TEMA 7. CURSORES EN PL/SQL.

1. INTRODUCCIÓN.

Un cursor no es más que una estructura que almacena el conjunto de filas devuelto por una consulta a la base de datos.

Oracle usa áreas de trabajo para ejecutar sentencias SQL y almacenar la información procesada. Hay 2 clases de cursores: **implícitos** y **explícitos**.

PL/SQL declara implícitamente un cursor para todas las sentencias SQL de manipulación de datos, incluyendo consultas que devuelven **una sola fila** (Los que hemos visto en el tema anterior). **Para las consultas que devuelven más de una fila, se debe declarar explícitamente un cursor** para procesar las filas individualmente.

Oracle abre un cursor **implícito** para procesar cada sentencia SQL que no esté asociada con un cursor declarado explícitamente. Con un cursor implícito no podemos usar las sentencias `OPEN`, `FETCH` y `CLOSE` para controlar el cursor. Pero sí podemos usar los atributos del cursor para obtener información sobre las sentencias SQL más recientemente ejecutadas.

2. ATRIBUTOS DE UN CURSOR.

Cada cursor tiene 4 atributos que podemos usar para obtener información sobre la ejecución del mismo o sobre los datos. Estos atributos pueden ser usados en PL/SQL, pero no en SQL. Aunque estos atributos se refieren en general a cursores explícitos y tienen que ver con las operaciones que hayamos realizado con el cursor, es deseable comentarlas aquí y en el siguiente apartado tomarán pleno sentido.

- ✓ `%FOUND`: Después de que el cursor esté abierto y antes del primer `FETCH`, `%FOUND` devuelve `NULL`. Después del primer `FETCH`, `%FOUND` devolverá `TRUE` si el último `FETCH` ha devuelto una fila y `FALSE` en caso contrario. Para cursores implícitos `%FOUND` devuelve `TRUE` si un `INSERT`, `UPDATE` o `DELETE` afectan a una o más de una fila, o un `SELECT ... INTO ...` devuelve una o más filas. En otro caso `%FOUND` devuelve `FALSE`.
- ✓ `%NOTFOUND`: Es lógicamente lo contrario a `%FOUND`.
- ✓ `%ISOPEN`: Evalúa a `TRUE` si el cursor está abierto y `FALSE` en caso contrario. Para cursores implícitos, como Oracle los cierra automáticamente.
- ✓ `%ROWCOUNT`: Para un cursor abierto y antes del primer `FETCH`, `%ROWCOUNT` evalúa a 0. Después de cada `FETCH`, `%ROWCOUNT` es incrementado y evalúa al número de filas que hemos procesado. Para cursores implícitos `%ROWCOUNT` evalúa al número de filas afectadas por un `INSERT`, `UPDATE` o `DELETE` o el número de filas devueltas por un `SELECT ... INTO ...`

La evaluación de los atributos según las operaciones que hayamos realizado con el cursor son :				
Operación realizada.	**%FOUND**	**%NOTFOUND**	**%ISOPEN**	**%ROWCOUNT**
Antes del OPEN	Excepción.	Excepción.	FALSE	Excepción.
Después del OPEN	NULL	NULL	TRUE	0
Antes del primer FETCH	NULL	NULL	TRUE	0
Después del primer FETCH	TRUE	FALSE	TRUE	1
Antes de los siguientes FETCH	TRUE	FALSE	TRUE	1
Después de los siguientes FETCH	TRUE	FALSE	TRUE	Depende datos.
Antes del último FETCH	TRUE	FALSE	TRUE	Depende datos.
Después del último FETCH	FALSE	TRUE	TRUE	Depende datos.
Antes del CLOSE	FALSE	TRUE	TRUE	Depende datos.
Después del CLOSE	Excepción.	Excepción.	FALSE	Excepción.

