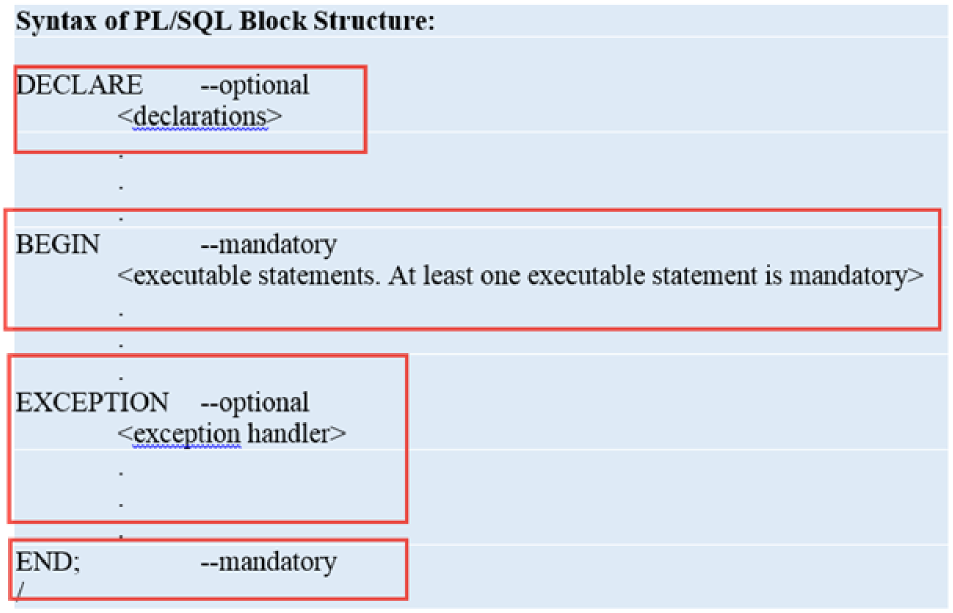
CONTROL STATEMENT



Syntax of PL/SQL Block Structure:

```
DECLARE        --optional
          <declarations>
          .
          .
          .
BEGIN          --mandatory
          <executable statements. At least one executable statement is mandatory>
          .
          .
          .
EXCEPTION      --optional
          <exception handler>
          .
          .
          .
END;           --mandatory
/
```

Decision making statements

Syntax: (IF-THEN statement):

```
IF condition
THEN
Statement: {It is executed when condition is true}
END IF;
```

****
*

Syntax: (IF-THEN-ELSE statement):

```
IF condition
THEN
   {...statements to execute when condition is TRUE...}
ELSE
   {...statements to execute when condition is FALSE...}
END IF;
```

****

Syntax: (IF-elsif)

```
IF condition1
THEN
   {...statements to execute when condition1 is TRUE...}
ELSIF condition2
THEN
   {...statements to execute when condition2 is TRUE...}
ELSE
   {...statements to execute when both condition1 and condition2 are
FALSE...}
END IF;
```

EXAMPLE-1

```
DECLARE
   a number(3) := :x;
BEGIN
```

```
  -- check the boolean condition using if statement
  IF( a < 0 ) THEN
     -- if condition is true then print the following
     dbms_output.put_line('a is negative ' );
  ELSIF(a>0) then
     dbms_output.put_line('a is positive ' );
   else
     dbms_output.put_line('a is zero ' );
  END IF;

END;
```

EXAMPLE-2

```
DECLARE
ACNO VARCHAR2(5);
CNM VARCHAR2(20);
AM NUMBER(8,2);
INS NUMBER(8,2);

BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER NAME');
CNM:= :X;
SELECT ACTNO,CNAME,AMOUNT INTO ACNO,CNM,AM
FROM  DEPOSIT WHERE CNAME=CNM;
```

```
IF AM>=2000 THEN
 INS:=AM*0.3;
ELSIF AM>=1500 AND AM<2000 THEN
 INS:=AM*0.2;
ELSE
INS:=200;
END IF;
DBMS_OUTPUT.PUT_LINE('NAME= '||CNM||' ACTNO='||ACNO||'
INTEREST='||INS);
END;
```

Syntax of CASE

```
CASE [ expression ]
WHEN condition_1 THEN result_1
   WHEN condition_2 THEN result_2
   ...
   WHEN condition_n THEN result_n
 ELSE result
END
```

