4. Manejo de Cursores

Este tema se centra en cursores, los cuales permiten manejar grupos de datos que se obtienen como resultado de una consulta SQL que retorna una o más filas.

PL/SQL utiliza dos tipos de cursores: implícitos y explícitos.

•PL/SQL utiliza cursores para gestionar las instrucciones **SELECT**. Un cursor es un conjunto de registros devuelto por una instrucción SQL. Técnicamente los cursores son fragmentos de memoria que son reservados para procesar los resultados de una consulta **SELECT**.

Podemos distinguir dos tipos de cursores:

•**Cursores implícitos**. Este tipo de cursores se utiliza para operaciones **SELECT INTO**. Se usan cuando la consulta devuelve un único registro.
•**Cursores explícitos**. Son los cursores que son declarados y controlados por el programador. Se utilizan cuando la consulta devuelve un conjunto de registros. Ocasionalmente también se utilizan en consultas que devuelven un único registro por razones de eficiencia. Son más rápidos.
• Un cursor se define como cualquier otra variable de PL/SQL y debe nombrarse de acuerdo a los mismos convenios que cualquier otra variable. Los cursores implícitos no necesitan declaración.

Cursores Implícitos

•Los cursores implícitos se utilizan para realizar consultas **SELECT** que devuelven un único registro.
•Deben tenerse en cuenta los siguientes puntos cuando se utilizan cursores implícitos:
–Con cada cursor implícito debe existir la palabra clave **INTO**.
–Las variables que reciben los datos devueltos por el cursor tienen que contener el mismo tipo de dato que las columnas de la tabla.
–Los cursores implícitos solo pueden devolver una única fila. En caso de que se devuelva más de una fila (o ninguna fila) producirá una excepción. PL/SQL gestiona los errores a través de las excepciones.

•El siguiente ejemplo muestra un cursor implícito:

```
declare
vdescripcion VARCHAR2(50);
begin
SELECT DESCRIPCION
INTO vdescripcion
from ASIGNATURA
WHERE COD_ASIGN = 'TSW';
dbms_output.put_line('La lectura del cursor es: ' || vdescripcion);
end;
```

La salida del programa generaría la siguiente línea:

La lectura del cursor es: TALLER DE SOFTWARE

•Los cursores implícitos sólo pueden devolver una fila, por lo que pueden producirse determinadas excepciones.
•Las más comunes que se pueden encontrar son **no_data_found** y **too_many_rows**.

–NO_DATA_FOUND: Se produce cuando una sentencia SELECT intenta recuperar datos pero ninguna fila satisface sus condiciones. Es decir, cuando *"no hay datos"*
–TOO_MANY_ROWS: Dado que cada cursor implícito sólo es capaz de recuperar una fila , esta excepción detecta la existencia de más de una fila.

Cursor Explícito

•Los Cursores Explícitos son aquellos que devuelven cero, una o más filas, dependiendo de los criterios con que hayan sido construidos. Un cursor puede ser declarado en la primera sección de un programa PL/SQL ("declare").

•Existen tres comandos para controlar un cursor: OPEN, FETCH y CLOSE.
•En un principio, el cursor se inicializa con la instrucción OPEN. Enseguida, se utiliza la instrucción FETCH para recuperar la primera fila o conjunto de datos. Se puede ejecutar FETCH repetidas veces hasta que todas las filas hayan sido recuperadas. Cuando la última fila ya ha sido procesada, el cursor se puede liberar con la sentencia CLOSE.
•Es posible procesar varias queries en paralelo, declarando y abriendo múltiples cursores

Declaración de Cursores
•Los cursores deben ser declarados antes de ser utilizados en otras sentencias. Cuando se declara un cursor, a éste se le da un nombre y se asocia con una consulta específica usando la sintaxis siguiente:

DECLARE
CURSOR *nombre_cursor* [(parámetro1 [, parámetro2]…)]
[RETURN *tipo_de_retorno*] IS *sentencia_select* ;

Donde *tipo_de_retorno* debe representar a un registro o una fila en una tabla de la base y los parámetros siguen la siguiente sintaxis:

nombre_del_parametro [IN] *tipo_de_dato* [{ := | DEFAULT} *expresión*]

