Pregrado

Programa de Ingeniería de Sistemas

GESTIÓN DE DATOS E INFORMACIÓN II

Sesión 8

Tema:

Manejo de cursores con procedimientos almacenados

Resultado de aprendizaje

Implementa procedimientos almacenados, cursores y triggers sobre una base de datos utilizando T-SQL para resolver las diversas actividades de una organización.

Evidencia de aprendizaje

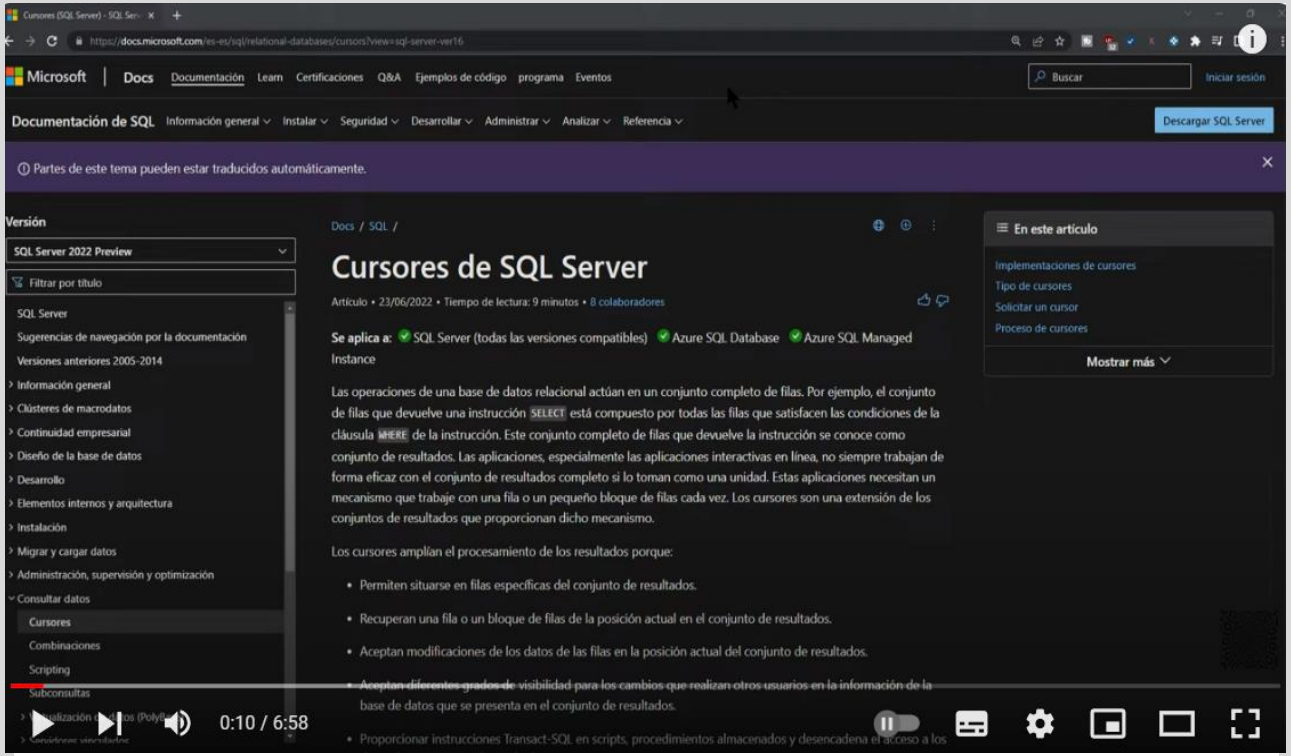
Los estudiantes demostrarán su capacidad para manejar cursores para procesar filas de resultados de manera iterativa que permitirá aplicar y comprender su uso en el contexto de procedimientos almacenados.

Contenido

Nombre del tema

- Manejo de cursores con procedimientos almacenados

Revisa el siguiente video:



https://youtu.be/8QYhuzIL_K4?si=pz8ttmSJbQUODvVD

Después de haber visualizado el video en la slide anterior, reflexionamos y respondemos las siguientes interrogantes:

01 ¿Qué es un cursor en SQL Server?

02 ¿Para qué sirve los cursores?

