

BD

Bases de Datos

UD 10

Cursores

ÍNDICE

1. Introducción
2. Gestión de cursores (sintaxis específica SQL Server)
3. Gestión de cursores (mediante el uso de bucles)
4. Ejemplos de cursores

1. Introducción

Un cursor es una variable que nos permite recorrer con un conjunto de resultados obtenidos a través de una sentencia **SELECT** fila por fila.

Se utilizarán los cursores ANSI cuando sea necesario tratar las filas de manera individual en un conjunto o cuando SQL no pueda actuar únicamente sobre las filas afectadas. Los cursores API serán utilizados por las aplicaciones cliente para tratar volúmenes importantes o para gestionar varios conjuntos de resultados.

Cuando trabajemos con cursores, debemos seguir los siguientes pasos:

- Declarar el cursor, utilizando **DECLARE**
- Abrir el cursor, utilizando **OPEN**
- Leer los datos del cursor, utilizando **FETCH ... INTO**
- Cerrar el cursor, utilizando **CLOSE**
- Liberar el cursor, utilizando **DEALLOCATE**

La sintaxis general para trabajar con un cursor es la siguiente:

```
-- DECLARACIÓN DEL CURSOR
DECLARE NOMBRE_CURSOR CURSOR FOR
SENTENCIA SELECT

-- APERTURA DEL CURSOR
OPEN NOMBRE_CURSOR

-- LECTURA DE LA PRIMERA FILA DEL CURSOR
FETCH NEXT FROM NOMBRE_CURSOR INTO LISTA VARIABLES SEPARADAS POR COMAS

WHILE (@@FETCH_STATUS = 0)
BEGIN
    -- ACCIONES A REALIZAR PARA CADA REGISTRO
    ...
    ...

    -- LECTURA DE LA SIGUIENTE FILA DEL CURSOR
    FETCH NEXT FROM NOMBRE_CURSOR INTO LISTA VARIABLES SEPARADAS POR COMAS
END -- FIN DEL BUCLE WHILE

-- CIERRA EL CURSOR
CLOSE NOMBRE_CURSOR

-- LIBERA LOS RECURSOS DEL CURSOR (MEMORIA)
DEALLOCATE NOMBRE_CURSOR
```

A continuación, estudiaremos cada una de las partes que conforman un cursor.

2. Gestión de cursores (sintaxis específica SQL Server)

Declaración del cursor

Esta instrucción permite la declaración y la descripción del cursor.

Sintaxis:

```
DECLARE Cur_TABLA  
CURSOR [READ_ONLY] FOR  
    SELECT ...  
    FROM TABLA
```

Es una buena práctica anteponer la palabra “**Cur_**” y a continuación indicar la tabla que se va a recorrer en el nombre del cursor.

Apertura del cursor

Esta instrucción prepara la sentencia SELECT para ser ejecutada.

Sintaxis:

```
OPEN Cur_TABLA
```

Volcado en memoria de los registros que devuelve la SELECT

Esta instrucción ejecuta la sentencia SELECT y la vuelca en memoria (variables).

En caso en el que la SELECT del cursor devuelva solo un campo/columna:

Sintaxis:

```
FETCH NEXT FROM Cur_TABLA INTO @campo
```

La palabra **INTO** vuelca el registro en la variable @campo, que deberá ser del mismo tipo que el registro de la tabla.

En caso en el que la SELECT del cursor devuelva MÁS de un campo/columna:

Sintaxis:

```
FETCH NEXT FROM Cur_TABLA INTO @campo1, @campo2, @campo3, ...
```

Deberemos definir una variable para cada columna que devuelva el cursor y éstas deberán recorrerse en el mismo orden en el que aparezcan en la tabla. Igual que para el caso anterior, las variables deberán ser del mismo tipo que la columna de la tabla.

Por lo tanto, con cada sentencia FETCH se leerá un registro o fila de los que devuelva la consulta SELECT del cursor, pero ¿cómo podremos saber si hemos llegado a la última fila?

Disponemos de una variable de sistema en SQL Server llamada **@@FETCH_STATUS** para comprobar se ha realizado correctamente la lectura del registro.