•Por ejemplo, los cursores **c1** y **c2** se pueden declarar como sigue:

```
DECLARE
  CURSOR c1 IS
  SELECT empno, ename, job, sal FROM emp WHERE sal>1000 ;
```

```
  CURSOR c2 RETURN dept%ROWTYPE IS
       SELECT * from dept WHERE deptno = 10 ;
…
```

•El nombre del cursor (*c1* y *c2* en el ejemplo) corresponde a un identificador no declarado, no a un nombre de variable de PL/SQL. No se pueden asignar valores a un nombre de cursor ni utilizarlos en una expresión.
•Un cursor puede recibir parámetros, los cuales deben ser declarados con la cláusula IN para formalizar su incorporación. Los parámetros pueden ser inicializados con algún valor, pero estos pueden ser cambiados en cualquier oportunidad.
•El alcance de los parámetros es local al cursor, lo que significa que ellos sólo pueden ser referenciados dentro de la consulta especificada en la declaración del mismo. Estos valores son utilizados por la query cuando el cursor se abre.

Apertura de un Cursor

•Al abrir un cursor se ejecuta inmediatamente la consulta e identifica el conjunto resultado, el que consiste de todas las filas que concuerdan con el criterio de selección de éste. Para los cursores que se abren con la cláusula “For Update”, la sentencia de apertura del cursor además bloquea esas filas retornadas. Un ejemplo es el siguiente:

```
DECLARE
  CURSOR c1 IS SELECT ename, job FROM emp
  WHERE sal > 3000;
  …
BEGIN
  OPEN c1;
  …
END;
```

Paso de Parámetros
•Se utiliza también la sentencia OPEN para pasar los parámetros al cursor, en caso de que éste los requiera. Por ejemplo:

```
DECLARE
  emp_name          emp.name%TYPE;
  v_salary          emp.sal%TYPE;
  CURSOR c1 (name VARCHAR2, p_salary NUMBER) IS SELECT…where salario =
p_salary and …
```

Cualquiera de estas sentencias abre el cursor:
•OPEN c1(emp_name, 3000);
•OPEN c1(“John”, 1500);
•OPEN c1(emp_name, v_ salary);

Recuperación de Filas
•La sentencia FETCH permite recuperar los conjuntos de datos de a uno a la vez. Después de cada recuperación y carga de un set de datos el cursor avanza a la fila siguiente.
•Ejemplo:

FETCH c1 INTO my_empno, my_ename, my_deptno;

•Para cada columna retornada en un cursor y especificada en la declaración del mismo debe existir una variable compatible en tipo en la lista INTO.
Recuperación de Filas
•Típicamente se utiliza la sentencia FETCH dentro de un conjunto de instrucciones como el siguiente:

```
LOOP
  FETCH c1 INTO mi_registro;
   EXIT WHEN c1%NOTFOUND;
   --- se procesa el registro
END LOOP;
```

•Eventualmente, la sentencia FETCH fallará, es decir, no retornará ningún conjunto de datos. Cuando esto sucede, no se gatilla una excepción, luego se debe detectar esta condición utilizando los atributos del cursor %FOUND y/o %NOTFOUND.

Entonces…
•Para declarar un cursor debemos emplear la siguiente sintaxis:

CURSOR *nombre_cursor* IS *instrucción_SELECT*

•Con Parámetros:

CURSOR *nombre_cursor**(param1 tipo1, ..., paramN tipoN)* **IS** *instrucción_SELECT*

•Para abrir el cursor

OPEN *nombre_cursor;*

•o bien (en el caso de un cursor con parámetros)

OPEN *nombre_cursor**(valor1, valor2, ..., valorN);***

Para recuperar los datos en variables PL/SQL.

FETCH nombre_cursor INTO *lista_variables; -- o bien ...*
FETCH nombre_cursor INTO *registro_PL/SQL;   --ó*
FETCH nombre_cursor INTO *variable_cursor; -- (con uso de %rowtype)*

•Para cerrar el cursor:

CLOSE nombre_cursor;

Ejemplo
•El siguiente ejemplo ilustra el trabajo con un cursor explicito. Hay que tener en cuenta que al leer los datos del cursor debemos hacerlo sobre variables del mismo tipo de datos de la tabla (o tablas) que trata el cursor.

```
DECLARE
CURSOR cempleado
 IS
 SELECT COD_EMP, NOM_EMP
 FROM EMPLEADO;

  v_cod_emp VARCHAR2(3);
 v_nom_emp VARCHAR2(50);


BEGIN
  OPEN cempleado;
  FETCH cempleado INTO v_cod_emp,v_nom_emp;
  CLOSE cempleado;
END;
```

