IES CHAN DO MONTE
C.S. de Desarrollo de Aplicaciones Multiplataforma
*Módulo Base de datos*

# UNIDAD 9 parte 1: Cursores

## Índice

# 1. Cómo funcionan los cursores Transact-SQL.

- ✓ Las **operaciones de una base de datos relacional** actúan en un **conjunto completo de filas (Conjunto de resultados)**.
  - El **conjunto de filas que devuelve una instrucción SELECT está compuesto por todas las filas que satisfacen las condiciones de la cláusula WHERE** de la instrucción.
  - Las aplicaciones, especialmente las aplicaciones interactivas en línea, no siempre trabajan de forma eficaz con el conjunto de resultados completo si lo toman como una unidad. Estas aplicaciones necesitan un **mecanismo que trabaje con una fila o un pequeño bloque de filas cada vez**. Los cursores son una extensión de los conjuntos de resultados que proporcionan dicho mecanismo.
- ✓ Los **cursores** ofrecen la habilidad de **desplazarse hacia delante y hacia atrás a través de un conjunto de resultados, fila a fila, para poder ver o procesar datos**.
  - Podemos pensar en un cursor como un puntero señalando una fila específica en un conjunto de resultados.

Los cursores, extienden el procesado de conjunto de resultados mediante que:

- Permiten **situarse en filas específicas** del conjunto de resultados.
- **Recuperan una fila** o un bloque de filas de la **posición actual** en el conjunto de resultados.
- **Aceptan modificaciones de los datos** de las filas en la **posición actual** del conjunto de resultados.
- **Aceptan diferentes grados de visibilidad** para los **cambios** que **realizan otros usuarios** en la información de la base de datos que se presenta en el conjunto de resultados.
- **Proporcionan instrucciones Transact-SQL** en secuencias de comandos, procedimientos almacenados y acceso de desencadenadores a los datos de un conjunto de resultados

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

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

```
-- Declaración del cursor
DECLARE <nombre_cursor> CURSOR
FOR
        <sentencia_sql>
 -- apertura del cursor
OPEN <nombre_cursor>
 -- Lectura de la primera fila del cursor
FETCH <nombre_cursor> INTO <lista_variables>
  WHILE (@@FETCH_STATUS = 0)
  BEGIN
    ..............  //lógica con la fila dónde está el cursor
   -- Lectura de la siguiente fila de un cursor
       FETCH <nombre_cursor> INTO <lista_variables>
         ...
  END -- Fin del bucle WHILE
 -- Cierra el cursor
CLOSE <nombre_cursor>
-- Libera los recursos del cursor
DEALLOCATE <nombre_cursor>
```

Los siguientes pasos nos permiten definir y utilizar cursores:

1. **Declarar el cursor**, utilizando **DECLARE**. Esto **incluye la instrucción SELECT** que genera un conjunto de resultados y define las características del cursor, como si las filas en el cursor pueden ser actualizadas.
2. **Abrir el cursor**, ejecutando la instrucción **OPEN** para **generar un conjunto de resultados y llenar el cursor**.
3. **Recuperar filas** del conjunto de resultados del cursor, utilizando **FETCH ... INTO**. La operación para recuperar una fila o un bloque de filas de un cursor se denomina recuperación. La realización de una serie de obtenciones se denomina desplazamiento.
4. **Utilizar los valores o ejecutar operaciones en la fila en la posición actual del cursor.**
5. **Cerrar el cursor, utilizando CLOSE**
6. **Liberar los recursos** utilizados por el cursor, utilizando **DEALLOCATE.**

