# 4. CURSORES EXPLÍCITOS.



## 4.1.- INTRODUCCIÓN:

Los cursores se utilizan para trabajar con consultas que devuelven más de una fila. Todas las sentencias SQL ejecutadas por el servidor tienen un cursor asociado, bien implícito declarado para sentencias DML y PL/SQL SELECT.

Existen otros tipos de cursores explícitos, que son creados por el programador y especifican el área de memoria que debe reservar el servidor. Estos últimos serán motivo de estudio de este tema.

## 4.2.- DECLARACIÓN DE CURSORES

Los cursores se deben definir en la zona de declaraciones junto a las variables, constantes, tablas, registros y excepciones. No incluyendo INTO en la declaración del cursor y se puede utilizar la cláusula ORDER BY para procesar filas en una secuencia.

Sintaxis:

```
CURSOR nb_cursor IS
sentencia SELECT;
```

## 4.3.- ABRIR CURSORES

Una vez declarado el cursor procedemos a abrirlo, para ello se usa la orden OPEN, que presenta la siguiente sintaxis :

```
OPEN nb_cursor;
```

La apertura del cursor presenta siempre las siguientes características:

- Abrir el cursor para ejecutar la consulta e identificar el juego activo.
- Si la consulta no devuelve ninguna fila, no se producirá ninguna excepción.
- Utilizar atributos del cursor para comprobar los resultados producidos tras una recuperación.

## 4.4.– RECUPERACIÓN DE DATOS DEL CURSOR

Sintaxis:

```
FETCH nb_cursor INTO [ variable1, variable2, ... | nb_registro] ;
```

La sentencia FETCH recupera las filas del juego de resultados de una en una. Después de cada recuperación, el cursor avanza a la siguiente fila del juego de resultados.

Directrices:

- Recuperar los valores de la fila actual e introducirlos en variables de salida.
- Incluir el mismo número de variables.
- Relacionar posicionalmente las variables y las columnas.
- Comprobar si el cursor tiene filas.

## 4.5.- CERRAR UN CURSOR

Una vez utilizado el cursor, si no se necesita más podemos cerrarlo para que no ocupe memoria. Para centrarlo se utiliza la orden CLOSE.

Sintaxis:

```
CLOSE nb_cursor;
```

Ejemplo 1:

Funcionamiento y manejo de los cursores:

```
DECLARE
        CURSOR MATRI IS
                SELECT MATRICULA, BECARIO FROM ALUMNOS;
        TEMPORAL MATRI%ROWTYPE;
        VAR1 ALUMNOS.MATRICULA%TYPE;
        VAR2 ALUMNOS.MATRICULA%TYPE;
BEGIN

        OPEN MATRI;
        LOOP
                FETCH MATRI INTO TEMPORAL;
                EXIT WHEN MATRI%NOTFOUND
                IF TEMPORAL.BECARIO IS NULL THEN
                        VAR1 := VAR1 + MATRI.MATRICULA;
                ELSE
                        VAR2:= VAR2 + MATRI.MATRICULA;
                END IF
        END LOOP;
        INSERT INTO TEMP(COL1,COL2) VALUES (VAR1, VAR2);
        CLOSE MATRI;
END;
```

Ejemplo 2:

Crear un bloque PL que acepte un número como entrada, elija a los empleados dependiendo del número con el sueldo más alto y los inserte en la tabla OTRA.

```
DECLARE
        CURSOR EMP_CURSOR IS
        SELECT ENAME, SAL FROM EMP WHERE SAL IS NOT NULL
        ORDER BY SAL;
        V_ENAME EMP.ENAME%TYPE;
        V_SAL EMP.SAL%TYPE;
        V_NUM NUMBER(3) := &NÚMERO;

BEGIN

        OPEN EMP_CURSOR;
        FETCH EMP_CURSOR INTO V_ENAME , V_SAL;
        WHILE (EMP_CURSOR%ROWCOUNT <= V_NUM ) AND
                EMP_CURSOR%FOUND LOOP
                INSERT INTO OTRA (ENAME, SALARY)
                        VALUES (V_ENAME, V_SAL);
                FETCH EMP_CURSOR INTO V_ENAME, V_SAL;
        END LOOP;
        CLOSE EMP_CURSOR;
        COMMIT;
END;
/
```

