DBMS\_OUTPUT.PUT\_LINE('value 3');

ELSE

DBMS\_OUTPUT.PUT\_LINE('no matching CASE found');

END CASE;

END;

/

#### **Example Result**

value 3