**Ejemplo:**

- Este ejemplo **crea un cursor** basado en una **consulta** que devuelve los nombres y el departamento de los empleados mujeres de la base de datos Compañia.
- Después de que se ha abierto el cursor, la secuencia de comandos se desplaza a través del conjunto de resultados del cursor.
- Cuando el proceso del cursor se ha completado, el cursor se cierra y se liberan los recursos utilizados.

```
USE Compañia
SET NOCOUNT ON
---- Declaración del cursor
DECLARE EmpMujeres_cursor CURSOR FOR
-- Realizamos la consulta que queremos guardar en la variable cursor
        SELECT nombre,apel1,apel2,nombreDep
        FROM Empleados E INNER JOIN Departamentos D    ON E.Numdep=D.Numdep
        WHERE Sexo= 'M'
        ORDER BY nombre
--Declaración de variables para almacenar las columnas de la fila a las que apunta el cursor
DECLARE @nombre varchar(25)
DECLARE @Apellido1 varchar(30)
DECLARE @Apellido2 varchar(30)
DECLARE @NombreDepart varchar(60)

PRINT '    -------LISTADO DE EMPLEADOS MUJERES--------'
-- Apertura del cursor y llena el conjunto de resultados
 OPEN EmpMujeres_cursor
-- Lectura de la primera fila de un cursor y se gurda en las variables
FETCH NEXT FROM EmpMujeres_cursor INTO @nombre , @Apellido1, @Apellido2,    @NombreDepart

--Mientras el cursor tenga datos, visualizamos las variables y leemos la siguiente fila
WHILE @@FETCH_STATUS =0
BEGIN
   PRINT 'Estado de FETCH: ' + CAST(@@FETCH_STATUS AS varchar(2))
   PRINT 'NOMBRE: ' + @nombre  + '  APELLIDOS:'+@Apellido1+ ' '+
                @Apellido2
   PRINT 'DEPARTAMENTO DONDE TRABAJA:'+ @NombreDepart
   PRINT ' '
-- Lectura de la siguiente fila de un cursor
    FETCH NEXT FROM EmpMujeres_cursor INTO @nombre , @Apellido1, @Apellido2, @NombreDepart
END
PRINT 'Estado de FETCH: ..........' + CAST(@@FETCH_STATUS AS varchar(2))
CLOSE EmpMujeres_cursor  --Cerrar el cursor
DEALLOCATE EmpMujeres_cursor  --Liberar los recursos utilizados
```

En resumen:

- **DECLARE CURSOR** define **los atributos de un cursor** de servidor Transact-SQL, como su **comportamiento de desplazamiento** y la **consulta** utilizada para generar el conjunto de resultados en que opera el cursor.
- La instrucción **OPEN llena el conjunto de resultados**
- La instrucción **FETCH devuelve una fila** del conjunto de resultados.
- La instrucción **CLOSE libera el conjunto de resultados** actual asociado con el cursor.
- La instrucción **DEALLOCATE libera los recursos** que utiliza el cursor.

# 2. Tipos de cursores Transact-SQL

Los **diferentes tipos de cursores se diferencian según puedan:**

✓ **Detectar cambios en el conjunto de resultados**: Si se tiene un cursor abierto y los **datos en la tabla** o tablas en los que está basado el cursor, son **modificados por otras conexiones**, los diferentes tipos de **cursores pueden reflejar o no esos cambios**.

Pueden reflejar cambios en:

- En los **valores de las columnas**
- En el **orden de las filas**
- En la afiliación **(número de filas afectadas por el cursor)**

✓ En los **recursos que utilizan**, como memoria y espacio en la base de datos tempdb

SQL Server **soporta cuatro tipos de cursores**:

1. **sólo-avance**
2. **estático**
3. **dinámico y**
4. **controlado por un conjunto de claves**

- **Sólo-avance**
  - ✓ Sólo se pueden **recuperar filas en serie** desde la primera a la última fila del cursor.
  - ✓ **El efecto de todas las instrucciones INSERT, UPDATE y DELETE** que **cualquier otra conexión** realiza **antes de que se recupere la fila son visibles cuando se recupera la fila**.
- **Estático**
  - ✓ **Fija el conjunto de resultados cuando se abre el cursor** y es siempre de **sólo lectura**.

    De este modo **no es posible actualizar los datos de las tablas definidas de un cursor estático a través del cursor**.
  - ✓ El **conjunto de resultados se almacena en tempdb cuando se abre el cursor estático**.
  - ✓ Los **cambios realizados por otras conexiones después de que el cursor ha sido abierto nunca son reflejados por el cursor.**
- **Dinámico**
  - ✓ Los cursores dinámicos son los opuestos de los cursores estáticos.
  - ✓ Un cursor dinámico **refleja todos los cambios que son realizados a los valores de datos, orden y afiliación de filas** en el conjunto de resultados cuando se desplaza a través de un conjunto de resultados de cursor.
  - ✓ Los **efectos de todas las instrucciones UPDATE, INSERT y DELETE** que realiza cualquier **usuario son visibles mientras** las filas son recuperadas por el cursor.
  - ✓ Se **pueden recuperar filas aleatoriamente desde cualquier parte del cursor**.

