Cursos SQL Server 2008 R2

Cursores

Cursos SQL Server 2008 R2

Índice

T-SQL - Cursores

PROCESAMIENTO FILA A FILA

La manera más eficaz de procesar consultas es mediante el conjunto de resultados predeterminados, que es el modo para el que SQL Server está optimizado. El orden en el que se procesan las filas lo decide el optimizador de consultas para hacer el trabajo lo más eficiente posible.

En ciertos casos puede ser necesario procesar las filas individuales de un conjunto de resultados predeterminado en un orden específico. En este caso podemos usar cursores. SQL Server admite cursores Transact-SQL y cursores de aplicación.

Existen dos formas totalmente distintas en las que se puede aplicar un proceso de negocio a un conjunto de filas:

- Recorrer el conjunto de resultados en la forma en que se prefiera y aplicar el proceso de negocio a cada fila individualmente, enviando a SQL Server una o más instrucciones Transact-SQL para cada fila. (***1-FilaPorFila.sql)***
- Enviar a SQL Server una única instrucción Transact-SQL que describa el modo de aplicar el proceso de negocio a todo el conjunto de resultados y dejar que SQL Server decida como aplicarlo de la mejor forma. (***1-Update.sql)***

TIPOS DE CURSORES.

SQL Server permite el uso de cuatro tipos de cursores:

- Cursores de desplazamiento solo hacia delante.
- Cursores estáticos.
- Cursores dinámicos.
- Cursores controlados por conjunto de claves.

Los ***estáticos*** usan más espacio de almacenamiento que los dinámicos o los controlados por conjunto de claves, pero una vez creados, SQL Server usa menos recursos para recorrer los datos. Un cursor estático suministra un conjunto fijo de datos que no detecta cambios hechos por otras conexiones.

Los ***dinámicos*** usan poco espacio, pero usan más recursos para su desplazamiento que los estáticos o los controlados por claves. Un dinámico suministra un conjunto de datos flexible, que refleja los cambios hechos a los datos por otras conexiones.

Los cursores controlados por ***conjuntos de claves*** representan un equilibrio entre las necesidades de espacio de almacenamiento y los recursos de navegación. Estos cursores suministran un conjunto de datos con una cantidad fija de filas, que refleja los cambios hechos por otras conexiones.

Los de ***desplazamiento solo hacia delante*** no se consideran como un tipo de cursor distinto, sino como una propiedad opcional que pueden tener los otros tipos de cursores.

CURSORES SOLO HACIA DELANTE

La lectura de cada fila no se lleva a cabo hasta que se solicite mediante la instrucción FETCH. Por esta razón, cuando se usa este tipo de cursor podemos ver los cambios hechos por cualquier conexión a los datos que aún no hemos leído. Los cambios a los datos ya leídos no se pueden ver, porque este tipo de cursor no permite desplazamientos hacia atrás.

Microsoft suministra una mejora que es una variante a este tipo y se denomina FAST_FORWARD que es un cursor optimizado de desplazamiento rápido hacia delante.

Existen situaciones en las que esta opción no es válida:

- La instrucción SELECT hace referencias a columnas BLOB, por ejemplo columnas de tipo TEXT, NTEXT o IMAGE.
- El cursor no está abierto como de solo lectura.
- La consulta hace referencia a tablas remotas ubicadas en servidores vinculados.

RECOMENDACIÓN: cuando se pueda, se deben usar los rápidos solo hacia delante.

CURSORES ESTÁTICOS

Se construye todo el conjunto de resultados en la base de datos ***tempdb***. En los desplazamientos no se ven los cambios que hagan otras instrucciones. La creación de la tabla de trabajo consume espacio y tiempo, por lo que la aplicación que use este cursor tardará más tiempo en abrirlo.

Una vez lleno y hasta el momento de su cierre, un cursor estático contiene un número fijo de filas, con valores fijos para las columnas.

El desplazamiento por la base de datos tempdb es muy rápido.

Este cursor no detecta la inserción de nuevas filas, la eliminación de las existentes y los cambios a los valores de las columnas, salvo que lo cerremos y volvamos a abrirlo.

2. Cursor-Estático.sql

CURSORES DINÁMICOS

Este tipo de cursor refleja todos los cambios realizados por otras conexiones.

No genera una tabla temporal en la base de datos ***tempdb***, y debido a ello, se abre más rápidamente y usa menos espacio de almacenamiento que los estáticos.

Una vez abierto y hasta el momento de su cierre, es indefinida la cantidad de filas que contiene y los valores de las columnas pueden cambiar.

