CURSO PL-SQL

BLOQUES PL-SQL

sintaxis

WHILE condicion LOOP

bloque de instrucciones

END LOOP;

También se puede hacer un ciclo FOR con la siguiente sintaxis:

FOR contador IN limite_inferior..limite superior
LOOP
bloque de instrucciones
END LOOP;

Se puede hacer un ciclo for EN REVERSA

FOR contador IN REVERSE limite_Superior..limite_inferior
LOOP
bloque de instrucciones
END LOOP;

CURSO PL-SQL

Ejemplo 1. LOOP

Create table Temporal (
contador number,
mensaje varchar2(100)
);

```
Create tABLE Temporal (
contador number,
mensaje varchar2(100)
);
CREATE OR REPLACE  PROCEDURE BUCLE_SIMPLE
AS
v_contador binary_integer:=1;

BEGIN

LOOP
insert into temporal
VALUES (v_contador, 'indice del bucle');
 v_contador:= v_contador+ 1;

IF v_contador >50 THEN
  EXIT;
 END IF;
END LOOP;
END BUCLE_SIMPLE;

EXECUTE BUCLE_SIMPLE();
```

Indica la terminación del ciclo

```
CREATE OR REPLACE  PROCEDURE BUCLE_SIMPLE_2
AS
v_contador binary_integer:=1;
BEGIN


LOOP
insert into temporal
VALUES (v_contador, 'indice del bucle');
v_contador:= v_contador+ 1;

--condicion de salida propia del loop
EXIT WHEN v_contador> 100;
END LOOP;

END BUCLE_SIMPLE_2;

EXECUTE BUCLE_SIMPLE_2();
Select *from temporal;
```

Otra forma de Indicar la terminación del ciclo

CURSO PL-SQL

Ejemplo 2. LOOP

Usando un ciclo While

```
CREATE OR REPLACE  PROCEDURE BUCLE_SIMPLE_3
AS
v_contador binary_integer:=1;
BEGIN

--Evalua la condicion para entrar al bucle

WHILE v_contador <=100 LOOP
insert into temporal
VALUES (v_contador, 'indice del bucle');
v_contador:= v_contador+ 1;
END LOOP;

END BUCLE_SIMPLE_3;
```

Primero evalúa la condición

Ejemplo 3. FOR

```
CREATE OR REPLACE  PROCEDURE BUCLE_SIMPLE_4
AS
v_contador binary_integer:=10;

BEGIN

insert into temporal
VALUES (v_contador, 'indice del bucle');

--Evalua la condicion para entrar a bucle

FOR contador in 20..30 LOOP

insert into temporal
VALUES (v_contador, 'indice del bucle');
v_contador:= v_contador + 1;

END LOOP;

END BUCLE_SIMPLE_4;
```

Inicia en 10 hasta que el contad llega a 20 (1 ejecuciones)

CURSO PL-SQL

Ejemplo 4. LOOP

Usando un ciclo for en reversa

```
CREATE OR REPLACE  PROCEDURE BUCLE_SIMPLE_5_REVERSA
AS
v_contador binary_integer:=10;

BEGIN

insert into temporal
VALUES (v_contador, 'indice del bucle');

--insercion bucle en reversa

FOR contador IN REVERSE 20..30 LOOP

insert into temporal
VALUES (contador, 'indice del bucle');
v_contador:= v_contador+ 1;

END LOOP;

END BUCLE_SIMPLE_5_REVERSA;
execute BUCLE_SIMPLE_5_REVERSA();
```

LA VARIABLE “CONTADOR” DEL LOOP VA A LA INSERCION DE LA TABLA

CURSO PL-SQL

Ejemplo 4. LOOP

Usando un ciclo for en reversa Con inserción secuencial y función randómica.

```
CREATE OR REPLACE  PROCEDURE INSERTAALE
AS
precio number(6,3);

stock number;
id number(5);
cadenal varchar(50);
aleatoria NUMBER(6,2);
BEGIN

id := 11;
cadenal := id;

LOOP

SELECT
ABS(dbms_random.normal) into
aleatoria FROM dual;

DBMS_OUTPUT.PUT_LINE(aleatoria);

precio := MOD(aleatoria,10)+ 100;

stock := MOD(aleatoria,10) + 1;

cadenal := id;

insert into libros values
 (id, CONCAT ('Libro', cadenal), CONCAT('Autor',cadenal), precio, stock);
id := id + 1;
EXIT WHEN id > 1000;
END LOOP;
END INSERTAALE;
```

La variable “aleatoria” obtiene un entero por módulo entre 1 y 10 y le suma 100

La variable “cadena1” recibe un id numérico para usarlo como varchar.