- **Controlados por un conjunto de claves:**
  - ✓ La **afiliación (número de filas afectadas por el cursor) y orden de filas** en un cursor controlado por un conjunto de claves **son fijados cuando se abre el cursor**.
  - ✓ Son controlados por un conjunto de identificadores únicos (claves), denominado conjunto de claves.
    - Las claves se generan desde un conjunto de columnas que identifican únicamente a un conjunto de filas en el conjunto de resultados.
    - **El conjunto de claves es el conjunto de valores clave de todas las filas que se calificaron para la instrucción SELECT cuando el cursor fue abierto y la afiliación y el orden de las filas nunca es actualizado.**
  - ✓ Las **actualizaciones que el usuario realiza a través del cursor son visibles, así como los cambios que otras conexiones realizan a valores de datos en columnas sin conjuntos de claves**.
  - ✓ El **conjunto de claves de un cursor controlado por un conjunto de claves** se almacena en **tempdb** cuando quiera que se haya abierto el cursor.

Los cursores con características fijas de afiliación, orden y valores (es decir, los cursores estáticos), actúan más rápido que los cursores con características dinámicas, aunque pueden ser un poco más lentos en el momento de su apertura.
Los cursores estáticos y controlados por un conjunto de claves incrementan la utilización de la base de datos tempdb:

- Los cursores de estáticos generan el cursor completo en tempdb.
- Los cursores controlados por un conjunto de claves generan el conjunto de claves en tempdb. Por esta razón, si un gran número de usuarios abren cada uno un cursor estático o controlado por un conjunto de claves, tempdb puede quedarse sin espacio

# 3. Sintaxis de los cursores Transact-SQL

Se utilizan cinco instrucciones cuando se trabaja con cursores Transact-SQL. La siguiente tabla resume estas cinco instrucciones.

| Instrucción | Descripción |
|---|---|
| **DECLARE CURSOR** | Define la estructura del cursor y asigna recursos. |
| **OPEN** | Llena un cursor declarado con un conjunto de resultados. |
| **FETCH** | Se desplaza dentro de un conjunto de resultados de cursor. |
| **CLOSE** | Libera el conjunto de resultados actual y libera cualquier bloqueo de cursor mantenido en filas en que está posicionado el cursor. |
| **DEALLOCATE** | Elimina la definición del cursor y desasigna el cursor. |

## 3.1 Funciones

Hay muchas funciones que devuelven información acerca del estado de un cursor. La siguiente tabla describe tres de las funciones más utilizadas:

| Función | Descripción |
|---|---|
| **@@FETCH_STATUS** | Devuelve el estado de la última instrucción de cursor FETCH que fue emitida en la conexión |

| | |
|---|---|
| | **0**: La instrucción FETCH se ejecutó correctamente.<br>**-1**: La instrucción FETCH no se ejecutó correctamente o la fila estaba más allá del conjunto de resultados.<br>-**2**: Falta la fila recuperada |
| **@@CURSOR_ROWS** | **Devuelve el número de filas** del último cursor que esté abierto en la conexión. |

| Valor devuelto | Descripción |
|---|---|
| *-m* | El cursor se llena de forma asincrónica. El valor devuelto (*-m*) es el número de filas que contiene actualmente el conjunto de claves. |
| **-1** | **El cursor es dinámico**. Como los cursores dinámicos reflejan todos los cambios, el número de filas correspondientes al cursor cambia constantemente. Nunca se puede afirmar que se han recuperado todas las filas que correspondan. |
| **0** | **No se han abierto cursores**, no hay filas calificadas para el último cursor abierto, o éste se ha cerrado o su asignación se ha cancelado. |
| ***n*** | El cursor está completamente lleno. **El valor devuelto (*n*) es el número total de filas del cursor**. |

| | |
|---|---|
| **CURSOR_STATUS** | Después de llamar a un procedimiento almacenado que devuelve un cursor como parámetro de salida, la función le permite **determinar el estado del cursor que ha sido devuelto**. |

## 3.2 DECLARE CURSOR

✓ DECLARE CURSOR define:

- **Los atributos de un cursor**,
- Su **comportamiento de desplazamiento**
- **Y la consulta utilizada** para generar el conjunto de resultados en que opera el cursor.