## 4.6. – ATRIBUTOS DE CURSORES EXPLÍCITOS

Un cursor posee una serie de atributos que podemos asociarle, para conocer su estado en un momento determinado. Son los siguientes:

| Atributo | Tipo | Descripción |
|---|---|---|
| %NOTFOUND | Booleano | Resulta TRUE si la recuperación más reciente no devuelve una fila. |
| %FOUND | Booleano | Resulta TRUE si la recuperación más reciente devuelve una fila; complemento de %NOTFOUND. |
| %ROWCOUNT | Numérico | Da el número total de filas devueltas hasta ese momento. |
| %ISOPEN | Booleano | Resulta TRUE si el cursor está abierto. |

- %NOTFOUND**:** se activa si el último FETCH no ha recuperado ninguna fila del cursor.

```
LOOP
        FETCH CURSOR INTO VARIABLE;
        EXIT WHEN CURSOR%NOTFOUND;
END LOOP;
```

- %FOUND**:** se activa si el último FETCH ha recuperado alguna fila del cursor.

```
LOOP
        FETCH CURSOR INTO VARIABLE
        IF CURSOR%FOUND THEN
                ……;
        ELSE
                EXIT;
        END IF;
END LOOP;
```

- %ROWCOUNT: devuelve el número de filas recuperadas hasta ese momento con la instrucción FETCH, al iniciarse vale:

```
LOOP
        FETCH CURSOR INTO VARIABLE
        IF CURSOR%ROWCOUNT = 5
                THEN EXIT;
        END IF;
END LOOP;
```

- %ISOPEN: es verdadero cuando el cursor está abierto, falso si está cerrado.

```
IF CURSOR%ISOPEN THEN
        CLOSE CURSOR;
    ELSE
        OPEN CURSOR;
END IF;
```

## 4.7.– BUCLES FOR DE CURSOR

Un bucle FOR de cursor:

- Procesa filas en un cursor explicito.
- Abre, recupera y cierra de forma automática.
- No declarar la variable registro, ya que se declara implícitamente.

Sintaxis:

```
FOR nb_registro IN nb_cursor LOOP
        órdenes;
        …
END LOOP;
```

Ejemplo:

Recuperar, de uno en uno, todas las líneas de articulo pedido hasta que no queden mas líneas, utilizando un bucle FOR de cursor.

```
.....
FOR PED_REG IN CURSOR_LIN LOOP
        V_TOTAL:=V_TOTAL+(PED_REG.PRECIO*PED_REG.CANTIDAD);
        I:=I+1;
        TB_PRODUCTOS(I):=PED_REG.PRODUCTO;
        TB_TOTAL(I):=V_TOTAL;
END LOOP;
.....
```

Ejemplos de cursores (I).

Escribir un bloque PL/SQL que visualice el apellido y la fecha de alta de todos los empleados de la tabla EMP, ordenados ascendentemente por fecha de alta en la empresa.

**Bucle LOOP**

```
DECLARE

        CURSOR C_EMP
        IS
        SELECT ENAME, HIREDATE
        FROM EMP
        ORDER BY HIREDATE;

        V_ENAME EMP.ENAME%TYPE;
        V_HIREDATE EMP.HIREDATE%TYPE;

BEGIN

        OPEN C_EMP;
        LOOP
                FETCH C_EMP INTO V_ENAME, V_HIREDATE;
                EXIT WHEN C_EMP%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE (V_ENAME||'*'||V_HIREDATE);
        END LOOP;
        CLOSE C_EMP;

END;
/
```

**Bucle WHILE**

```
DECLARE

        CURSOR C_EMP
        IS
        SELECT ENAME, HIREDATE
        FROM EMP
```

```
        ORDER BY HIREDATE;

        V_ENAME EMP.ENAME%TYPE;
        V_HIREDATE EMP.HIREDATE%TYPE;

BEGIN

        OPEN C_EMP;
        FETCH C_EMP INTO V_ENAME, V_HIREDATE;
        WHILE C_EMP%FOUND LOOP
                DBMS_OUTPUT.PUT_LINE (V_ENAME||'*'||V_HIREDATE);
                FETCH C_EMP INTO V_ENAME, V_HIREDATE;
        END LOOP;
        CLOSE C_EMP;
END;
/
```

**Bucle FOR**

```
DECLARE

        CURSOR C_EMP
        IS
        SELECT ENAME, HIREDATE
        FROM EMP
        ORDER BY HIREDATE;

BEGIN

        FOR I IN C_EMP LOOP
                DBMS_OUTPUT.PUT_LINE (I.ENAME||'*'||I.HIREDATE);
        END LOOP;

END;
/
```

Ejemplos de cursores (II).