CURSO PL-SQL

Ejemplo 4. LOOP

Usando un ciclo for en reversa Con inserción secuencial y función randómica.

```
CREATE OR REPLACE  PROCEDURE INSERTAALE
AS
precio number(6,3);

stock number;
id number(5);
cadenal varchar(50);
aleatoria NUMBER(6,2);
BEGIN

id := 11;
cadenal := id;

LOOP

SELECT
ABS(dbms_random.normal) into
aleatoria FROM dual;

DBMS_OUTPUT.PUT_LINE(aleatoria);

precio := MOD(aleatoria,10)+ 100;

stock := MOD(aleatoria,10) + 1;

cadenal := id;

insert into libros values
 (id, CONCAT ('Libro', cadenal), CONCAT('Autor',cadenal), precio, stock);
id := id + 1;
EXIT WHEN id > 1000;
END LOOP;
END INSERTAALE;
```

La variable “aleatoria” obtiene un entero por módulo entre 1 y 10 y le suma 100

La variable “cadena1” recibe un id numérico para usarlo como varchar.

CURSO PL-SQL

EJERCICIO.

Realizar un procedimiento almacenado que genere seis números para el juego del baloto y los almacene en una tabla. Teniendo en cuenta que:

1 . Todos los números son distintos
2. Los números son de dos dígitos entre el 01 y el 45
3. Los números se obtienen de manera aleatoria.
4. Al ejecutar el procedimiento se generar el juego y se almacena en una tabla llamada juegos como la del siguiente ejemplo:

Fecha	usuario	No.1	No.2	No.3	No.4	No.5	No.6
27/08/2019	HR	12	15	22	33	36	45

CURSO PL-SQL

Ejemplo 5. WARRAYS

The PL/SQL programming language provides a data structure called the **VARRAY**, which can store a fixed-size sequential collection of elements of the same type. A varray is used to store an ordered collection of data, however it is often better to think of an array as a collection of variables of the same type.

All varrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element

```
CREATE OR REPLACE TYPE varray_type_name IS VARRAY(n) of <element_type>
```

Where,

- *varray_type_name* is a valid attribute name,
- *n* is the number of elements (maximum) in the varray,
- *element_type* is the data type of the elements of the array.

Maximum size of a varray can be changed using the **ALTER TYPE** statement.

For example,

```
CREATE Or REPLACE TYPE namearray AS VARRAY(3) OF VARCHAR2(10);
/

Type created.
```

CURSO PL-SQL

Ejemplo 5. WARRAYS

The PL/SQL programming language provides a data structure called the **VARRAY**, which can store a fixed-size sequential collection of elements of the same type. A varray is used to store an ordered collection of data, however it is often better to think of an array as a collection of variables of the same type.

All varrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element

```
create OR REPLACE PROCEDURE Varray1
AS
   type Nombresarray IS VARRAY(5) OF VARCHAR2(10);
   type cursos IS VARRAY(5) OF INTEGER;
   nombres Nombresarray;
   identificacion_c cursos;
   total integer;
BEGIN
   nombres := Nombresarray('KEvin', 'Pinto', 'Ayan', 'Risha', 'Arenis');
   identificacion_c:= cursos(98, 97, 78, 87, 92);
   total := nombres.count;
   dbms_output.put_line('En Total hay : '|| total || ' Estudiantes');
   FOR i in 1 .. total LOOP
      dbms_output.put_line('Estudiantes: ' || nombres(i) || '
      identificaion: ' || identificacion_c(i));
   END LOOP;
END Varray1;

execute Varray1();
```

CURSO PL-SQL

Ejemplo 6. CURSORES EXPLICITOS

El siguiente ejenmplo extrae de la table empleados2 los nombres de todos los registros de la Tabla en un cursor llamado cursor_clientes, el objeto que guarda los registros se llama nombre_lista

```
DECLARE
   CURSOR cursor_clientes is
   SELECT  Nombre FROM empleados2;
   type c_list is varray (6) of empleados2.nombre%type;
   nombre_lista c_list := c_list();
   contador integer :=0;
BEGIN
   FOR n IN cursor_clientes LOOP
      contador := contador + 1;
      nombre_lista.extend;
      nombre_lista(contador)  := n.nombre;
      dbms_output.put_line('Personal:('||contador ||'):'||nombre_lista(contador));
   END LOOP;
END;
```