**Sintaxis simplificada de la extensión de Transact-SQL**

```
DECLARE nombre_cursor CURSOR [ LOCAL | GLOBAL ]
   [ FORWARD_ONLY | SCROLL ]
   [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
   [ READ_ONLY ]
   FOR INSTRUCCIÓN_SELECT
```

### Argumentos

- **nombreCursor**

  Se trata del **nombre del cursor** SQL que se va a definir.

- **instrucción_SELECT**

  Es **una instrucción SELECT** estándar que define el conjunto de resultados del cursor.

  La palabra clave **INTO no está permitida** en la *instrucción SELECT* de la declaración de cursor.

  Si se utiliza **DISTINCT, UNION, GROUP BY o HAVING** o si se incluye **una expresión de agregado el cursor se creará como STATIC**.

- **[LOCAL | GLOBAL]**
  - **LOCAL**

    Limita el uso del cursor al proceso por lotes, procedimiento almacenado o desencadenador en el que fue creado. El **nombre del cursor sólo es válido dentro de este alcance**.

- **GLOBAL**

  Especifica que el alcance del cursor es **global para la conexión**.

  Puede hacerse referencia al nombre del cursor en cualquier procedimiento almacenado o proceso por lotes que se ejecute **durante la conexión.**

  Sólo se **cancela implícitamente la asignación del cursor cuando se realiza la desconexión**.

  **Si no se especifica GLOBAL o LOCAL**, el valor predeterminado se controla a opción de base de datos **cursor local como predeterminado( CURSOR_DEFAULT).** Suele ser **Global (false o 0)**

  El estado de esta opción se puede determinar mediante la columna **is_local_cursor_default** de la vista de catálogo **sys.databases** ( 0-false-Global 1-True-Local)

```
SELECT name,is_local_cursor_default FROM sys,databases
```

  Para **cambiar el ámbito predeterminado del cursor**:

```
ALTER DATABASE BASE_DE_DATOS
SET CURSOR_DEFAULT {LOCAL|GLOBAL }
```

- **[FORWARD_ONLY | SCROLL ]**

  - **FORWARD_ONLY**

    Especifica que el cursor **sólo se puede desplazar desde la primera a la última fila (**FETCH NEXT es la única opción de recuperación aceptada).

    ✓ Si **solo se especifica FORWARD_ONLY** (sin las palabras clave STATIC, KEYSET o DYNAMIC), el cursor funciona como un **cursor DYNAMIC**.

    ✓ **Si no se especifica FORWARD_ONLY ni SCROLL: FORWARD_ONLY es la opción predeterminada.**

    ✓ Excepto **si sólo se especifican las palabras claves STATIC, KEYSET o DYNAMIC** ( sin especificar **FORWARD_ONLY ni SCROLL)** , los cursores son de tipo **SCROLL**.

  - **SCROLL**

    **Especifica que están disponibles todas las opciones de recuperación** (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE).

    ✓ **Si no se especifica SCROLL,** única opción de recuperación que se admite es NEXT.

    ✓ No es posible especificar **SCROLL si** se incluye también **FAST_FORWARD** (**FORWARD_ONLY, READ_ONLY)**.

- **[STATIC | KEYSET | DYNAMIC | FAST_FORWARD].**

  - **STATIC**

    Define un cursor que **hace una copia temporal de los datos que utiliza.**

    Todas las **peticiones al cursor se responden** desde esta tabla temporal de **tempdb**; por tanto:

    - Las **modificaciones realizadas en las tablas base no se reflejarán en los datos devueltos** por las recuperaciones realizadas en el cursor
    - y además el **cursor no admite modificaciones**.

  - **KEYSET**

    Especifica que **la pertenencia (afiliación) y el orden de las filas del cursor se fijan al abrir éste**.

    El conjunto de claves que identifica de forma única las filas está integrado en una tabla de **tempdb** conocida como **keyset**.

    ✓ Los **cambios en valores que no sean claves de las tablas base**, **son visibles** al desplazarse por el cursor.

    ✓ Las **inserciones realizadas por otros usuarios no son visibles** (**no es posible hacer inserciones a través de un cursor Transact-SQL**).