Crear un bloque PL que acepte un número como entrada, elija a los empleados dependiendo del número con el sueldo mas alto y los inserte en una tabla llamada OTRA.

```
CREATE TABLE OTRA (ENAME VARCHAR2(10),SAL NUMBER(7,2));

DECLARE
        CURSOR EMP_CURSOR
        IS
        SELECT ENAME,SAL
        FROM EMP WHERE SAL IS NOT NULL
```

```
        ORDER BY SAL;

        V_ENAME EMP.ENAME%TYPE;
        V_SAL EMP.SAL%TYPE;
        V_NUM NUMBER(3):=&NUMERO;

BEGIN

        OPEN EMP_CURSOR;
        FETCH EMP_CURSOR INTO V_ENAME,V_SAL;
        WHILE(EMP_CURSOR%ROWCOUNT<=V_NUM) AND
EMP_CURSOR%FOUND LOOP
        INSERT INTO OTRA (ENAME,SAL) VALUES (V_ENAME,V_SAL);
        FETCH EMP_CURSOR INTO V_ENAME,V_SAL;
        END LOOP;
        CLOSE EMP_CURSOR;
        COMMIT;

END;
/
```

## 4.8.- CURSORES CON PARÁMETROS:

Los parámetros permiten que los valores se transfieran a un cursor cuando éste esté abierto, y se puede abrir varias veces en un bloque, devolviendo un juego activo distinto cada vez.

Cada parámetro formal de la declaración del cursor debe tener un parámetro real correspondiente en la sentencia OPEN. Los distintos tipos de datos del parámetro son los mismos que los de las variables escalares, pero no se les asigna tamaños. Los nombres de los parámetros son referenciados en la expresión de la consulta del cursor.

Sintaxis :

```
CURSOR      nb_cursor
            [ (nb_parámetro tipo_dato [ { :=  | DEFAULT } expr ]....)]
            [RETURN tipo_return] IS
            Sentencia SELECT;
```

De los cursores con parámetros podemos decir a modo de resumen que:

- Transfieren los valores de los parámetros a un cursor cuando se abre y se ejecuta la consulta.
- Abre un cursor explícito varias veces con un juego activo distinto cada vez.

Ejemplo:

```
DECLARE
        CURSOR EJEMPLO (VAR1 NUMBER) IS
        SELECT * FROM ALUMNOS WHERE MATRICULA > VAR1;
BEGIN
        OPEN EJEMPLO (150000);
        ……
END;
```

## 4.9.- CLÁUSULAS UTILIZADAS EN LOS CURSORES:

### FOR UPDATE

Esta cláusula se coloca en la SELECT del cursor en la zona declarativa. Permite:

- Bloquear algunas filas antes de la actualización o supresión.
- El bloqueo explicito le permite denegar el acceso mientras dura una transacción.

Sintaxis:

```
SELECT          .....
FROM            .....
FOR UPDATE [ OF col_tabla_referenciada][NOWAIT];
```

Donde:

NOWAIT: devuelve error si las filas fueron bloqueadas en otra sesión.

Ejemplo:

Recuperar los pedidos de importes superiores a 1000$ que se han procesado hoy.

```
DECLARE
    CURSOR C1 IS
        SELECT CUSTID, ORDID
        FROM ORD
    WHERE ORDERDATE = SYSDATE AND TOTAL>1000.00
    ORDER BY CUSTID
    FOR UPDATE NOWAIT;
    …
```

### WHERE CURRENT OF:

Esta cláusula se utiliza siempre que se haga referencia a la fila actual de un cursor explícito, esto permitirá realizar actualizaciones y supresiones a la fila que se está tratando actualmente.

Se incluirá la cláusula FOR UPDATE en la consulta del cursor para bloquear primero las filas.

Se utilizará la cláusula WHERE CURRENT OF para hacer referencia en la fila actual de un cursor explícito. Esta cláusula normalmente se utiliza en la zona BEGIN .

Sintaxis:

```
WHERE CURRENT OF nb_cursor;
```

Ejemplo:

```
DECLARE
        ……
        CURSOR EMP_CURSOR IS
        SELECT ……
        FOR UPDATE;
        ……
BEGIN
        ……
        FOR EMP_RECORD IN EMP_CURSOR LOOP
                UPDATE ……
                WHERE CURRENT OF EMP_CURSOR;
                ……
        END LOOP;
        COMMIT;
END;
```