03 ¿Qué ventajas ofrece utilizar cursores?

Tema

cursores con procedimientos almacenados

CURSORES

Un cursor es una estructura de datos creada en memoria RAM producto de una sentencia SELECT y que nos permite navegar dentro de las filas para obtener la información.

Los cursores extienden el procesamiento de resultados al:

- ✓ Permitir el posicionamiento en filas específicas del conjunto de resultados.
- ✓ Recuperar una fila o un bloque de filas de la posición actual en el conjunto de resultados.
- ✓ Admite modificaciones de datos en las filas en la posición actual en el conjunto de resultados.

CURSORES

Los cursores extienden el procesamiento de resultados al:

- ✓ Admitir diferentes niveles de visibilidad de los cambios realizados por otros usuarios en los datos de la base de datos que se presentan en el conjunto de resultados.

- ✓ Proporcionar declaraciones Transact-SQL en scripts, procedimientos almacenados y desencadena el acceso a los datos en un conjunto de resultados.

CURSORES

Cada cursor contiene las siguientes 5 partes:

- ✓ **Declarar cursor:** en esta parte declaramos variables y devolvemos un conjunto de valores.
- ✓ **Abrir el cursor:** Esta es la parte inicial del cursor.
- ✓ **Recuperar los datos:** se utiliza para recuperar los datos fila por fila desde un cursor.
- ✓ **Cerrar el cursor:** Esta es una parte de salida del cursor y se usa para cerrar un cursor.
- ✓ **Desalojar:** en esta parte eliminamos la definición del cursor y liberamos todos los recursos del sistema (memoria) asociados con el cursor.

CURSORES: SINTAXIS

Cada cursor

```
DECLARE <Nombre_Cursor> CURSOR [ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR <Declaración_Select>
[ FOR UPDATE [ OF <Nombre_Columna>[ ,…n ] ] ] [;]
```

ALCANCE DEL CURSOR

Microsoft SQL Server admite las palabras clave **GLOBAL** y **LOCAL** en la instrucción **DECLARE CURSOR** para definir el alcance del nombre del cursor.

GLOBAL: especifica que el cursor es global para la conexión, es decir se puede usar en cualquier momento y por cualquier procedimiento, trigger, función o consulta en la sesión.

LOCAL: especifica que el cursor es local para el Procedimiento almacenado, el desencadenador o la consulta que contiene el cursor.

OPCIÓN DE RECUPERACIÓN DE DATOS EN CURSORES

Microsoft SQL Server admite las siguientes dos opciones de recuperación de datos:

FORWARD_ONLY: especifica que el cursor solo se puede desplazar desde la primera hasta la última fila.

SCROLL: proporciona 6 opciones para obtener los datos (FIRST, LAST, PRIOR, NEXT, RELATIVE y ABSOLUTE).

TIPOS DE CURSORES

Microsoft SQL Server admite los siguientes 4 tipos de cursores.

STATIC

Un cursor estático llena el conjunto de resultados durante la creación del cursor y el resultado de la consulta se almacena en caché durante la vida útil del cursor. Un cursor estático puede moverse hacia adelante y hacia atrás.

FAST_FORWARD

Este es el tipo de cursor predeterminado. Es idéntico al estático, excepto que solo puede desplazarse hacia adelante.

TIPOS DE CURSORES

DINAMIC

En un cursor dinámico, las adiciones y eliminaciones son visibles para otros en la fuente de datos mientras el cursor está abierto.

KEYSET

Esto es similar a un cursor dinámico, excepto que no podemos ver registros que otros agreguen. Si otro usuario elimina un registro, es inaccesible desde nuestro conjunto de registros.

BLOQUEOS

El bloqueo es el proceso por el cual un **DBMS** restringe el acceso a una fila en un entorno multiusuario.

Cuando una fila o columna se bloquea exclusivamente, otros usuarios no pueden acceder a los datos bloqueados hasta que se libere el bloqueo. Se utiliza para la integridad de los datos. Esto garantiza que dos usuarios no puedan actualizar simultáneamente la misma columna en una fila.

TIPOS DE BLOQUEOS

Microsoft SQL Server admite los siguientes tres tipos de bloqueos.

