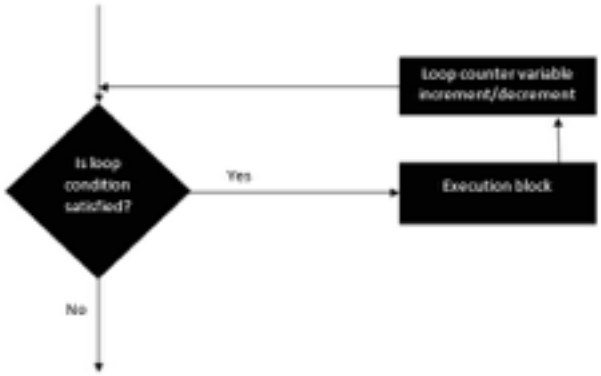
SENTENCIAS DE CONTROL.

SENTENCIAS REPETITIVAS

Las sentencias repetitivas carecen prácticamente de sentido hasta que trabajamos con cursores PL/SQL, los cuales serán estudiados a continuación.



Antes de empezar a ver los distintos tipos de bucles en PL/SQL vamos a ver para que sirven ciertas instrucciones relacionadas con bu que nos las encontraremos en muchos códigos PL.

Continue

Esta palabra reservada sirve para que el motor de PL/SQL cuando la encuentra, salta automáticamente el resto del código que se está ejecutando en ese momento , y empieza inmediatamente la siguiente iteración. Será principalmente usado si el código dentro del loop queremos que sea saltado con ciertos valores de la iteración.

EXIT / EXIT WHEN

Esta palabra reservada sirve para que el motor de PL/SQL cuando la encuentra, salga automáticamente del bucle en el que se encuentra. Si el exit está en un bucle anidado, saldrá solo del bucle mas interno o que tenga el exit, no de los bucles externos.
EXIT WHEN va seguido de una expresión que devuelve un valor Booleano. Si el resultado es TRUE , entonces saldrá del bucle.

GOTO

Goto etiqueta, transfiere el control a otro punto del código.

El uso de GOTO no está recomendado , pues hace que el mantenimiento sea muy complejo.

Sentencia LOOP:

- Este tipo de sentencias sirve para repetir un cierto número de veces las sentencias que englobamos dentro del loop
- Empieza con la palabra Loop y termina con End loop
- para salir del bucle se necesita la sentencia exit.

```
LOOP
    sentencias1;
    EXIT WHEN condición;
    sentencias2;
END LOOP;
```

```
LOOP
  sueldo := sueldo * 2
  EXIT WHEN cont > 7;
  cont := cont + 1;
END LOOP;
```

- La condición que ponemos es la condición de salida del bucle
- Si pongo EXIT; saldremos del bucle.
- En el ejemplo anterior, saldrá del bucle cuando el contador llegue a 8
- si no especificamos un exit, se vuelve un bucle infinito del que no se sale nunca.

Ejemplo:

```
      DECLARE
a NUMBER:=1;
BEGIN
dbms_output.put_line('Program started.');
LOOP
dbms_output.put_line(a);
a:=a+1;
EXIT WHEN a>7;
END LOOP;
dbms_output.put_line('Program completed');
END:
/
```

Sentencia WHILE:

```
WHILE condición LOOP
        sentencias;
END LOOP;
```

```
WHILE sueldo < 100000
  sueldo:=sueldo+7000;
END LOOP;
```

- Mientras la condición sea verdadera seguimos dentro del bucle
- Si pongo EXIT; saldremos del bucle

Sentencia FOR:

Tiene algunas características propias del lenguaje PL/SQL que estudiaremos a continuación.

```
FOR valor IN [REVERSE] valor_ini .. valor_fin LOOP
        sentencias;
END LOOP;
```

Si pongo EXIT; saldremos del bucle

valor.- No es necesario declararse, ya que por defecto se define como tipo BINARY_INTEGER (NUMBER), toma valores enteros y es de solo lectura. El programador no puede modificar su valor pero si consultarlo, aunque únicamente dentro del bucle.
Cuando acaba el bucle la variable desaparece.
Por defecto se incrementa desde el valor_ini hasta el valor_fin. Si el límite superior es menor que el límite inferior, el bucle no se ejecuta.

REVERSE.- Por cada iteración el valor decrece desde valor_fin hasta valor_ini.

El rango de valores de ejecución puede ser establecido en tiempo de ejecución.

Ejemplos:

```
FOR i IN REVERSE 1 .. 7 LOOP
          sueldo := sueldo + 7000;
        END LOOP;
```

```
SELECT COUNT(*) INTO tope FROM vecinos;
FOR i IN 1 .. tope LOOP
          pts_comunidad := pts_comunidad + 700;
        END LOOP;
```

Existe una versión específica del bucle FOR para el tratamiento de cursores y registros, pero la estudiaremos posteriormente.