- ✓ Si **se elimina una fila**, el **intento de recuperarla obtendrá un valor de -2** en @@FETCH_STATUS (es decir falta la fila recuperada).
- ✓ Las **actualizaciones** de los **valores de claves desde fuera del cursor** se asemejan a la eliminación de la fila antigua seguida de la inserción de la nueva**. La fila con los nuevos valores no es visible** y los intentos de recuperar la fila de los valores antiguos devuelven el valor -2 en @@FETCH_STATUS.

- **DYNAMIC**

  Define un cursor que, al desplazarse por él, refleja en su conjunto de **resultados todos los cambios realizados en los datos de las filas**.

  Los valores de los datos, el orden y la pertenencia de las filas pueden cambiar en cada recuperación.

  **La opción de recuperación ABSOLUTE no se puede utilizar en los cursores dinámicos.**

- **FAST_FORWARD**

  Especifica un cursor **FORWARD_ONLY, READ_ONLY** con las optimizaciones de rendimiento habilitadas.

  No se puede especificar FAST_FORWARD si se especifica también SCROLL o FOR_UPDATE.

- **READ_ONLY**

  **Impide que se realicen actualizaciones a través de este cursor**.

## 3.3 OPEN

La instrucción **OPEN abre y llena el cursor** ejecutando la instrucción SELECT.

- Si se declara el cursor con la opción: **STATIC y KEYSET**, se crea una ***tabla temporal*** para mantener las filas del cursor o el conjunto de claves respectivamente.

**Sintaxis**

```
OPEN @nombre_variable_cursor
```

## 3.4 FETCH

La instrucción **FETCH recupera una fila específica** del cursor.

**Sintaxis**

```
FETCH
[ [NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n
FROM   @nombre_variable_cursor
INTO @nombre_variable [,...n]
```

**Argumentos**

- **NEXT**: Devuelve **la fila inmediatamente posterior a la fila actual**, y **aumenta la fila actual a la fila devuelta**. Si es la primera operación de recuperación en un cursor, se devuelve **la primera fila. Es la opción predeterminada.**
- **PRIOR:** Devuelve la **fila inmediatamente anterior a la fila actual**, y la **convierte en la fila actual**. Si **la primera operación de recuperación** en un cursor, no se devuelve ninguna fila y el cursor queda posicionado delante de la primera fila.
- **FIRST :** Devuelve **la primera fila del cursor y la convierte en la fila actual**.

- **LAST:** Devuelve **la última fila del cursor y la convierte en la fila actual**.
- **ABSOLUTE n** ( n debe ser una constante entera )

  Si **n es positivo**, se devuelve **la fila n desde el principio del cursor** y se convierte en la nueva **fila actual**.
  Si **n es negativo**, se devuelve la **fila n anterior al final del cursor** y se convierte en la nueva **fila actual**.
  Si **n o es 0**, no se **devuelve ninguna fila**.
- **RELATIVE n** ( n debe ser una constante entera )

  Si **n es positivo**, se devuelve **la fila n posterior a la fila actual** y se convierte en la nueva **fila actual**.
  Si **n es negativo**, se devuelve **la fila n anterior a la fila actual** y se convierte en la nueva **fila actual**.
  Si **n es 0, se devuelve la fila actual.**
- **@nombre_variable_cursor**:

  Es el nombre de una variable de cursor que hace referencia al cursor abierto desde el que se va efectuar la recuperación.
- **INTO @nombre_variabl [ ,...n]**
  - ✓ Permite **colocar en variables los datos de las columnas de una recuperación**.
  - ✓ Todas las **variables de la lista, de izquierda a derecha, están asociadas a las columnas correspondientes del conjunto de resultados** del cursor.
  - ✓ **El tipo de datos de cada variable tiene que coincidir o ser compatible con el tipo de datos de la columna** correspondiente del conjunto de resultados.
  - ✓ El **número de variables debe coincidir con el número de columnas** de la lista de selección del cursor.

