# 1.- Cursores

Cuando dentro de un intérprete SQL escribimos una consulta SELECT, el intérprete nos muestra las distintas filas del resultado para que podamos verlas. Sin embargo, cuando usamos procedimientos almacenados o funciones ORACLE tenemos un problema, ya que lo más común no es mostrar el resultado, sino almacenarlo en variables para su posterior tratamiento. Para solucionar este problema utilizamos los cursores.

Por lo tanto los cursores son la herramienta que vamos a utilizar cuando queramos procesar un conjunto de registros dentro de un procedimiento o función.

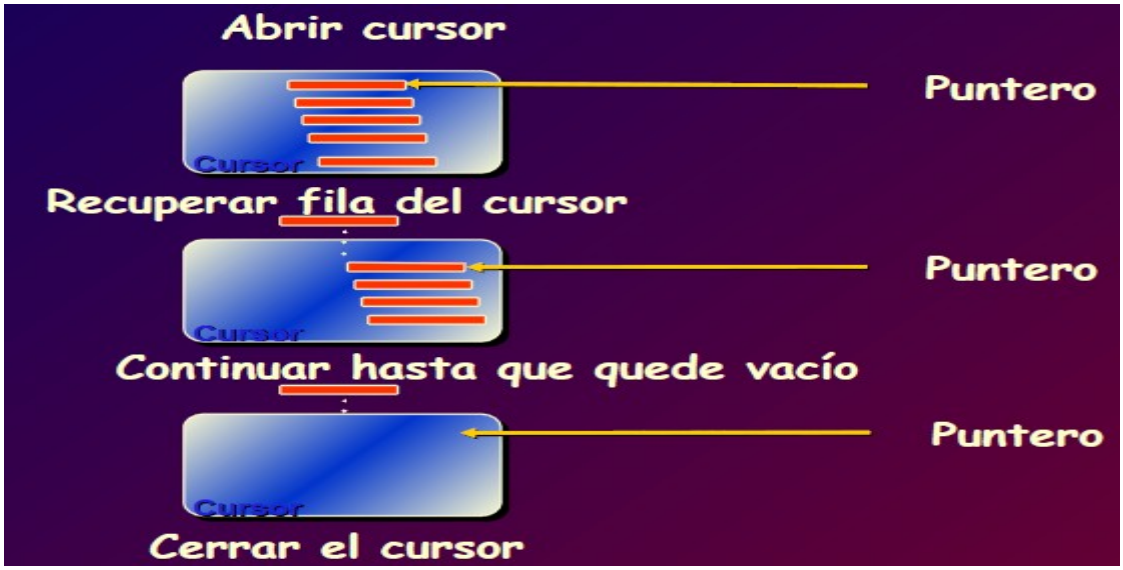
Podemos distinguir dos tipos de cursores:

1. **Cursores implícitos**. Se usan cuando la consulta devuelve un único registro.
2. **Cursores explícitos**. Se usan 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.

## 1.1.-Procesamiento de cursores

Para llevar a cabo el procesamiento de un cursor es necesario realizar los siguientes 4 pasos:

1. Declarar el cursor (dentro de la sección DECLARE del procedimiento o funcion ORACLE).
2. Abrir el cursor.
3. Recuperar cada una de las filas (bucle).
4. Cerrar el cursor.




### 1.1.1.- Paso 1: Declarar el cursor

Define el nombre que tendrá el cursor y qué consulta SELECT ejecutará. Su sintaxis es la siguiente:

```
CURSOR nombre_del_cursor IS Consulta_SELECT;
```

nombre_del_cursor es el nombre que identificará al cursor.
Consulta_SELECT es la consulta que debe ser procesada.

Ejemplo:

```
CURSOR emp_cursor IS
SELECT employee_id, last_name
FROM employees;


CURSOR dept_cursor IS
SELECT *
FROM departments
WHERE department_id = 10;
BEGIN
...
```

Una vez que el cursor ya está declarado, podrá ser usado dentro del bloque de código.

### 1.1.2.-Paso 2: Abrir el cursor

La sintaxis para la apertura de un cursor es la siguiente:

```
OPEN nombre_del_cursor;
```

Abrir un cursor significa:

a) Reservar memoria suficiente para el cursor
b) Ejecutar la sentencia SELECT a la que se refiere el cursor
c) Colocar el puntero de recorrido de registros en la primera fila.

Si la sentencia SELECT del cursor no devuelve registros, Oracle no devolverá una excepción. Hasta intentar leer no sabremos si hay o no resultados.

