**CURSORES, EXCEPCIONES Y CONTROL DE TRANSACCIONES**

PL/SQL aporta un conjunto de estructuras llamadas cursores que permiten gestionar grandes cantidades de información de la base de datos.

Existen dos tipos de cursores: implícitos y explícitos.

**Cursores implícitos. Son** abiertos automáticamente cuando se procesa un comando SQL del tipo INSERT, UPDATE o DELETE, y además cuando se ejecuta una orden SELECT INTO y esta consulta devuelve una sola fila. El nombre de este cursor implícito siempre es **SQL**  y dispone de cuatro atributos de los cuales se puede consultar el valor. El valor de los atributos se refiere en cada momento a la última orden ejecutada. Son los siguientes

* **SQL%NOTFOUND** que devuelve TRUE si la última orden no ha afectado a ninguna fila, no se ha insertado ningún dato, ni se ha modificado, ni borrado.
* **SQL%FOUND,**
* **SQL%ROWCOUNT,** devuelve el número de filas afectadas por la última orden.
* **SQL%ISOPEN,** siempre devuelve FALSE ya que se cierra automáticamente después de cada orden SQL.

Hay que tener en cuenta lo siguiente:

* Si la orden que se ejecuta es SELECT. .. INTO devuelve más de una fila, se producirá un error y se levantará automáticamente una excepción que puede ser:
  + NO\_DATA\_FOUND, si la consulta no devuelve ninguna fila.
  + TOO\_MANY\_ROWS, si la consulta devuelve más de una fila.

Con cualquier excepción, se detiene la ejecución normal del programa y se bifurca a la sección de excepciones, el programa no llegará nunca a cualquier orden después de esta SELECT … INTO.

* Con el resto de las órdenes SQL, no se produce error y no se levanta ninguna excepción.

En el siguiente bloque primero se intenta cambiar la localidad de un departamento que no está en la tabla, no se produce error, pero si se intenta consultar, sí se produce excepción:

DECLARE

vnombre depart.dnombre%TYPE:='MARKETING'; --supongamos que no existe

vloc depart.loc%TYPE;

BEGIN

UPDATE depart SET loc='SEVILLA' WHERE dnombre=vnombre;

IF SQL%notfound THEN dbms\_output.put\_line('NO SE PUDO ACTUALIZAR');

END IF;

dbms\_output.put\_line ('PASA POR AQUÍ');

SELECT loc INTO vloc FROM depart WHERE dnombre=vnombre;

IF SQL%notfound THEN dbms\_output.put\_line('NUNCA ENTRA EN ESTE IF');

END IF;

END;

* Cuando un SELECT…INTO hace referencia a una función de grupo, nunca levantará la excepción NO\_DATA\_FOUND y SQL%FOUND siempre será verdadero ya que las funciones de grupo siempre devuelven un valor aunque sea 0 o NULL.

Si tenemos dentro de un bloque una consulta como:

SELECT MAX(SALARIO) INTO MAXIMO WHERE DNOMBRE=’MARKETING’;

Aunque no existiera ningún departamento con el nombre MARKETING devolvería y nunca saltaría una excepción. Esta propiedad se suele utilizar para evitar que si un dato no existe en una tabla evitemos que salte la excepción.

SELECT COUNT(\* ) INTO X FROM TABLA WHERE DEPTNO=VX;

Con esta orden si el departamento VX no existe, simplemente devolverá 0 y no saltará la excepción.

**Cursores explícitos.** Se utilizan para trabajar con consultas que pueden devolver más de una fila. Con estos cursores hay que realizar cuatro operaciones:

1.- Declaración del cursor en la zona de declaraciones:

**CURSOR nombreCursor IS sentenciaSelect;**

2.- Apertura del cursor en el cuerpo del programa, en este momento se ejecuta la sentencia SELECT asociada al cursor y su resultado se guarda en memoria aunque aún no es accesible.

**OPEN nombreCursor;**

3.- Recoger la información desde memoria con la orden FETCH:

**FETCH nombreCursor INTO variable o lista de variables;**

Donde variable puede ser nombreCursor%ROWTYPE y asumirá todas los campos de la SELECT asociada al cursor y lista de variables, puede haber una por cada columna de la sentencia.

4.- Cerrar el cursor, ya no se podrá acceder a su información

**CLOSE nombreCursor**

En el siguiente ejemplo se utiliza un cursor explícito para visualizar el nombre y la localidad de todos los departamentos de la tabla DEPART.

DECLARE

**CURSOR C1 IS SELECT dnombre, loc FROM DEPART;**

Nombre varchar(25);

Loca varchar(25);

BEGIN

**OPEN C1;**

**FETCH C1 INTO nombre, loca;**

WHILE **C1%FOUND** LOOP

DBMS\_OUTPUT.PUT\_LINE(nombre||’ ‘|| loca);

**FETCH C1 INTO nombre, loca;**

END LOOP;

**CLOSE C1;**

END;

En el bucle WHILE preguntaremos la condición C1%FOUND es decir, si el último FETCH ha recuperado algún valor, en otro caso devolverá FALSE.

Los cursores explícitos también poseen los mismos atributos que los implícitos aunque tienen distinto significado.

* **%FOUND**, devolverá TRUE si el último FETCH ha recuperado algún valor, si no devolverá FALSO. También devolverá FALSE si el cursor no estaba abierto o si aún no se ha ejecutado una orden FETCH.
* **%NOTFOUND**, funciona al contrario que el anterior y se suele utilizar como fin de bucle de recorrido de un cursor

EXIT WHEN nombreCursor%NOTFOUND

* **%ROWCOUNT**, devuelve el número de filas recuperadas hasta el momento por el cursor.
* **%ISOPEN**, devuelve TRUE si el cursor está abierto.

