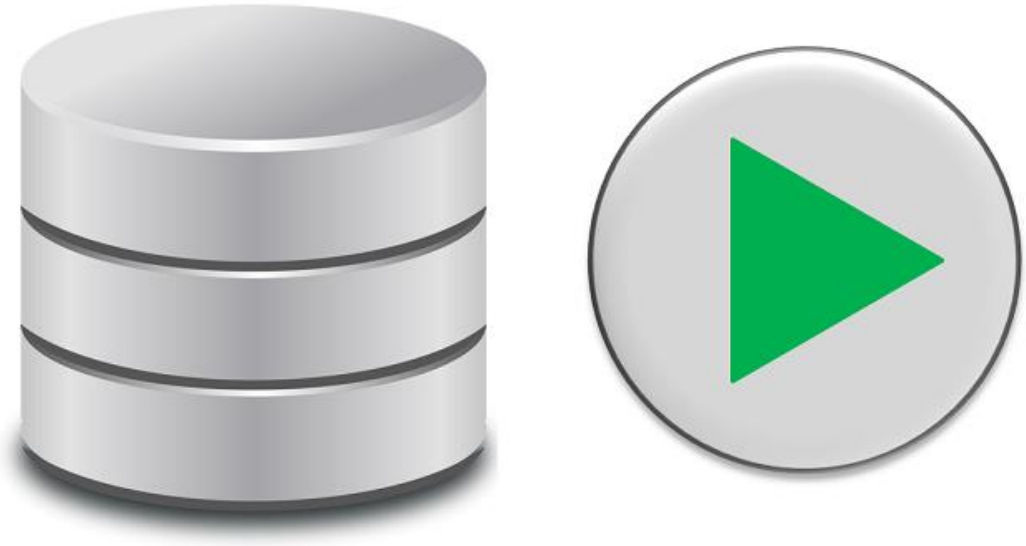
ORACLE PL/SQL



MÓDULO 04

ESTRUCTURAS DE CONTROL

GUSTAVO CORONEL

www.desarrollasoftware.com

ORACLE PL/SQL

Contenido

ESTRUCTURAS SELECTIVAS

Estructura IF

Caso 1: Condicional Simple

Sintaxis

```
   if (condicion) then
      -----------
      -----------
      -----------
   end if
```

Script 1

Función para encontrar el mayor de 3 números positivos.

```
create or replace function fn102 (n1 number, n2 number, n3 number) return number
is
   mayor number := 0;
begin
   if (n1>mayor) then
      mayor := n1;
   end if;
   if (n2>mayor) then
      mayor := n2;
   end if;
   if (n3>mayor) then
      mayor := n3;
   end if;
   return mayor;
end;
```

Ejecución:

```
SQL> select fn102(12,45,4) from dual;

FN102(12,45,4)
--------------
            45
```

Caso 2: Condicional Doble

Sintaxis

```
   if (condicion) then
      -------------
      -------------
   else
      -------------
      -------------
   end if
```

Script 2

Función para encontrar el mayor de 3 números.

```
create or replace function fn103 (n1 number, n2 number, n3 number) return number
is
   mayor number;
begin
   if (n1>n2) then
      mayor := n1;
   else
      mayor := n2;
   end if;
   if (n3>mayor) then
      mayor := n3;
   end if;
   return mayor;
end;
```

Ejecución:

```
SQL> select fn103(-4,-8,-23) from dual;

FN103(-4,-8,-23)
----------------
              -4
```

Caso 3: Condicional Multiple

```
   if (condicion1) then
      --------------
      --------------
   elsif (condicion2) then
      --------------
      --------------
   elsif (condicion3) then
      --------------
      --------------
   . . .
   [else
      --------------
      --------------]
   end if;
```

Script 3

Clasificar el salario de un empleado.

```
create or replace function fn104 (p_empno emp.empno%type) return varchar2
is
   v_sal emp.sal%type; v_msg varchar2(30);
begin
   select sal into v_sal from emp where empno = p_empno;
   if (v_sal<2500) then
      v_msg := 'Salario Bajo';
   elsif (v_sal<4000) then
      v_msg := 'Salario Regular';
   else
      v_msg := 'Salario Bueno';
   end if;
   v_msg := to_char(v_sal) || ' - ' || v_msg;
   return v_msg;
end;
```