3. CURSORES EXPLÍCITOS.

Cuando una consulta devuelve múltiples filas, podemos almacenar el resultado de la consulta, declarando explícitamente un cursor para procesar las filas devueltas. Cuando declaramos un cursor, lo que hacemos es darle un nombre y asociarle una consulta usando la siguiente sintaxis:

```
CURSOR nombre_cursor [(parametro [, parametro] ...)] [RETURN tipo_devuelto] IS sentencia_select;
```

Donde tipo_devuelto debe representar un registro o una fila de una tabla de la base de datos, y parámetro sigue la siguiente sintaxis:

```
parametro := nombre_parametro [IN] tipo_dato [{:= | DEFAULT} expresion]
```

Ejemplos:

```
CURSOR cAgentes IS SELECT * FROM agentes;
CURSOR cFamilias RETURN familias%ROWTYPE IS SELECT * FROM familias WHERE ...
```

Además, como hemos visto en la declaración, un cursor puede tomar parámetros, los cuales pueden aparecer en la consulta asociada como si fuesen constantes. Los parámetros serán de entrada, un cursor no puede devolver valores en los parámetros actuales. A un parámetro de un cursor no podemos imponerle la restricción `NOT NULL`.

```
CURSOR c1 (cat INTEGER DEFAULT 0) IS SELECT * FROM agentes WHERE categoria = cat;
```

Cuando abrimos un cursor, lo que se hace es ejecutar la consulta asociada e identificar el conjunto resultado, que serán todas las filas que emparejen con el criterio de búsqueda de la consulta. Para abrir un cursor usamos la sintaxis:

```
OPEN nombre_cursor [(parametro [, parametro] ...)];
```

Ejemplos:

```
OPEN cAgentes;
OPEN c1(1);
OPEN c1;
```

Para poder acceder a la información que contiene el cursor utilizamos la instrucción **FETCH**:

```
FETCH nombrecursor INTO <variable>|<listavariables>;
```

Después del INTO figurará:

- ✓ una variable que recogerá la información de todas las columnas. En este caso la variable puede ser declarada de la forma:

```
<variable>  <nombrecursor>%ROWTYPE
```

¿Qué sucede cuando la consulta obtiene información de varias tablas? ¿Podemos declarar una variable que nos permita identificar y almacenar datos concretos de diferentes tablas?

Para procesar un cursor entero deberemos hacerlo por medio de un bucle.

```
DECLARE
   REG DEPART%ROWTYPE;
   CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
  OPEN EJEM;
  LOOP
    FETCH EJEM INTO REG;
    EXIT WHEN EJEM %NOTFOUND;
    DBMS_OUTPUT.PUT_LINE ('EL NOMBRE DEL DEPARTAMENTO ES ' || REG.DNOMBRE);
  END LOOP;
  CLOSE EJEM;
END;
```

- ✓ o bien un conjunto de variables. Después de cada `FETCH`, el cursor avanza a la próxima fila en el conjunto resultado, igual que en el caso anterior.

```
FETCH EJEM INTO mi_id, mi_nom, mi_LOC;
```

Para cada valor de columna devuelto por la consulta asociada al cursor, debe haber una variable que se corresponda en la lista de variables después del INTO.

Para procesar un cursor entero deberemos hacerlo por medio de un bucle.

```
DECLARE
   MI_ID DEPART.DEPT_NO%TYPE;
   MI_NOM DEPART.DNOMBRE%TYPE;
   MI_LOC DEPART.LOC%TYPE;
   CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
  OPEN EJEM;
    LOOP
      FETCH EJEM INTO MI_ID, MI_NOM, MI_LOC;
      EXIT WHEN EJEM %NOTFOUND;
      DBMS_OUTPUT.PUT_LINE ('EL NOMBRE DEL DEPARTAMENTO ES ' || MI_NOM);
    END LOOP;
  CLOSE EJEM;
END;
```

Una vez procesado el cursor, deberemos cerrarlo, con lo que deshabilitamos el cursor y el conjunto resultado queda indefinido.

```
CLOSE EJEM;
```

Una vez cerrado el cursor podemos reabrirlo, pero cualquier otra operación que hagamos con el cursor cerrado lanzará la excepción `INVALID_CURSOR`.