Hay que considerar los siguientes hechos cuando utilice una **instrucción FETCH**:

- El tipo de opción FETCH que se soporta dentro de un cursor depende del tipo de cursor declarado.
- La función **@@FETCH_STATUS** se actualiza con cada ejecución de la instrucción FETCH.
- Se utiliza **@@FETCH_STATUS** antes de intentar operar con los datos para determinar si una recuperación ha tenido éxito o no.

| Valor devuelto | Descripción |
|---|---|
| **0** | La instrucción FETCH se ejecutó correctamente. |
| **-1** | La instrucción FETCH no se ejecutó correctamente o la fila estaba más allá del conjunto de resultados. |
| **-2** | Falta la fila capturada. |

Cuando definimos un cursor que, al desplazarse por él, refleje en su conjunto de **resultados todos los cambios realizados en los datos de las filas, s**i se elimina una fila, el intento de **recuperarla obtendrá un valor de -2 en @@FETCH_STATUS** (es decir falta la fila recuperada), por tanto, el siguiente código terminaría el procesamiento de las filas aunque hubiera más.

```
FETCH NEXT FROM .....INTO .....
  WHILE @@FETCH_STATUS =0
BEGIN
      ........
       ..............
    -- Lectura de la siguiente fila de un cursor
     FETCH NEXT FROM .....INTO .....
   --si se elimina la fila se obtendría un valor -2 en @@FETCH_STATUS y se saldría del bucle
END
```

Para seguir procesando las filas tendríamos que hacer:

```
WHILE (@@FETCH_STATUS <> -1)
  BEGIN
    IF (@@FETCH_STATUS = -2)
     BEGIN
                    FETCH NEXT FROM …….. INTO ……..
                  CONTINUE
     END
      . . .
   FETCH NEXT FROM …….. INTO ……..
 END
```

**Ejemplo de desplazamiento a través de un cursor**. Tenemos la siguiente tabla

| Nombre | Apellido_1 | Apellido_2 |
|---|---|---|
| Mariña | Bello | Arias |
| Valeriano | Boo | Boo |
| Celia | Bueno | Valiña |
| Rocio | López | Ferreiro |
| Anxos | Loures | Freire |
| Rosa | Mariño | Rivera |
| Breixo | Pereiro | Lamela |
| Javier | Quintero | Alvarez |

```
USE Empresa_Clase;
GO
-- Se declara el cursor.
DECLARE Empleado_cursor  CURSOR STATIC SCROLL FOR
SELECT Nombre,Apellido_1,Apellido_2 FROM copiaEmpleado
ORDER BY Apellido_1,Apellido_2,Nombre;
--Declaración de variable
DECLARE @nombre varchar(25), @Apellido1 varchar(30)
DECLARE @Apellido2 varchar(30)
--Se abre el cursor
OPEN Empleado_cursor;
-- Lectura de la primera fila del cursor
FETCH FIRST FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado FIRST: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
-- Lectura de la última fila del cursor
FETCH LAST FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado LAST: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
--Lectura de la fila inmediatamente anterior a la fila actual
FETCH PRIOR FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado PRIOR: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
-- Lectura de la segunda fila del cursor
FETCH ABSOLUTE 2 FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado ABSOLUTE: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
--Lectura de la fila que está tres filas despues de la fila actual
FETCH RELATIVE 3 FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado RELATIVE: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
```

```
--Lectura de la fila que está dos filas antes de la fila actual.
FETCH RELATIVE -2 FROM Empleado_cursor INTO @nombre,@Apellido1,@Apellido2;
PRINT 'resultado RELATIVE -: ' + @nombre+' '+ @apellido1+' '+ @Apellido2
--Se cierra el cursor y se libera sus recursos
CLOSE Empleado_cursor;
DEALLOCATE Empleado_cursor;
```

La salida es:

```
resultado FIRST: Mariña Bello Arias
resultado LAST: Javier Quintero Alvarez
resultado PRIOR: Breixo Pereiro Lamela
resultado ABSOLUTE: Valeriano Boo Boo
resultado RELATIVE: Anxos Loures Freire
resultado RELATIVE -: Celia Bueno Valiña
```

## 3.5 CLOSE

- ✓ Libera el conjunto de resultados actual.
- ✓ Libera cualquier bloqueo que esté mantenido en la fila en la que está posicionado el cursor **pero deja las estructuras de datos accesibles para su reapertura**.
- ✓ Las Modificaciones y recuperaciones no están permitidas hasta que el cursor esté reabierto.
- ✓ Los cursores cerrados pueden ser reabiertos.

