3 – ESTRUCTURAS DE CONTROL

3.1 – INTRODUCCIÓN

Como cualquier lenguaje procedural, PL/SQL dispone de una serie de órdenes que nos permiten establecer controles y condiciones a los programas, pudiéndose por tanto representarlos mediante operaciones secuenciales, selectivas y repetitivas. Las dos últimas se apoyan en condiciones para establecer su comportamiento.

3.2 – CONTROL CONDICIONAL IF

Nos permite seleccionar unas determinadas órdenes a ejecutarse dependiendo de una condición.

La sintaxis de la condicional simple IF-THEN sería como sigue a continuación:

```
IF condición THEN
        órdenes;
END IF;
```

Las ordenes se ejecutarían en el caso de que la condición fuera verdadera, si por el contrario fueran falsas o devolviera un valor NULL entonces no se realizarían.

Ejemplo 1:

```
IF V_MATRI<12000 THEN
        UPDATE ALUMNOS
        SET MATRÍCULA=12500
        WHERE COD=10;
END IF;
```

La sintaxis para una condicional doble del tipo IF-THEN-ELSE sería la siguiente:

```
IF condición THEN
        órdenes;
ELSE
        órdenes;
END IF;
```

Si la condición es falsa o NULL, entonces se ejecutan las ordenes que siguen a ELSE, en caso de ser verdaderas las que están dentro de THEN.

Ejemplo 2:

Si en la variable 'renta' tenemos una cantidad, que es el total de las matriculas del curso 3, podríamos montar una estructura como esta para saber si el rentable o no suponiendo que la rentabilidad esté por encima del millón de Ptas.

```
……
SELECT SUM(MATRICULA) INTO RENTA
FROM ALUMNOS
WHERE CODIGO_CURSO=3;
IF RENTA > 1000000 THEN
        INSERT INTO TEMP(COL3) VALUES ('RENTABLE');
ELSE
        INSERT INTO TEMP(COL3) VALUES ('NO RENTABLE');
END IF;
……
```

Cuando sea posible, se utilizará la cláusula ELSIF en lugar de anidar tantas veces las sentencias IF. El código es más fácil de leer y de entender y la lógica está claramente definida. Si la acción de la cláusula ELSE únicamente consiste en otra sentencia IF, es preferible utilizar la sentencia ELSIF.

Sintaxis:

```
IF condición THEN
        órdenes;
ELSIF condición2 THEN
        órdenes;
ELSE
        órdenes;
END IF;
```

3.3 - ESTRUCTURAS REPETITIVAS

PL/SQL proporciona prestaciones para estructurar bucles y así repetir varias veces una sentencia o un conjunto de éstas.

Los bucles pueden ser de 3 tipos:

- LOOP BÁSICO.
- WHILE.
- FOR.

BUCLE LOOP

Proporciona acciones repetitivas sin condiciones globales y requiere la sentencia EXIT para finalizar el bucle. Si se omite dicha cláusula daría lugar a un bucle infinito.

Sintaxis:

```
LOOP
        órdenes;
        EXIT [WHEN condición];
END LOOP;
```

Ejemplo 1:

Insertar en la tabla ITEM, diez registros, se saldrá del bucle cuando se hayan insertado esos registros.

```
...
LOOP
        INSERT INTO ITEM(ORDER, ITEMID) VALUES (V_ORDER, V_ITEMID);
        V_COUNTER:=V_COUNTER+1;
        EXIT WHEN V_COUNTER>10;
END LOOP;
…
```

BUCLE WHILE

Nos permite asociar una determinada condición a la hora de ejecutar una serie de comandos. Se utiliza WHILE mientras la condición sea cierta.

Sintaxis:

```
WHILE condición LOOP
        órdenes;
END LOOP;
```

Ejemplo 2:

Diseñar un bucle WHILE que sume uno a una variable, mientras que esta sea menor o igual que 100.

```
…
WHILE V_X<=100 LOOP
        V_X:=V_X+1;
END LOOP;
…
```

BUCLE FOR

Ejecuta las órdenes un número predeterminado de veces según la condición que va declarada. Las directrices para la construcción de FOR son:

- Hacer referencia al índice o variable dentro del bucle ya que no se define en DECLARE.
- Utilizar una expresión para hacer referencia al valor actual del índice.
- No hacer referencia al índice o variable como objetivo de una asignación.

Sintaxis:

```
FOR     indice IN [REVERSE]
                limite_inferior . . limite_superior LOOP
        órdenes;
END LOOP;
```

Ejemplo 3:

Insertar las 10 primeras líneas del pedido número 101.

```
...
V_ORDID ITEM.ORDID%TYPE := 101;
BEGIN
…
FOR I IN 1..10 LOOP
        INSERT INTO ITEM(ORDID, ITEMID)
        VALUES (V_ORDID,I);
END LOOP;
 …
```