Valor 0: Lectura correcta

Valor -1: Lectura incorrecta o la fila está más allá del conjunto de resultados

Valor -2: Falta la fila recuperada

CIERRE

Cuando se realiza el cierre del cursor, se cerrará el cursor para que no se puedan realizar más consultas sobre él.

Sintaxis:

```
CLOSE Cur_TABLA
```

DESALOJO

Cuando se realiza el desalojo del cursor, se liberará toda la memoria reservada al realizar el FETCH.

Sintaxis:

```
DEALLOCATE Cur_TABLA
```

3. Gestión de cursores (mediante el uso de bucles)

Alternativamente a la sintaxis de SQL Server para realizar la gestión de bucles, podemos utilizar la estructura de un bucle WHILE en el que vayamos controlando que se trata un elemento cada vez. En ciertas situaciones conviene evitar el uso de cursores mediante la sintaxis de FETCH puesto que genera utiliza muchísima memoria en el caso de que nuestra SELECT devuelva muchos registros.

Situación 1: Tabla con PK INT o BIGINT de tipo IDENTITY (autoincremento)

Este es el cursor más sencillo de emular mediante bucles. La estrategia consiste en empezar el bucle desde el primer registro de la tabla obteniéndolo mediante un MIN y acabarlo con el último registro utilizando un MAX (no confundir con COUNT que no contempla cuando existen huecos entre registros a consecuencia del borrado de alguno de ellos).

```
USE JARDINERIA

DECLARE @contador INT, @maxRegistro INT, @nombreCliente VARCHAR(100)

-- Obtenemos el valor mínimo que utilizaremos de contador
SELECT @contador = MIN(codCliente)
FROM CLIENTES;

-- Obtenemos el valor máximo al que llegará el bucle
SELECT @maxRegistro = MAX(codCliente)
FROM CLIENTES;

WHILE @contador <= @maxRegistro
BEGIN
    SELECT @nombreCliente = nombre_cliente
    FROM CLIENTES
    WHERE codCliente = @contador

    -- Antes de mostrar el registro comprobamos si tiene valor
    IF @nombreCliente IS NOT NULL
        PRINT CONCAT('ID: ', @contador, ' Nombre: ', @nombreCliente)

    -- Inicializamos la variable por si el registro siguiente no existe
    -- que no tome el nombre de la iteración anterior
    SET @nombreCliente = NULL

    -- Incrementamos el siguiente id de la tabla CLIENTES
    SET @contador += 1
END
```

Resultado:

```
ID: 1 Nombre: GoldFish Garden
ID: 3 Nombre: Gardening Associates
ID: 4 Nombre: Gerudo Valley
ID: 5 Nombre: Tendo Garden
ID: 6 Nombre: Lasas S.A.
ID: 7 Nombre: Beragua
ID: 8 Nombre: Club Golf Puerta del hierro
ID: 9 Nombre: Naturagua
ID: 10 Nombre: DaraDistribuciones
ID: 11 Nombre: Madrileña de riegos
ID: 12 Nombre: Lasas S.A.
```

...

Situación 2: Tabla con PK de tipo CHAR

Este cursor es un poco más complejo. La estrategia que debemos seguir es la de ordenar los registros ascendentemente o descendentemente e ir tratándolos uno a uno.

```
DECLARE @contador INT = 1, @maxRegistros INT
DECLARE @codOfic CHAR(10), @ciudadOficina VARCHAR(100)

-- Obtenemos el número de registros que tiene la tabla
SELECT @maxRegistros = COUNT(1)
FROM OFICINAS;

WHILE @contador <= @maxRegistros
BEGIN
    -- Si estamos en la primera iteración
    IF @contador > 1
    BEGIN
        -- Obtenemos la siguiente oficina a la que se está tratando
        SELECT TOP(1) @codOfic = codOficina,
                      @ciudadOficina = ciudad
        FROM OFICINAS
        WHERE codOficina > @codOfic
        ORDER BY codOficina ASC;
    END

    ELSE
    BEGIN
        -- Obtenemos la primera oficina a tratar
        SELECT TOP(1) @codOfic = codOficina,
                      @ciudadOficina = ciudad
        FROM OFICINAS
        ORDER BY codOficina ASC;
    END

    PRINT CONCAT('CodOficina: ', @codOfic, ', Ciudad: ', @ciudadOficina)

    -- Incrementamos
    SET @contador += 1
END
```