•Podemos simplificar el ejemplo utilizando el atributo de tipo **%ROWTYPE** sobre el cursor.

```
  DECLARE
  CURSOR cempleado
  IS
  SELECT COD_EMP, NOM_EMP, SALARIO
  FROM EMPLEADO;

  registro cempleado%ROWTYPE;

BEGIN
  OPEN cempleado;
  FETCH cempleado INTO registro;
  CLOSE cempleado;
END;
```

•El mismo ejemplo, pero utilizando parámetros:

```
DECLARE
  CURSOR cempleado (p_nom_emp VARCHAR2)
```

```
 IS
 SELECT COD_EMP, NOM_EMP, SALARIO
 FROM EMPLEADO
 WHERE NOM_EMP = p_nom_emp;

 registro cempleado%ROWTYPE;

BEGIN
 OPEN cempleado('CARLA');
 FETCH cempleado INTO registro;
 CLOSE cempleado;
END;
```

•Cuando trabajamos con cursores debemos considerar:
–Cuando un cursor está cerrado, no se puede leer.
–Cuando leemos un cursor debemos comprobar el resultado de la lectura utilizando los atributos de los cursores.
–Cuando se cierra el cursor, es ilegal tratar de usarlo.
–Es ilegal tratar de cerrar un cursor que ya está cerrado o no ha sido abierto

Atributos de cursores

•Toman los valores TRUE, FALSE o NULL dependiendo de la situación:

Atributo	Antes de abrir	Al abrir	Durante la recuperación	Al finalizar la recuperación	Después de cerrar
%NOTFOUND	ORA-1001	NULL	FALSE	TRUE	ORA-1001
%FOUND	ORA-1001	NULL	TRUE	FALSE	ORA-1001
%ISOPEN	FALSE	TRUE	TRUE	TRUE	FALSE
%ROWCOUNT	ORA-1001	0	*	**	ORA-1001

*: Número de registros que ha recuperado hasta el momento
**: Número total de registros

Uso de %FOUND

•Luego de que un cursor ha sido abierto, pero antes de recuperar la primera fila el valor del atributo %FOUND es nulo.

•A continuación, tomará el valor TRUE cada vez que obtenga una fila del set de resultados (en cada FETCH que se haga)

•Sólo alcanzará el valor FALSE cuando ya no existan más filas para mostrar en el set de resultados.

•Ejemplo:

```
LOOP
  FETCH c1 INTO …
  IF c1%FOUND THEN       -- fetch exitoso
        …
  ELSE
    EXIT; -- fetch falló; se sale del loop
  END IF;
END LOOP;
```

Uso de %NOTFOUND

•Es el opuesto lógico de %FOUND. Cada vez que una sentencia FETCH retorne una fila válida, este atributo devolverá FALSO. Sólo alcanzará el valor TRUE cuando no haya más filas en un cursor y se ejecute la sentencia FETCH (sin éxito).

•Ejemplo:

```
LOOP
  FETCH c1 INTO …
  EXIT WHEN c1%NOTFOUND;
  …
END LOOP;
```

Uso de %ISOPEN

Este atributo toma el valor verdadero (TRUE) cuando un cursor se encuentra abierto. De otra manera, retorna FALSO, por ejemplo:

```
  IF NOT c1%ISOPEN THEN
    OPEN C1;
  END IF;
```

Uso de %ROWCOUNT

•Cuando un cursor es abierto, este atributo es seteado en 0 (cero). En adelante, cada vez que se recuperen filas exitosamente con un FETCH, este valor se irá incrementando en uno.

•C1: nombre del cursor

Ej:

```
…
End loop;
Dbms_output.put_line(C1%rowcount);
Close c1;
…
```

Cierre de un Cursor

•La sentencia que deshabilita un cursor, CLOSE, se utiliza de la siguiente manera:

CLOSE c1;

•Una vez que un cursor ya ha sido cerrado, es posible volverlo a abrir sin tener que declararlo otra vez.
•Cualquier otra operación que se desee efectuar sobre un cursor no operativo (cerrado) provocará una excepción del tipo "invalid_cursor".