Una vez abierto no podrá volver a abrirse hasta que no sea cerrado.

### 1.1.3.-Recuperar cada una de sus filas

Para ello usamos la siguiente sintaxis:

```
FETCH nombre_del_cursor INTO lista_de_variables;
```

Esta instrucción almacena el contenido de la fila a la que apunta actualmente el puntero en la lista de variables indicada. La lista de variables tiene tener el mismo tipo y número que las columnas representadas en el cursor (por supuesto el orden de las variables se tiene que corresponder con la lista de columnas). Tras esta instrucción el puntero de registros avanza a la siguiente fila (si la hay).

Ejemplo:

```
FETCH cursorProvincias INTO v_nombre, v_poblacion;
```

Una instrucción FETCH lee una sola fila y su contenido lo almacena en variables. Por ello se usa siempre dentro de bucles a fin de poder leer todas las filas de un cursor:

```
LOOP
      FETCH cursorProvincias INTO (v_nombre, v_poblacion);
      EXIT WHEN... --aquí se pondría la condición de salida
      ... --instrucciones de proceso de los datos del cursor
END LOOP;
```

### 1.1.4.-Cerrar el cursor

Para cerrar el cursor usamos la siguiente sintaxis:

```
CLOSE nombre_del_cursor;
```

Al cerrar el cursor se libera la memoria que ocupa y se impide su procesamiento. Tras cerrarlo se podría abrir de nuevo.

## 1.2.-Atributos de los cursores

Para poder procesar adecuadamente los cursores se pueden utilizar una serie de atributos que devuelven **verdadero** o **falso** según la situación actual del cursor. Estos atributos facilitan la manipulación del cursor. Se utilizan indicando el nombre del cursor, el símbolo % e inmediatamente el nombre del atributo a valorar (por ejemplo ***cursorProvincias%ISOPEN***)

a) **%ISOPEN**: Devuelve verdadero si el cursor ya está abierto.
b) **%NOTFOUND**: Devuelve verdadero si la última instrucción FETCH no devolvió ningún valor.
Ejemplo:

```
CURSOR cursorProvincias IS
SELECT p.nombre, SUM(poblacion) AS poblacion
FROM LOCALIDADES l
JOIN PROVINCIAS p USING (n_provincia)
GROUP BY p.nombre;
v_nombre PROVINCIAS.nombre%TYPE;
v_poblacion LOCALIDADES.poblacion%TYPE;
BEGIN
   OPEN cursorProvincias;
   LOOP
      FETCH cursorProvincias INTO v_nombre, v_poblacion;
      EXIT WHEN cursorProvincias%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(v_nombre    || ',' || v_poblacion);
   END LOOP;
   CLOSE cursorProvincias;
END;
```

En el ejemplo anterior se recorre el cursor hasta que el FETCH no devuelve ninguna fila. Lo que significa que el programa anterior muestra el nombre de cada provincia seguida de una coma y de la población de la misma.

c) **%FOUND**: Instrucción contraria a la anterior, devuelve verdadero si el último FETCH devolvió una fila.

d) **%ROWCOUNT**: Indica el número de filas que se han recorrido en el cursor (inicialmente vale cero). Es decir, indica cuántos FETCH se han aplicado sobre el cursor.

## 1.3.-Variables de registro

Los registros son una estructura estática de datos presente en casi todos los lenguajes clásicos (**record** en Pascal o **struct** en C). Se trata de un tipo de datos que se compone de datos más simple. Por ejemplo el registro ***persona*** se compondría de los datos simples ***nombre, apellidos, dirección, fecha de nacimiento,*** etc.

En PL/SQL su interés radica en que cada fila de una tabla o vista se puede interpretar como un registro, ya que cada fila se compone de datos simples. Gracias a esta interpretación, los registros facilitan la manipulación de los cursores ya que podemos entender que un cursor es un conjunto de registros (cada registro sería una fila del cursor).

### 1.3.1.-Declaración

Para utilizar registros, primero hay que definir los datos que componen al registro. Así se define el tipo de registro (por eso se utiliza la palabra TYPE). Después se declarará una variable de registro que sea del tipo declarado (es decir, puede haber varias variables del mismo tipo de registro).