READ_ONLY

Especifica que el cursor no se puede actualizar.

SCROLL_LOCKS

Proporciona integridad de datos en el cursor. Especifica que el cursor bloqueará las filas a medida que se leen en el cursor para garantizar que las actualizaciones o eliminaciones realizadas con el cursor tengan éxito.

OPTIMISTIC

Especifica que el cursor no bloquea las filas a medida que se leen en el cursor. Por lo tanto, las actualizaciones o eliminaciones realizadas con el cursor no tendrán éxito si la fila se ha actualizado fuera del cursor.

FETCH_STATUS

Cuando trabajamos con cursores, la función **@@FETCH_STATUS** nos indica el estado de la última instrucción **FETCH** emitida, los valores posibles son:

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 recuperada.

CURSORES: EJEMPLO N°1

```
USE NORTHWIND
GO
–DECLARANDO EL CURSOR
DECLARE CURSOR1 CURSOR SCROLL
  FOR SELECT * FROM DBO.CUSTOMERS
–ABRIR EL CURSOR
OPEN CURSOR1
–NAVEGAR
FETCH FIRST FROM CURSOR1
–CERRAR EL CURSOR
CLOSE CURSOR1
–LIBERAR DE MEMORIA
DEALLOCATE CURSOR1
```

CURSORES: EJEMPLO N°2

```
DECLARE @ProductoName AS nvarchar(400)
DECLARE ProdInfo CURSOR FOR
SELECT ProductName FROM Products
OPEN ProdInfo
FETCH NEXT FROM ProdInfo INTO @ProductoName
WHILE @@fetch_status = 0
BEGIN
    PRINT @ProductoName
    FETCH NEXT FROM ProdInfo INTO @ProductoName
END
CLOSE ProdInfo
DEALLOCATE ProdInfo
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 1:

Con el siguiente código estamos creando el procedimiento y creando dos variables de tipo varchar.

```
CREATE PROCEDURE [dbo].[PA_CUR_SEL_ARCHIVOS]
@_ID_FRM VARCHAR(MAX),@PATH VARCHAR(MAX) AS
BEGIN TRY
DECLARE @ID_FRM VARCHAR(MAX),@ID_ARCHIVO VARCHAR(MAX)
SET NOCOUNT ON
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 2:

En el siguiente paso estamos declarando el cursor con nombre “CUR_SEL_ARCHIVOS”.

```
DECLARE CUR_SEL_ARCHIVOS CURSOR  FOR
```

PASO 3:

Estamos ejecutando una consulta de tipo “SELECT“, llamando dos columnas “ID_FRM” y “ID_ARCHIVO” , las que usaremos más adelante.

```
SELECT ID_FRM,ID_ARCHIVO
FROM ARCHIVOS WITH(NOLOCK)
WHERE ID_FRM = @_ID_FRM OPTION(KEEPFIXED PLAN)
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 4:

Abrimos el cursor con la sentencia “OPEN CUR_SEL_ARCHIVOS“, luego recorremos el primer registro obtenido de la consulta realizada en el paso 3, para luego usar esas columnas en la sentencia “FETCH NEXT FROM CUR_SEL_ARCHIVOS INTO @ID_FRM,@ID_ARCHIVO“, por cada uno de los registros de la consulta.

Para repetir el proceso hasta recorrer todos los registros de la consulta del paso 3 hasta terminar de recorrer todos los registros “WHILE @@fetch_status = 0“.

```
OPEN CUR_SEL_ARCHIVOS
FETCH NEXT FROM CUR_SEL_ARCHIVOS INTO @ID_FRM,@ID_ARCHIVO
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 5:

En este paso podemos ejecutar cualquier consulta “INSERT“, “UPDATE“, “DELETE“, o un procedimiento almacenado como el siguiente ejemplo usando los parámetros anteriormente obenidos.

```
WHILE @@fetch_status = 0
 BEGIN
 EXEC dbo.PA_EXPORT_ARCHIVOS @ID_FRM,@PATH,@ID_ARCHIVO
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 6:

Volvemos a comprobar si existen más registros por recorrer.