EXAMPLE 1(CASE)

```
DECLARE
N NUMBER:=&N;
BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER A NUMBER');
```

```
CASE N
WHEN 0 THEN
        DBMS_OUTPUT.PUT_LINE('ZERO');
WHEN 1 THEN
        DBMS_OUTPUT.PUT_LINE('ONE');
WHEN 2 THEN
        DBMS_OUTPUT.PUT_LINE('TWO');
WHEN 3 THEN
        DBMS_OUTPUT.PUT_LINE('THREE');
WHEN 4 THEN
        DBMS_OUTPUT.PUT_LINE('FOUR');
WHEN 5 THEN
        DBMS_OUTPUT.PUT_LINE('FIVE');
WHEN 6 THEN
        DBMS_OUTPUT.PUT_LINE('SIX');
WHEN 7 THEN
        DBMS_OUTPUT.PUT_LINE('SEVEN');
WHEN 8 THEN
        DBMS_OUTPUT.PUT_LINE('EIGHT');
WHEN 9 THEN
        DBMS_OUTPUT.PUT_LINE('NINE');
ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('NOT A SINGLE DIGIT NO');

END CASE;

END;
```

EXAMPLE 2(CASE)

```
DECLARE
ACNO VARCHAR2(5);
CNM VARCHAR2(20);
AM NUMBER(8,2);
INS NUMBER(8,2);
BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER NAME');
CNM:= :X;
SELECT ACTNO,CNAME,AMOUNT INTO ACNO,CNM,AM
FROM  DEPOSITT WHERE CNAME=CNM;
CASE
WHEN AM>=2000 THEN
 INS:=AM*0.3;
WHEN AM>=1500 AND AM<2000 THEN
 INS:=AM*0.2;
ELSE
INS:=22;
```

END CASE;

DBMS_OUTPUT.PUT_LINE('NAME= '||CNM||' ACTNO='||ACNO||' INTEREST='||INS);

END;

PL/SQL Loop

The PL/SQL loops are used to repeat the execution of one or more statements for specified number of times. These are also known as iterative control statements.

****

→Syntax for a basic loop:

```
LOOP
  Sequence of statements;
END LOOP;
```

Types of PL/SQL Loops

****

There are 3 types of PL/SQL Loops.

→Basic Loop / Exit Loop

→While Loop

→For Loop

****

→Syntax of exit loop:

LOOP

statements;

EXIT WHEN condition;}

END LOOP;

****

EXAMPLE

```
DECLARE
N NUMBER :=1;
SUM1 NUMBER :=0;

BEGIN
LOOP
SUM1:=SUM1+N;
N :=N+1;
EXIT WHEN N>10;
END LOOP;
DBMS_OUTPUT.PUT_LINE('SUM OF 1ST 10 NATURAL NOS IS
'||SUM1);
END;
```

→PL/SQL While Loop

PL/SQL while loop is used when a set of statements has to be executed as long as a condition is true, the While loop is used. The condition is decided at the beginning of each iteration and continues until the condition becomes false.

Syntax of while loop:

WHILE <condition>

 LOOP

statements;

END LOOP;

****

<u>EXAMPLE1 OF WHILE LOOP</u>

```
set serveroutput on;
declare
  num1 number(5);
   sum1 number(5);
begin
  num1:=1;
  sum1:=0;
  while num1<=10
  loop

sum1:= sum1+num1;
num1:=num1+1;
end loop;
dbms_output.put_line('welcome'||sum1);
end;
/
```

<u>EXAMPLE2 OF WHILE LOOP Reverse of a number</u>

```
declare
  num1 number(5);
  num2 number(5);
  rev number(5);
```

```
begin
  num1:= :x;
  rev:=0;
  while num1>0
  loop
    num2:=mod(num1,10);
    rev:=num2+(rev*10);
    num1:=floor(num1/10);
  end loop;
  dbms_output.put_line('Reverse number  is: '||rev);
end;
```

➔PL/SQL FOR Loop

PL/SQL for loop is used when when you want to execute a set of statements for a predetermined number of times. The loop is iterated between the start and end integer values. The counter is always incremented by 1 and once the counter reaches the value of end integer, the loop ends.

Syntax of for loop:

```
FOR counter IN initial_value .. final_value
LOOP
  LOOP statements;
END LOOP;
```

//Factorial of a number using for loop

declare

```
      i number(4):=1;
      n number(4):= :x;
      f number(4):=1;
 begin
    for i in 1..n
    loop
      f:=f*i;
    end loop;
    Dbms_output.put_line('the factorial of '||n||' is:'||f);
 end;
//Reverse of a string using for loop

declare
    str1 varchar2(50):='&str';
    str2 varchar2(50);
    len number;
    i number;

begin
    len:=length(str1);

    for i in  1..len
    loop
        str2:=str2 || substr(str1,i,1);
    end loop;

    dbms_output.put_line('Reverse of String is:'||str2);
```

```
end;

/
```

PL/SQL Procedure

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.
- **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

How to pass parameters in procedure:

When you want to create a procedure or function, you have to define parameters .There is three ways to pass parameters in procedure:

1. **IN parameters:** The IN parameter can be referenced by the procedure or function. The value of the parameter cannot be overwritten by the procedure or the function.
2. **OUT parameters:** The OUT parameter cannot be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.
3. **INOUT parameters:** The INOUT parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.