En el siguiente ejemplo se muestran los nombres de los empleados del departamento y se numera cada uno de ellos:

DECLARE

CURSOR C20 IS SELECT APELLIDO FROM EMPLE WHERE DEPTNO=20;

NOMBRE VARCHAR(20);

BEGIN

OPEN C20;

FETCH C20 INTO NOMBRE;

WHILE C20%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE(C20%ROWCOUNT||’ – ‘ ||NOMBRE);

FETCH C20 INTO NOMBRE;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(‘SE HAN MOSTRADO ‘ || C20%ROWCOUNT||’ EMPLEADOS’);

CLOSE C20;

END;

Ejercicio 1. Diseñar un bloque que realice la misma tarea que el anterior pero en este caso se utilizará el bucle LOOP … EXIT WHEN.

En el ejemplo anterior se localizan exactamente los empleados del departamento 20, a veces el valor con el que hay que comprobar en la condición sólo se conoce en el momento de la ejecución o depende de otros resultados, para poder resolver estas situaciones se utilizan **variables de acoplamiento** en los cursores.

Ejemplo, procedimiento que muestra el nombre de los empleados de un departamento cuyo código entra como argumento:

CREATE OR REPLACE PROCEDURE VERNOMBRE (**CODIGO** NUMBER)

IS

CURSOR CX IS SELECT APELLIDO FROM EMPLE WHERE DEPTNO=**CODIGO**;

NOMBRE EMPLE.APELLIDO%TYPE;

BEGIN

OPEN CX;

FETCH CX INTO NOMBRE;

WHILE CX%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE(NOMBRE);

FETCH C20 INTO NOMBRE;

END LOOP;

END;

Ejercicio 2. Diseñar un procedimiento que reciba una cadena y visualice el apellido y el número de empleado de todos los empleados cuyo apellido contenga la cadena especificada. Al terminar se mostrará el número de empleados mostrados. Se utilizará una variable de acoplamiento.

Existe una estructura exclusiva para recorrer cursores, **FOR … LOOP** que evita la apertura del cursor, la declaración de una variable para recoger sus datos, la orden FETCH para ir accediendo a sus datos y el cierre del cursor. Todas estas operaciones se realizarán de manera implícita, lo único que hay que hacer es declarar dicho cursor.

Una vez declarado el cursor en la zona de declaraciones se procesará con la siguiente estructura:

**FOR varCursor IN nombreCursor LOOP**

**…**

**END;**

Donde varCursor es el nombre de la variable que se declara implícitamente y que permitirá recoger los datos del cursor. Al empezar el bucle el cursor se abrirá de manera automática y la orden FETCH también se irá ejecutando en cada pasada del bucle, hasta que no existan más filas en el cursor.

En el siguiente ejemplo se visualizan el apellido y el oficio de los empleados cuya comisión supera 500.

DECLARE

CURSOR COMi IS SELECT APELLIDO, OFICIO FROM EMPLE WHERE COMISION>500;

BEGIN

FOR VCOM IN COMi LOOP

DBMS\_OUTPUT.PUT\_LINE(VCOM.APELLIDO||’ ‘||VCOM.OFICIO);

END LOOP;

END;

Ejercicio 3. Realizar el ejercicio 2 utilizando el bucle FOR … LOOP. Ten en cuenta que el cursor está cerrado al salir del bucle y no están disponibles sus atributos (%ROWCOUNT)

Cuando las columnas que contiene la SELECT asociada al cursor son expresiones se utilizarán **alias** para poder ser referenciadas con la variable que recoja el contenido del cursor.

En el siguiente ejemplo el cursor contiene el código de cada departamento y el contador de empleados de dicho departamento, será necesario utilizar un alias para este último valor:

CURSOR C1 IS

SELECT DEPTNO, COUNT(\*) **CONTADOR** FROM EMPLE GROUP BY DEPTNO;

Al utilizar cursores, en lugar de variables de acoplamiento se pueden usar parámetros para que la sentencia SELECT asociada se ejecute con determinados valores que vamos a indicar en tiempo de ejecución. La declaración de estos cursores se realiza indicando la lista de parámetros entre paréntesis a continuación del nombre del cursor.

CURSOR nombreCursor (lista de parámetros) is SELECT (sentencia SELECT con parámetros)

La lista de parámetros del cursor tendrá la siguiente sintaxis:

nombreParametro tipoDato [{DEFAULT| :=} valor ]

Ejemplo: se declara un cursor que consulta el nombre y el salario de los empleados que pertenecen a un determinado departamento y a un determinado oficio, ambos, departamento y oficio son los parámetros del cursor.

CURSOR C1 (VDEPAR NUMBER, VOFICIO VARCHAR2 DEFAULT ‘DIRECTOR’) IS

SELECT apellido, salario FROM EMPLE

WHERE DEPTNO=VDEPAR AND OFICIO=VOFICIO;

Al abrir el cursor es cuando hay que pasar los parámetros. Cualquiera de las siguientes órdenes sería válida para el cursor anterior:

BEGIN

…

OPEN C1(VDEP);

OPEN C1(VDEP, VOFI);

OPEN C1(20, ‘VENDEDOR’);

…

En el caso de utilizar un cursor FOR …LOOP en el que no es necesario abrirlo, el paso de parámetros se hará a continuación del identificador del cursor:

FOR VC1 IN C1(20, ‘VENDEDOR’) LOOP

….