Un cursor se puede procesar de diferentes formas, pero siempre dentro de un proceso repetitivo. Podemos simplificar la operación de procesamiento de un cursor, por medio de los bucles para cursores, los cuales declaran implícitamente una variable índice definida como `%ROWTYPE` para el cursor, abren el cursor, se van trayendo los valores de cada fila del cursor, almacenándolas en la variable índice, y finalmente cierran el cursor.

```
DECLARE
   CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
  FOR EJEM_REG IN EJEM LOOP
     DBMS_OUTPUT.PUT_LINE ('EL NOMBRE DEL DEPARTAMENTO ES ' || EJEM_REG.DNOMBRE);
  END LOOP;
END;
```

Este es el método más sencillo, ya que no hace falta abrir ni cerrar el cursor, tampoco hace falta que contralar si llegamos al final del cursor. Además, nos permite manejar fácilmente los datos, cuando la consulta devuelve información de varias tablas.

4. EJEMPLOS CON CURSORES CON DIFERENTES PROCESOS REPETITIVOS:

4.1. UTILIZANDO EL BUCLE LOOP

```
DECLARE
        REG DEPART%ROWTYPE;
        CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
      OPEN EJEM;
```

```
        LOOP
                FETCH EJEM INTO REG;
                EXIT WHEN EJEM%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('NUM:'||TO_CHAR(REG.DEPT_NO));
                DBMS_OUTPUT.PUT_LINE ('NOMBRE:' || REG.DNOMBRE);
                DBMS_OUTPUT.PUT_LINE('LOCALIDAD:' || REG.LOC);
        END LOOP;
        CLOSE EJEM;
END;
```

4.2. UTILIZANDO UN BUCLE WHILE

1. Vamos a leer de la tabla los datos y los vamos a almacenar en un cursor.
2. Declara una estructura %rowtype. Utiliza un cursor.
3. El cursor se utiliza para recorrer las tablas fila a fila.

```
DECLARE
        REG DEPART%ROWTYPE;
        CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
        OPEN EJEM;

        FETCH EJEM INTO REG;
        WHILE EJEM%FOUND   LOOP
                DBMS_OUTPUT.PUT_LINE('NUM:'||TO_CHAR(REG.DEPT_NO));
                DBMS_OUTPUT.PUT_LINE('NOMBRE:' || REG.DNOMBRE);
                DBMS_OUTPUT.PUT_LINE('LOCALIDAD:' || REG.LOC);
                FETCH EJEM INTO REG;
        END LOOP;
        CLOSE EJEM;
END;
```

4.3. UTILIZANDO EL BUCLE FOR

```
DECLARE
  CURSOR EJEM IS SELECT DEPT_NO, DNOMBRE, LOC FROM DEPART;
BEGIN
  FOR REG IN EJEM LOOP
        DBMS_OUTPUT.PUT_LINE('NUM:'||TO_CHAR(REG.DEPT_NO));
    DBMS_OUTPUT.PUT_LINE ('NOMBRE:' || REG.DNOMBRE);
    DBMS_OUTPUT.PUT_LINE ('LOCALIDAD:' || REG.LOC);
    DBMS_OUTPUT.PUT_LINE ('');
  END LOOP;
END;
```

5. UN EJEMPLO:

Realizar un programa que visualice por cada departamento el salario máximo y el/los empleados con ese salario.

```
DEPT_NO   NOMBRE DEPARTAMENTO SALARIO_MÁXIMO NOMBRE EMPLEADO
--------- ---------------------------------- ---------------------- ----------------------------
xxxxxx          xxxxxxxxxxxxxxxxx          xxxxxxxxxxxxx          xxxxxxxxxxx
xxxxxx          xxxxxxxxxxxxxxxxx          xxxxxxxxxxxxx          xxxxxxxxxxx

NOMBRE DEPARTAMENTO CON EL MÁXIMO SALARIO:    DEPMAXSAL,  SALMAXDEP
```

```
Select Dd.Dept_No,  Dnombre, Salario, Apellido
    From Depart Dd Join Emple On (Dd.Dept_No = Emple.Dept_No)
      Where Salario = ( Select Max(Salario)
            From Emple Where Dept_No = Dd.Dept_No );
```

DECLARE

Declaración de variables y del cursor.

BEGIN

CABECERA, PARTE INICIAL DEL INFORME

Inicializar los totales, si se necesita

Escribir las cabeceras

abrir cursor

Hacer la 1ª lectura de los reg del cursor

DETALLE, TRATAMIENTO DE LOS DATOS.