```
FETCH NEXT FROM CUR_SEL_ARCHIVOS INTO @ID_FRM,@ID_ARCHIVO
END
CLOSE CUR_SEL_ARCHIVOS
```

PASO 7:

Cerramos el cursor con nombre “CUR_SEL_ARCHIVOS“, y finalmente matamos el cursor con nombre “CUR_SEL_ARCHIVOS“.

```
DEALLOCATE CUR_SEL_ARCHIVOS
SET NOCOUNT OFF
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 8:

Finalmente ejecutamos la sentencia catch si ocurre algún error aparecerá.

```
END TRY
 BEGIN CATCH
  SELECT ERROR_NUMBER() AS errNumber
  , ERROR_SEVERITY() AS errSeverity
   , ERROR_STATE() AS errState
   , ERROR_PROCEDURE() AS errProcedure
    , ERROR_LINE() AS errLine
     , ERROR_MESSAGE() AS errMessage
      ROLLBACK TRAN CREA_EXPEDIENTES
  END CATCH
```

CURSORES DENTRO DE UN PROCEDIMIENTO ALMACENADO

PASO 9:

Para ejecutar realizar lo siguiente.

```
EXEC [dbo].[PA_CUR_SEL_ARCHIVOS] 1,'C:\archivo.txt
```

Autoevaluación

Sesión 8

Pregunta 1

¿Cuál es el propósito principal de un cursor en SQL?

- Almacenar datos temporalmente.
- Recorrer y procesar filas de resultados de consultas de manera iterativa.
- Optimizar consultas complejas
- Ninguna.

Pregunta 2

¿Qué tipo de operaciones se pueden realizar utilizando un cursor en SQL?

- Solo operaciones de lectura.
- Operaciones de lectura y escritura.
- Operaciones de escritura exclusivamente.
- Todas.

Pregunta 3

¿Cuál es una consideración importante al utilizar cursores en términos de rendimiento?

- Los cursores siempre mejoran el rendimiento de las consultas.
- Es mejor evitar el uso de cursores para garantizar un rendimiento óptimo.
- El rendimiento no se ve afectado por el uso de cursores en SQL.
- Todas.

Autoevaluación

¡Vamos por más logros!

¡Felicitaciones!

Ha concluido la autoevaluación

Conclusiones

La utilización de **cursores** en **procedimientos almacenados** es una herramienta poderosa, pero su necesidad debe evaluarse en contextos específicos. Se debe considerar si existe una alternativa más eficiente para lograr el mismo resultado, ya que los cursores pueden impactar negativamente en el rendimiento.

Los **cursores** permiten una iteración controlada de los resultados de una consulta, lo que es esencial cuando se requiere procesar filas de manera individual y aplicar lógica de negocios específica a cada una. Esto proporciona una mayor flexibilidad en la manipulación de datos.

Es crucial implementar un sólido manejo de condiciones y excepciones al trabajar con **cursores**. Esto garantiza un comportamiento predecible del procedimiento almacenado incluso en situaciones inesperadas, como errores durante la iteración del cursor.

La optimización del rendimiento es un aspecto crítico al usar cursores. Se deben aplicar estrategias de diseño y codificación para minimizar el impacto negativo en la velocidad de ejecución. La elección de índices adecuados y la optimización de la lógica de procesamiento son consideraciones clave.

Aplicando lo aprendido:

Desarrollar la Guía de Laboratorio N°8

Referencias

CAPACHO, José y Wilson NIETO. Diseño de Bases de Datos [en línea]. Barranquilla: Universidad del Norte, 2017. ISBN 9789587418255. Disponible en: https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1690049&lang=es&site=ehost-live

WANUMEN Luis, RIVAS Edwin, Mosquera Darín. Bases de datos en SQL Server [en línea]. Bogotá: Ecoe Ediciones, 2017. ISBN 9789587715705. Disponible en: https://www.digitaliapublishing.com/a/66605

HUESO Luis. Bases de datos [en línea]. Madrid: Rama Editorial, 2014. ISBN 9788499641577.
Disponible en: https://www.digitaliapublishing.com/a/109943

PRIETO, Rafael. SGBD e instalación: administración de bases de datos (UF1469) [en línea]. Antequera : IC Editorial. ISBN 9788416433360. Disponible en: https://www.digitaliapublishing.com/a/86830

Pregrado