Resultado:

CodOficina: BCN-ES	, Ciudad: Barcelona
CodOficina: BOS-USA	, Ciudad: Boston
CodOficina: LON-UK	, Ciudad: Londres
CodOficina: MAD-ES	, Ciudad: Madrid
CodOficina: PAR-FR	, Ciudad: Paris
CodOficina: SFC-USA	, Ciudad: San Francisco
CodOficina: SYD-AU	, Ciudad: Sydney
CodOficina: TAL-ES	, Ciudad: Talavera de la Reina
CodOficina: TOK-JP	, Ciudad: Tokyo

4. Ejemplos de cursores

Ejemplo 1. Cursor que use la BD TIENDA y recorra la tabla FABRICANTE, imprimiendo por pantalla el nombre del fabricante.

```
USE TIENDA;

-- Declaración del cursor
DECLARE Cur_Fabricante CURSOR FOR
SELECT nombre
  FROM FABRICANTE;

-- Declaración de una variable del tipo de dato que devolverá el cursor
DECLARE @nomfabricante AS VARCHAR(100);

-- Apertura del cursor
OPEN Cur_Fabricante;

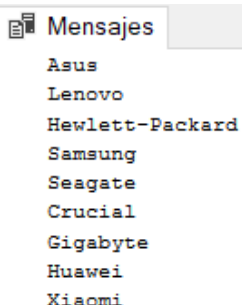
-- Volcado del registro en la variable
FETCH NEXT FROM Cur_Fabricante INTO @nomfabricante;

-- Entramos si la lectura ha sido correcta
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Imprimimos el fabricante actual
    PRINT @nomfabricante

    -- Pasamos al siguiente registro del cursor
    FETCH NEXT FROM Cur_Fabricante INTO @nomfabricante;
END

-- Cerramos y liberamos la memoria del cursor
CLOSE Cur_Fabricante;
DEALLOCATE Cur_Fabricante;
```

Salida:



Mensajes

Asus
Lenovo
Hewlett-Packard
Samsung
Seagate
Crucial
Gigabyte
Huawei
Xiaomi

Ejemplo 2. Cursor que use la BD TIENDA y recorra la tabla FABRICANTE, imprimiendo por pantalla el código del fabricante y su nombre de la siguiente manera:

Salida: Nombrefabricante - Código

```
USE TIENDA;

-- Declaración del cursor
DECLARE Cur_Fabricante CURSOR FOR
SELECT codigo, nombre
FROM FABRICANTE;

-- Declaración de una variable del tipo de dato que devolverá el cursor
DECLARE @codigo INT, @nomfabricante AS VARCHAR(100);

-- Apertura del cursor
OPEN Cur_Fabricante;

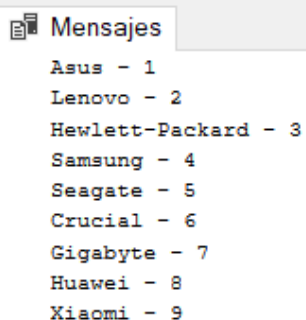
-- Volcado del registro en la variable
FETCH NEXT FROM Cur_Fabricante INTO @codigo, @nomfabricante;

-- Entramos si la lectura ha sido correcta
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Imprimimos el fabricante actual
    PRINT CONCAT(@nomfabricante, ' - ', @codigo);

    -- Pasamos al siguiente registro del cursor
    FETCH NEXT FROM Cur_Fabricante INTO @codigo, @nomfabricante;
END

-- Cerramos y liberamos la memoria del cursor
CLOSE Cur_Fabricante;
DEALLOCATE Cur_Fabricante;
```

Salida:



Mensajes

```
Asus - 1
Lenovo - 2
Hewlett-Packard - 3
Samsung - 4
Seagate - 5
Crucial - 6
Gigabyte - 7
Huawei - 8
Xiaomi - 9
```

NOTA: Es muy importante el orden en el que se recogen las columnas en las variables, sino estos no coinciden, el script finalizará con errores.