Bucle de tratamiento del cursor

Mientras tengamos datos en el cursor

Realizar cálculos si se necesitan, para la línea de detalle

Visualizar si se pide el detalle (reg del cursor)

Realizar cálculos que se necesiten para las líneas de pie o de totales

volver a leer un reg del cursor

fin del bucle

LÍNEAS DE TOTALES.

Se ponen totales que pida el informe, que se refiere a las filas tratadas

Hacer cálculos si se necesitan

Visualizar totales

cerramos cursor

fin

EXCEPTION

Si controlamos alguna excepción.

END;

```
DECLARE
        CURSOR C1 IS
           SELECT DD.DEPT_NO,  DNOMBRE, SALARIO, APELLIDO
             FROM DEPART DD JOIN EMPLE ON (DD.DEPT_NO = EMPLE.DEPT_NO)
                WHERE SALARIO = (SELECT MAX(SALARIO)
                                 FROM EMPLE WHERE DEPT_NO = DD.DEPT_NO);
          REG C1%ROWTYPE;
```

```
          DEPMAXSAL DEPART.DNOMBRE%TYPE;
          SALMAXDEP EMPLE.SALARIO%TYPE;
BEGIN
        -- CABECERAS
        SALMAXDEP:=0;
        DBMS_OUTPUT.PUT_LINE('DEPT_NO NOMBRE DEPARTAMENTO SALARIO_MÁXIMO NOMBRE EMPLEADO');
        DBMS_OUTPUT.PUT_LINE('------- ------------------ -------------- ---------------');
        OPEN C1;
        FETCH C1 INTO REG;
        -- DETALLE
           WHILE C1%FOUND LOOP
              DBMS_OUTPUT.PUT_LINE(RPAD(TO_CHAR(REG.DEPT_NO), 7)|| ' '||
               RPAD(REG.DNOMBRE, 19) || ' ' ||
               RPAD(TO_CHAR(REG.SALARIO),14) || ' ' ||REG.APELLIDO);
              IF REG.SALARIO > SALMAXDEP THEN
                  SALMAXDEP := REG.SALARIO ;
                  DEPMAXSAL := REG.DNOMBRE;
              END IF;
              FETCH C1 INTO REG;
           END LOOP;
         -- LÍNEAS DE TOTALES O PIE
          DBMS_OUTPUT.PUT_LINE('------- ------------------ -------------- ---------------');
          DBMS_OUTPUT.PUT_LINE('NOMBRE DEPARTAMENTO CON EL MÁXIMO SALARIO: '|| DEPMAXSAL);
          CLOSE C1;
END;
```

REALIZAR EJERCICIOS1 CURSORES

6. CURSORES CON PARÁMETROS.

Un cursor puede tener parámetros, para ello la sintaxis es:

CURSOR nombrecursor [(parámetro1, parámetro2,...)] IS
SELECT <sentencia select en la que
intervendrán los parámetros>;

Los parámetros formales indicados después del nombre del cursor tienen la siguiente sintaxis:

Nombredevariable [IN] tipodedato [{ := | DEFAULT } valor]

Todos los parámetros formales de un cursor son parámetros de entrada por ello se indica IN. El ámbito de dichos parámetros es local al cursor, por ello solamente pueden ser referenciados dentro de la consulta.

```
DECLARE
        ...
        CURSOR c1
                (menor NUMBER DEFAULT 0,
                 mayor NUMBER DEFAULT 999999)
                IS SELECT apellido FROM emple
WHERE salario BETWEEN menor AND mayor;
```

Para abrir un cursor pasándole parámetros lo haremos:

OPEN nombrecursor [(parámetro1, parámetro2, ...)];

Donde parámetro1, parámetro2, son expresiones que contienen los valores que se pasarán al cursor. No tienen por qué ser los mismos nombres de las variables indicadas como parámetros al declarar el cursor, es más, si lo fueran, serían consideradas como variables distintas.

```
DECLARE
  emp_name emple.apellido%TYPE;
  salary   emple.salario%TYPE;
  CURSOR c1 (name VARCHAR2, salary NUMBER) IS SELECT …
```

Cualquiera de los siguientes comandos abrirá el cursor:

```
OPEN c1(emp_name, 3000);
OPEN c1('Pepito', 1500);
OPEN c1(emp_name, salary);
```

La recogida de datos se hará como ya vimos con la orden FETCH.