**Sintaxis**

```
CLOSE nombre_variable_cursor
```

## 3.6 DEALLOCATE

- ✓ Elimina la asociación entre un cursor y la variable de cursor que haga referencia al cursor.
- ✓ **Se liberan las estructuras de datos utilizadas por el cursor.**

**Sintaxis**

```
DEALLOCATE @nombre_variable_cursor
```

## 3.7 Obtener información acerca de los cursores

| Procedimiento almacenado de sistema | Descripción |
|---|---|
| **sp_cursor_list** | Devuelve una lista de cursores abiertos actualmente por la conexión y atributos de los cursores. |
| **sp_describe_cursor** | Describe los atributos de un cursor, tanto si es sólo de avance como si es desplazable. |
| **sp_describe_cursor_columns** | Describe los atributos de las columnas del conjunto de resultados del cursor. |
| **sp_describe_cursor_tables** | Describe las tablas base a las que accede el cursor. |

# 4. Ejemplos

## A. Utilizar cursores simples y su sintaxis

El conjunto de resultados generado al abrir este cursor contiene todas las filas y todas las columnas de la tabla **autores** de la base de datos **biblioteca**. Este **cursor se puede actualizar,** y todas las actualizaciones y eliminaciones se representan en las recuperaciones realizadas contra el cursor. **FETCH NEXT** es la única recuperación disponible debido a que no se ha especificado la opción SCROLL.

```
USE Biblioteca
SET NOCOUNT ON
DECLARE revistas_cursor CURSOR FOR
        SELECT NumeroISBN, Titulo
        FROM Libros
        WHERE Formato = 'R'
        ORDER BY NumeroISBN
DECLARE @NumeroISBN char(13)
DECLARE @Titulo varchar(75)

PRINT '                    -------LISTADO DE REVISTAS--------'
OPEN revistas_cursor
FETCH NEXT FROM revistas_cursor INTO @NumeroISBN, @titulo
WHILE @@FETCH_STATUS <> -1
BEGIN
      IF (@@FETCH_STATUS = -2)
      BEGIN
                FETCH NEXT FROM revistas_cursor INTO @NumeroISBN, @titulo
        CONTINUE
      END
        PRINT 'Revista .......: ' + @NumeroISBN + '      Título: ' + @Titulo
        PRINT ' '
        FETCH NEXT FROM revistas_cursor INTO @NumeroISBN, @titulo
END
CLOSE revistas_cursor
DEALLOCATE revistas_cursor
```

## B. Utilizar cursores anidados para elaborar resultados de informes:

```
USE CompañiaConsultas
SET NOCOUNT ON
DECLARE EmpMujeres_cursor CURSOR FOR
        SELECT E.NSS,nombre,apel1,apel2,nombreDep
        FROM Empleados E INNER JOIN Departamentos D   ON E.Numdep=D.Numdep
        WHERE Sexo= 'M'
        ORDER BY nombre
DECLARE @numempleado char(8)
DECLARE @nombre varchar(25), @Apellido1 varchar(30)
DECLARE @Apellido2 varchar(30)
DECLARE @NombreDepart varchar(60)
PRINT '          -------LISTADO DE EMPLEADOS MUJERES--------'
OPEN EmpMujeres_cursor
FETCH NEXT FROM EmpMujeres_cursor INTO  @numempleado,@nombre ,
@Apellido1, @Apellido2,@NombreDepart
WHILE @@FETCH_STATUS =0
BEGIN
        PRINT 'NOMBRE: ' + @nombre  + '  APELLIDOS:'+@Apellido1+ ' '+
                @Apellido2
        PRINT 'DEPARTAMENTO DONDE TRABAJA:'+ @NombreDepart
        PRINT ' '
-- Declara un cursor basado en el @numempleado del otro cursor.
```

```
            PRINT '       << proyectos>> '
            DECLARE Proyectos_cursor CURSOR FOR
                        SELECT pr.nombreproy
                        FROM participacion p,proyectos pr
                        WHERE p.NSS=@numempleado AND p.Numproy = pr.Numproy
            DECLARE @nombreproy varchar(50)
            OPEN Proyectos_cursor
            FETCH NEXT FROM Proyectos_cursor  INTO @nombreproy
            IF @@FETCH_STATUS <> 0
            PRINT '       ---NINGUNO--- '
            WHILE @@FETCH_STATUS = 0
            BEGIN
                        PRINT @nombreproy
                        FETCH NEXT FROM Proyectos_cursor INTO @nombreproy
            END
            CLOSE Proyectos_cursor
            DEALLOCATE Proyectos_cursor
            PRINT ''
            FETCH NEXT FROM EmpMujeres_cursor INTO  @numempleado,
            @nombre,@Apellido1,@Apellido2,@NombreDepart
      END
      CLOSE EmpMujeres_cursor
      DEALLOCATE EmpMujeres_cursor
```

```
------LISTADO DE EMPLEADOS MUJERES--------
NOMBRE: Ana  APELLIDOS:Rodríguez SanTomé
DEPARTAMENTO DONDE TRABAJA:Marketing
      << proyectos>>
A
B
C
D
E

NOMBRE: Isabel  APELLIDOS:Tenorio Freire
DEPARTAMENTO DONDE TRABAJA:Informática
      << proyectos>>
A
F
G

NOMBRE: Lucia  APELLIDOS:Rodriguez Galán
DEPARTAMENTO DONDE TRABAJA:Servicios
      << proyectos>>
         ---NINGUNO---
     ..............
```