[ANIDACIÓN DE CURSORES]

Ejemplo 3. Cursor que use la BD JARDINERIA. Debe recorrer la tabla CLIENTES y para cada cliente imprimir el total gastado abriendo un cursor secundario en la tabla PAGOS. Se obtendrá el acumulado sumando el total devuelto por el segundo cursor en una variable local.

```
USE JARDINERIA;

-- Declaración de una variable del tipo de dato que devolverá el cursor principal
DECLARE @codCliente INT;

-- Declaración del cursor principal
DECLARE Cur_Clientes CURSOR FOR
SELECT codCliente
FROM CLIENTES;

-- Apertura del cursor principal
OPEN Cur_Clientes;

-- Volcado del registro en la variable
FETCH NEXT FROM Cur_Clientes INTO @codCliente;

-- Entramos si la lectura ha sido correcta
WHILE @@FETCH_STATUS = 0
BEGIN
    -----
    -- Gestionamos el cursor secundario
    -----
    DECLARE Cur_Pagos CURSOR FOR
    SELECT total
    FROM PAGOS
    WHERE codCliente = @codCliente;

    -- Declaración variable del tipo de dato que devolverá el cursor secundario
    DECLARE @total NUMERIC(15,2), @acumulado NUMERIC(15,2)
    SET @acumulado = 0;

    OPEN Cur_PAGOS;
    FETCH NEXT FROM Cur_PAGOS INTO @total;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @acumulado = @acumulado + @total;
        FETCH NEXT FROM Cur_PAGOS INTO @total;
    END
    CLOSE Cur_PAGOS;
    DEALLOCATE Cur_PAGOS;

    -- Imprimimos los datos del cliente y el acumulado
    PRINT CONCAT('CodCliente: ', @codCliente, ' ha pagado: ', @acumulado);

    -- Pasamos al siguiente registro del cursor principal
    FETCH NEXT FROM Cur_Clientes INTO @codCliente;
END

-- Cerramos y liberamos la memoria del cursor
CLOSE Cur_Clientes;
DEALLOCATE Cur_Clientes;
```

Salida:

Mensajes

```
CodCliente: 1 ha pagado: 4000.00
CodCliente: 3 ha pagado: 10926.00
CodCliente: 4 ha pagado: 81849.00
CodCliente: 5 ha pagado: 23794.00
CodCliente: 6 ha pagado: 0.00
CodCliente: 7 ha pagado: 2390.00
CodCliente: 8 ha pagado: 0.00
CodCliente: 9 ha pagado: 929.00
CodCliente: 10 ha pagado: 0.00
CodCliente: 11 ha pagado: 0.00
CodCliente: 12 ha pagado: 0.00
CodCliente: 13 ha pagado: 2246.00
CodCliente: 14 ha pagado: 4160.00
CodCliente: 15 ha pagado: 12081.00
CodCliente: 16 ha pagado: 4399.00
CodCliente: 17 ha pagado: 0.00
CodCliente: 18 ha pagado: 0.00
CodCliente: 19 ha pagado: 232.00
CodCliente: 20 ha pagado: 0.00
CodCliente: 21 ha pagado: 0.00
CodCliente: 22 ha pagado: 0.00
CodCliente: 23 ha pagado: 272.00
CodCliente: 24 ha pagado: 0.00
CodCliente: 25 ha pagado: 0.00
CodCliente: 26 ha pagado: 18846.00
CodCliente: 27 ha pagado: 10972.00
CodCliente: 28 ha pagado: 8489.00
CodCliente: 29 ha pagado: 0.00
CodCliente: 30 ha pagado: 7863.00
CodCliente: 31 ha pagado: 0.00
CodCliente: 32 ha pagado: 0.00
CodCliente: 33 ha pagado: 0.00
CodCliente: 35 ha pagado: 3321.00
CodCliente: 36 ha pagado: 0.00
CodCliente: 37 ha pagado: 0.00
CodCliente: 38 ha pagado: 1171.00
```