Sintaxis:

```
TYPE nombreTipoRegistro IS RECORD(
campo1 tipoCampo1 [:= valorInicial],
campo2 tipoCampo2 [:= valorInicial],
…
campoN tipoCampoN [:= valorInicial]
);
nombreVariableDeRegistro nombreTipoRegistro;
```

Ejemplo:

```
TYPE regPersona IS RECORD(
nombre VARCHAR2(25),
apellido1 VARCHAR2(25),
apellido2 VARCHAR2(25),
fecha_nac DATE
```

```
laura regPersona;
```

### 1.3.2.-Uso de %ROWTYPE

Al declarar registros, se puede utilizar el modificador %ROWTYPE que sirve para asignar a un registro la estructura de una tabla.

Por ejemplo:

```
regPersona personas%ROWTYPE;
```

```
laura regPersona;
```

personas debe ser una tabla. regPersona es un registro que constará de los mismos campos y tipos que las columnas de la tabla personas.

### 1.3.3.-Uso de registros

Para rellenar los valores de los registros se indica el nombre de la variable de registro seguida de un punto y el nombre del campo a rellenar:

## 1.4.-Cursores y registros

### 1.4.1.-Uso de FETCH con registros

Una de las desventajas, con lo visto hasta ahora, de utilizar **FETCH** reside en que necesitamos asignar todos los valores de cada fila del cursor a una variable. Por lo que si una fila tiene 10 columnas, habrá que declarar 10 variables.

En lugar de ello se puede utilizar una variable de registro y asignar el resultado de **FETCH** a esa variable.

Ejemplo (equivalente al %NOTFOUND):

## 1.5.-Bucle FOR de recorrido de cursores

Es la forma más habitual de recorrer todas las filas de un cursor. Es un bucle FOR que se encarga de realizar tres tareas:

1. Abre un cursor (realiza un OPEN implícito sobre el cursor antes de empezar el bucle)
2. Recorre todas las filas de un cursor (cada vez que se entra en el interior del FOR se genera un FETCH implícito) y en cada vuelta del bucle almacena el contenido de cada fila en una variable de registro. La variable de registro utilizada en el bucle FOR no se debe declarar en la zona DECLARE; se crea al inicio del bucle y se elimina cuando éste finaliza.
3. Cierra el cursor (cuando finaliza el FOR)

Sintaxis:

```
FOR variableRegistro IN cursor LOOP
        ..instrucciones
END LOOP;
```

Esa sintaxis es equivalente a:

```
OPEN cursor;
LOOP
        FETCH cursor INTO variableRegistro;
        EXIT WHEN cursor%NOTFOUND;
        ...instrucciones
END LOOP;
```

Ejemplo (equivalente al ejemplo comentado en los apartados anteriores):

```
CURSOR cursorProvincias IS
        SELECT p.nombre, SUM(poblacion) AS poblacion
        FROM LOCALIDADES l
        JOIN PROVINCIAS p USING (n_provincia)
        GROUP BY p.nombre;
```

```
                FOR rProvincias IN cursorProvincias LOOP
```

```
                DBMS_OUTPUT.PUT_LINE(rProvincias.nombre || ',' ||

                END LOOP;
END;
```

Naturalmente este código es más sencillo de utilizar y más corto que los anteriores.

## 1.6.-Actualizaciones al recorrer registros con cursores

En muchas ocasiones se realizan operaciones de actualización de registros sobre el cursor que se está recorriendo. Para evitar problemas se deben bloquear los registros del cursor a fin de detener otros procesos que también desearan modificar los datos.

Esta cláusula se coloca al final de la sentencia SELECT del cursor (iría detrás del ORDER BY). Opcionalmente se puede colocar el texto **NOWAIT** para que el programa no se quede esperando en caso de que la tabla esté bloqueada por otro usuario. Se usa el texto **OF** seguido del nombre del campo que se modificará (no es necesaria esa cláusula, pero se mantiene para clarificar el código).

Sintaxis:

```
CURSOR ...
SELECT...
FOR UPDATE [OF campo] [NOWAIT]
```

Ejemplo:

```
CURSOR c_emp IS
        SELECT id_emp, nombre, n_departamento, salario
        FROM empleados, departamentos
        WHERE empleados.id_dep=departamentos.id_dep
        AND empleados.id_dep=80
        FOR UPDATE OF salario NOWAIT;
```

A continuación en la instrucción UPDATE que modifica los registros se puede utilizar una nueva cláusula llamada **WHERE CURRENT OF** seguida del nombre de un cursor, que hace que se modifique sólo el registro actual del cursor.
Ejemplo:

```
FOR r_emp IN c_emp LOOP
        IF r_emp.salario<1500 THEN
        UPDATE empleados SET salario = salario *1.30
        WHERE CURRENT OF c_emp;
END LOOP;
```