Ejecución:

```
SQL> select fn104(7698) from dual;

FN104(7698)
--------------------------------------------------
2850 - Salario Regular
```

Estructura Case

Caso 1: Utilizando un Selector

Sintaxis

```
   case selector
      when valor1 then
         -----------------
         -----------------
      when valor2 then
         -----------------
         -----------------
      . . .
      [else
         -----------------
         -----------------]
   end case;
```

Script 4

Función para evaluar un número.

```
create or replace function fn105(n number) return varchar2
is
   rpta varchar2(30);
begin
   case n
      when 1 then
         rpta := 'Uno';
      when 2 then
         rpta := 'Dos';
      else
         rpta := 'None';
   end case;
   return rpta;
end;
```

Ejecución:

```
SQL> select fn105(1) from dual;

FN105(1)
-----------------------------------------
Uno
```

Caso 2: Utilizando Condiciones

Sintaxis

```
  case
     when condicion1 then
        -------------------
        -------------------
     when condicion2 then
        -------------------
        -------------------
     . . .
     [else
        -------------------
        -------------------]
  end case;
```

Script 5

Realizar una función para evaluar el salario de un empleado.

```
create or replace function fn106(v_empno emp.empno%type) return varchar2 is
   v_msg varchar2(40);
   v_sal emp.sal%type;
begin
   select sal into v_sal from emp where empno = v_empno;
   case
      when (v_sal > 0 and v_sal <= 2500) then
         v_msg := 'Salario Bajo';
      when (v_sal > 2500 and v_sal <= 4000) then
         v_msg := 'Salario Regular';
      when (v_sal > 4000) then
         v_msg := 'Salario Bueno';
      else
         v_msg := 'Caso Desconocido';
   end case;
   v_msg := to_char(v_sal) || ' - ' || v_msg;
   return v_msg;
end;
```

Ejecución:

```
SQL> select fn106(7782) from dual;

FN106(7782)
--------------------------------------------------
2450 - Salario Bajo
```

BUCLES

Objetos previos

Secuencia: sqtest

Script 6

```
SQL> create sequence sqtest;

Sequence created.
```

Tabla Temporal

Script 7

```
SQL> create global temporary table test (
  2   id number primary key,
  3   dato varchar2(30)
  4  ) on commit preserve rows;

Table created.
```

Nota

Las tablas temporales almacenan datos temporalmente, la opción **ON COMMIT PRESERVE ROWS** borra las filas de datos al cerrar la sesión, y la opción **ON COMMIT DELETE ROWS** borra los datos al ejecutar COMMIT.

Bucle Simple: LOOP

Sintaxis:

```
   loop
      ---------------
      ---------------
      if (Condición) then
         ---------------
         ---------------
         exit;
      end if;
      ---------------
      ---------------
      exit when (condición);
      ---------------
      ---------------
   end loop;
```

Script 8

Función para encontrar el factorial de un número.

```
create or replace function fn107 (n number) return number
is
   f number := 1;
   cont number := n;
begin
   loop
      f := f * cont;
      cont := cont - 1;
      exit when (cont=0);
   end loop;
   return f;
end;
```

Ejecución:

```
SQL> select fn107(5) from dual;

  FN107(5)
```

```
----------
       120
```

Bucle: While

Sintaxis

```
   while (condicion) loop
      ---------------
      ---------------
   end loop;
```

Script 9

Insertar datos en una tabla.

```
create or replace procedure pr102 (n number)
is
   k number := 0;
begin
   while (k<n) loop
      insert into test( id,dato )
         values( sqtest.nextval, 'Gustavo Coronel' );
      k := k + 1;
   end loop;
   commit;
   dbms_output.put_line('Proceso Ejecutado');
end;
```

Ejecución:

```
SQL> execute pr102(15);
Proceso Ejecutado

PL/SQL procedure successfully completed.
```

Verificar el contenido de la tabla:

```
SQL> select * from test;

        ID DATO
---------- ------------------------------
         1 Gustavo Coronel
         2 Gustavo Coronel
         3 Gustavo Coronel
         4 Gustavo Coronel
         5 Gustavo Coronel
         6 Gustavo Coronel
         7 Gustavo Coronel
         8 Gustavo Coronel
         9 Gustavo Coronel
        10 Gustavo Coronel
        11 Gustavo Coronel
        12 Gustavo Coronel
        13 Gustavo Coronel
        14 Gustavo Coronel
        15 Gustavo Coronel

15 rows selected.
```