Conviene señalar que la cláusula WHERE (y las variables que en ella intervienen) asociada al cursor se evalúa solamente en el momento de abrir el cursor.

Ejemplo:

Hacer un bloque PL que lea de teclado un departamento, y visualice los empleados de ese departamento.

La visualización será así:

```
Listado de empleados del departamento:10
APELLIDO        OFICIO          SALARIO   DEPARTAMENTO
--------------- --------------- --------- ------------
CEREZO          DIRECTOR        2885        10
REY             PRESIDENTE      4100        10
MUÑOZ           EMPLEADO        1690        10
```

Solución:

```
DECLARE
    CURSOR CEMPLEADOS(DPTO NUMBER)IS SELECT * FROM EMPLE WHERE DEPT_NO=DPTO;
    VARCURSOR CEMPLEADOS %ROWTYPE;
    INTRODEPT NUMBER :=&DEPARTAMENTO;
    EXISTE NUMBER(1);
BEGIN
    SELECT COUNT(*) INTO EXISTE FROM DEPART WHERE DEPT_NO=INTRODEPT;
    IF EXISTE=0 THEN
      DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NÚMERO ' || INTRODEPT || ' NO EXISTE');
    ELSE
      DBMS_OUTPUT.PUT_LINE('LISTADO DE EMPLEADOS DEL DEPARTAMENTO ' || INTRODEPT);
      DBMS_OUTPUT.PUT_LINE('APELLIDOS   OFICIO       SALARIO     DEPARTAMENTO');
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      OPEN CEMPLEADOS(INTRODEPT);
      FETCH CEMPLEADOS INTO VARCURSOR;
      WHILE(CEMPLEADOS %FOUND)LOOP
       DBMS_OUTPUT.PUT_LINE(RPAD(VARCURSOR.APELLIDO,12) ||
                            RPAD(VARCURSOR.OFICIO, 15) ||
                            RPAD(VARCURSOR.SALARIO, 12)||
                            VARCURSOR.DEPT_NO);
         FETCH CEMPLEADOS INTO VARCURSOR;
      END LOOP;
      CLOSE CEMPLEADOS;
    END IF;
END;
```

Hacer que además se visualice el salario medio.

```
DECLARE
    CURSOR CEMPLEADOS(DPTO NUMBER)IS SELECT * FROM EMPLE WHERE DEPT_NO=DPTO;
    VARCURSOR CEMPLEADOS %ROWTYPE;
    INTRODEPT NUMBER :=&DEPARTAMENTO;
```

```
    EXISTE NUMBER(1);
    SUMASAL NUMBER :=0;
    CONT NUMBER:=0;
BEGIN
    SELECT COUNT(*) INTO EXISTE FROM DEPART WHERE DEPT_NO=INTRODEPT;
    IF EXISTE=0 THEN
      DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NÚMERO ' || INTRODEPT || ' NO EXISTE');
    ELSE
      DBMS_OUTPUT.PUT_LINE('LISTADO DE EMPLEADOS DEL DEPARTAMENTO ' || INTRODEPT);
      DBMS_OUTPUT.PUT_LINE('APELLIDOS   OFICIO       SALARIO     DEPARTAMENTO');
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      OPEN CEMPLEADOS(INTRODEPT);
      FETCH CEMPLEADOS INTO VARCURSOR;
      WHILE(CEMPLEADOS %FOUND)LOOP
       DBMS_OUTPUT.PUT_LINE(RPAD(VARCURSOR.APELLIDO,12) ||
                            RPAD(VARCURSOR.OFICIO, 15) ||
                            RPAD(VARCURSOR.SALARIO, 12)||
                            VARCURSOR.DEPT_NO);
         SUMASAL := SUMASAL + VARCURSOR.SALARIO;
         CONT := CONT +1;
         FETCH CEMPLEADOS INTO VARCURSOR;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
                DBMS_OUTPUT.PUT_LINE('LA   MEDIA   DE   LOS   SALARIOS   ES:   '   ||
TO_CHAR((SUMASAL/CONT),'9G999D99'));
      DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
      CLOSE CEMPLEADOS;
    END IF;
END;
```