Manejo del Cursor

•Por medio de ciclo LOOP podemos iterar a través del cursor. Debe tenerse cuidado de agregar una condición para salir del bucle.
•Vamos a ver varias formas de iterar a través de un cursor.
•La primera es utilizando un bucle LOOP con una sentencia EXIT condicionada:

```
DECLARE
   CURSOR nombre_cursor
         IS
         SELECT …

     lista_variables TIPO_DATO;

BEGIN

         OPEN nombre_cursor;
         LOOP
                  FETCH nombre_cursor INTO lista_variables;
                  EXIT WHEN nombre_cursor%NOTFOUND;
        /* Procesamiento de los registros recuperados */
         END LOOP;
         CLOSE nombre_cursor;

END;
```

• Aplicada a nuestro ejemplo anterior:

```
SET SERVEROUTPUT ON;
DECLARE
CURSOR cempleado
IS
SELECT COD_EMP, NOM_EMP
FROM EMPLEADO;

v_cod_emp VARCHAR2(3);
v_nom_emp VARCHAR2(50);

BEGIN
 OPEN cempleado;
```

```
 LOOP
        FETCH cempleado INTO v_cod_emp, v_nom_emp;
        EXIT WHEN cempleado%NOTFOUND;
        dbms_output.put_line(v_nom_emp);
END LOOP;
CLOSE cempleado;
END;
```

• Otra forma es por medio de un bucle WHILE LOOP. La instrucción FETCH aparece dos veces.

```
OPEN nombre_cursor;
 FETCH nombre_cursor INTO lista_variables;
 WHILE nombre_cursor%FOUND
LOOP
/* Procesamiento de los registros recuperados */
   FETCH nombre_cursor INTO lista_variables;
END LOOP;
CLOSE nombre_cursor;
```

```
SET SERVEROUTPUT ON;
DECLARE
CURSOR cempleado
IS
SELECT COD_EMP, NOM_EMP
FROM EMPLEADO;

v_cod_emp VARCHAR2(3);
v_nom_emp VARCHAR2(50);

BEGIN
  OPEN cempleado;
  FETCH cempleado INTO v_cod_emp, v_nom_emp;
  WHILE cempleado%found
  LOOP
        DBMS_OUTPUT.PUT_LINE(v_nom_emp);
        FETCH cempleado INTO v_cod_emp, v_nom_emp;
  END LOOP;
  CLOSE cempleado;
END;
```

•Por último podemos usar un bucle FOR LOOP.
•Es la forma más corta ya que el cursor implícitamente ejecuta las instrucciones OPEN, FETCH y CLOSE.

```
    FOR variable IN nombre_cursor
     LOOP
     /* Procesamiento de los registros recuperados */
     END LOOP;

BEGIN
FOR REG IN (SELECT * FROM EMPLEADO)
LOOP
dbms_output.put_line(reg.nom_emp);
END LOOP;
END;
```

Cursores de actualización

•Los cursores de actualización se declaran igual que los cursores explícitos, añadiendo **FOR UPDATE** al final de la sentencia select.

CURSOR ***nombre_cursor***
IS ***instrucción_SELECT***
FOR UPDATE

•Para actualizar los datos del cursor hay que ejecutar una sentencia **UPDATE** especificando la cláusula **WHERE CURRENT OF** <*cursor_name*>.

UPDATE <*nombre_tabla*>
SET
<*campo_1*> = <*valor_1*>
[,<*campo_2*> = <*valor_2*>]
WHERE CURRENT OF <*cursor_name*>

•El siguiente ejemplo muestra el uso de un cursor de actualización:

```
DECLARE
        CURSOR empleado IS
        select COD_EMP, NOM_EMP
        from EMPLEADO
        FOR UPDATE;
        v_cod_emp VARCHAR2(3);
        v_nom_emp VARCHAR2(50);

    BEGIN
        OPEN empleado;
        FETCH empleado INTO v_cod_emp, v_nom_emp;
        WHILE empleado%found
        LOOP
                UPDATE EMPLEADO
                SET NOM_EMP = NOM_EMP || '.'
                WHERE CURRENT OF empleado;

                FETCH empleado INTO v_cod_emp, v_nom_emp;
        END LOOP;
        CLOSE empleado;
        COMMIT;

END;
```

Cuando trabajamos con cursores de actualización debemos tener en cuenta:
–Los cursores de actualización generan bloqueos en la base de datos.