.

# 5. Modificar datos a través de los cursores

- ✓ Se puede modificar datos a través de cursores Transact-SQL locales o globales.
- ✓ Se debe declarar el **cursor como actualizable** (**opción FOR UPDATE**), y **no se debe declarar como de sólo lectura**.
- ✓ En un cursor actualizable puede utilizar las instrucciones **UPDATE o DELETE** con la cláusula **WHERE CURRENT OF nombre-cursor para modificar la fila actual**.

**Ejemplo:** Subir el salario en un 5% a las mujeres que ganen más de 23000 y pertenezcan al departamento de contabilidad o informática y un 10% a las mujeres que ganen mas de 24000 que pertenezcan al departamento técnico

```
USE Empresa_Clase
SET NOCOUNT ON
---- Declaración del cursor
DECLARE SubirSalarioMujeres_cursor CURSOR FOR
-- Realizamos la consulta que queremos guardar en la variable cursor
    SELECT nombre,apellido_1,apellido_2,Salario,nombre_Departamento
    FROM Empleado E INNER JOIN Departamento D
    ON E.Num_departamento_pertenece=D.Num_departamento
    WHERE Sexo= 'M'
    ORDER BY apellido_1,apellido_2,nombre
    FOR UPDATE
--Declaracion de variables para almacenear las columnas de la fila
--a las que apunta el cursor
DECLARE @nombre varchar(25)
DECLARE @Apellido1 varchar(30)
DECLARE @Apellido2 varchar(30)
DECLARE @Salario money
DECLARE @NombreDepart varchar(60)
- Apertura del cursor y llena el conjunto de resultados
 OPEN SubirSalarioMujeres_cursor
-- Lectura de la primera fila de un cursor
FETCH NEXT FROM SubirSalarioMujeres_cursor INTO @nombre ,  @Apellido1,
            @Apellido2,  @salario,@NombreDepart
WHILE @@FETCH_STATUS <> -1
BEGIN
     IF (@@FETCH_STATUS = -2)
     BEGIN
       FETCH NEXT FROM SubirSalarioMujeres_cursor INTO @nombre ,
             @Apellido1, @Apellido2,@salario,@NombreDepart
              CONTINUE
     END
     IF( @Salario >23000 and
           @NombreDepart in ('CONTABILIDAD','INFORMÁTICA'))
        UPDATE EMPLEADO
       SET Salario= Salario*1.05
        WHERE CURRENT OF SubirSalarioMujeres_cursor
      else
        if(@Salario >24000 and @NombreDepart='TÉCNICO')
          UPDATE EMPLEADO
          SET Salario= Salario*1.10
          WHERE CURRENT OF SubirSalarioMujeres_cursor

    FETCH NEXT FROM SubirSalarioMujeres_cursor INTO @nombre ,
        @Apellido1, @Apellido2,@salario,@NombreDepart
 END
 PRINT 'Estado de FETCH: ……..' + CAST(@@FETCH_STATUS AS varchar(2))

CLOSE SubirSalarioMujeres_cursor
DEALLOCATE SubirSalarioMujeres_cursor
```