Bucle: FOR

Sintaxis

```
  for contador in [reverse] limite_inferior .. limite_superior loop
     ---------------
     ---------------
  end loop;
```

Script 10

Calcular el factorial de un número.

```
create or replace function fn108 ( n number ) return number
is
   f number := 1;
begin
   for k in 1 .. n loop
      f := f * k;
   end loop;
   return f;
end;
```

Ejecución:

```
SQL> select fn108(5) from dual;

  FN108(5)
----------
       120
```

Script 11

Ilustrar que no es necesario declarar el contador del for.

```
create or replace procedure pr103 ( n number, msg varchar2 )
is
   k number := 1000;
begin
   for k in 1 .. n loop
      dbms_output.put_line( k || ‘ – ‘ || msg );
   end loop;
   dbms_output.put_line( 'k = ' || k );
end;
```

Ejecución:

```
SQL> execute pr103(5,'ALIANZA CAMPEON');
1 - ALIANZA CAMPEON
2 - ALIANZA CAMPEON
3 - ALIANZA CAMPEON
4 - ALIANZA CAMPEON
5 - ALIANZA CAMPEON
k = 1000

PL/SQL procedure successfully completed.
```

Script 12

Aplicación de reverse. Tabla de multiplicar.

```
create or replace procedure pr104 ( n number )
is
   cad varchar2(30);
begin
   for k in reverse 1 .. 12 loop
      cad := n || ' x ' || k || ' = ' || (n*k);
      dbms_output.put_line( cad );
   end loop;
end;
```

Ejecución:

```
SQL> execute pr104(5);
5 x 12 = 60
5 x 11 = 55
5 x 10 = 50
5 x 9 = 45
5 x 8 = 40
5 x 7 = 35
5 x 6 = 30
5 x 5 = 25
5 x 4 = 20
5 x 3 = 15
5 x 2 = 10
5 x 1 = 5

PL/SQL procedure successfully completed.
```

OTROS ELEMENTOS DE PROGRAMACIÓN

Etiquetas

Crear una etiqueta:

```
<<nombre_etiqueta>>
```

Saltar a una etiqueta:

```
goto nombre_etiqueta;
```

Script 13

Elevar un número a una potencia.

```
create or replace function fn109( b number, p number ) return number
is
   r number := 1;
   k number := 0;
begin
   loop
      k := k + 1;
      r := r * b;
      if (k=p) then
         goto fin;
      end if;
   end loop;
   <<fin>>
   return r;
end;
```

Ejecución:

```
SQL> select fn109(-5,3) from dual;

FN109(-5,3)
-----------
       -125
```

Restricciones

1. No se puede realizar un salto al interior de un if, bucle, ó bloque interno.
2. No se puede saltar de una cláusula if a otra, en la misma instrucción if.
3. No se puede saltar de un bloque de excepciones al bloque de instrucciones.

Etiquetando los Bucles

Formato

```
<<abc>>
for ...
   --------------------------
   --------------------------
   <<xyz>>
   for ...
      --------------------------
      --------------------------
      if ...
         exit abc;
      end if;
   end loop xyz;
   --------------------------
   --------------------------
end loop abc;
```

Instrucción NULL

Formato

```
   if (condición) then
      null;
   else
      ----------------
      ----------------
   end if;
```

Script 14

Desarrollar una función para determinar si número es impar.

```
create or replace function fn110( n number )
return varchar2
is
   rtn varchar2(30) := '';
begin
   if ( mod(n,2) = 0 ) then
      null;
   else
      rtn := n || ' es impar';
   end if;
   return rtn;
end;
```

Ejecución:

```
SQL> select fn110(15) from dual;

FN110(15)
----------------------------------------
15 es impar
```

CURSOS VIRTUALES

En esta URL tienes cupones de descuento:

http://gcoronelc.blogspot.com/p/cursos-virtuales.html

Java Orientado a Objetos

JAVA

ORIENTADO A OBJETOS

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

https://www.udemy.com/java-orientado-a-objetos/

Programación de Base de Datos con Java JDBC

PROGRAMACIÓN DE BASE DE DATOS ORACLE CON JAVA JDBC

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
INSTRUCTOR

https://www.udemy.com/java-jdbc-oracle /