Cada vez que hace falta cargar una nueva fila en el cursor, SQL Server tiene que volver a ejecutar el plan de consulta para leer la nueva fila y esa operación consume tiempo, lo que provoca un desplazamiento más lento del cursor que para el caso de un cursor estático.

Este tipo de cursor refleja todos los cambios que se hagan a los datos existentes, ya sean altas, eliminaciones o modificaciones.

3. Cursor-Dinámico.sql

CURSORES CONTROLADOS POR CONJUNTOS DE CLAVES

Permiten resolver algunos de los problemas que plantean los cursores estáticos y los dinámicos.

Este tipo de cursor crea en ***tempdb*** una lista de valores no repetidos tomados de la consulta original, llamado conjunto de claves, en la que cada clave identifica de forma unívoca una sola fila del conjunto de resultados. Este conjunto de claves contiene marcadores que apuntan a los datos reales.

Dado que el conjunto de claves se construye por completo en el momento de abrir el cursor, la cantidad de filas permanecerá invariable hasta el cierre del cursor. Sin embargo, si se eliminan filas de las tablas subyacentes, al intentar cargar esas filas se producirá un mensaje de error por "fila inexistente" ya que los marcadores de estas filas apuntan a una ubicación no válida.

Las filas que se inserten en los datos y pudieran considerarse parte del conjunto de resultados no serán visibles a menos que cerremos el cursor y volvamos a abrirlo. Los cambios hechos por cualquier conexión a los valores de las columnas son visibles dentro del cursor, siempre que las columnas modificadas no formen parte del conjunto de claves.

ÓRDENES DE MANIPULACIÓN DE CURSORES.

Declaración del cursor.

La orden es DECLARE.

Se puede indicar si un cursor es:

- LOCAL: del lote, procedimiento almacenado, trigger o UDF donde se define el cursor.
- GLOBAL: de la conexión.

Se puede indicar si un cursor es:

- FORWARD ONLY: solo hacia adelante.
- SCROLL: en ambas direcciones.

El bloqueo de los datos del cursor se puede controlar mediante la instrucción DECLARE CURSOR con las palabras clave READ_ONLY, SCROLL_LOCKS y OPTIMISTIC.

- READ_ONLY: se impide llevar a cabo las modificaciones a las filas subyacentes del cursor mediante las instrucciones UPDATE / DELETE y la cláusula WHERE CURRENT OF. ***(5. Cursor-READ_ONLY.sql)***
- SCROLL_LOCKS: obliga a SQL Server a bloquear los datos cuando se leen con el cursor, con el fin de garantizar la realización de actualizaciones. Es lo que se denomina "Bloqueo Pesimista".
- OPTIMISTIC: esta opción desbloquea los datos tras cargarlos en el cursor y no vuelve a bloquearlos salvo para modificarlos o eliminarlos, si fuera necesario. En este caso, SQL Server debe verificar si otras conexiones modificaron la fila entre las operaciones de lectura y de escritura.

La definición del cursor debe hacerse con una instrucción SELECT, que es una instrucción SELECT común, con algunas salvedades. En la instrucción SELECT que define un cursor no se pueden usar las cláusulas COMPUTE, COMPUTE BY, FOR BROWSE o INTO.

Nota:

Si la instrucción SELECT produce un conjunto de resultados no actualizable, el cursor será READ_ONLY. Esto se puede deber al uso de funciones de agregación, la falta de permisos suficientes o a la lectura de datos de solo lectura.

Se puede restringir el conjunto de columnas modificables mediante la cláusula FOR UPDATE. Esta cláusula se puede usar de dos formas:

- FOR UPDATE OF columna1, ... , columnaN: se debe usar esta opción para definir que las columnas 1, … , N se pueden modificar a través del cursor.
- FOR UPDATE: esta es la opción predeterminada, y declara que todas las columnas del cursor son actualizables.

Apertura del cursor

La orden utilizada es OPEN <nombre del cursor>

Si se trata de un cursor Static o Keyset, SQL Server debe crear una tabla de trabajo en la base de datos tempdb para guardar el conjunto de resultados entero (si se trata de un cursor Static) o el conjunto de claves (si es un cursor Keyset).

Para optimizar el uso de cursores voluminosos, SQL Server puede llenarlos de forma asincrónica. En este caso, SQL Server crea un nuevo hilo (thread) para llenar el cursor en paralelo y devuelve el control a la aplicación tan pronto como sea posible.

Para averiguar la cantidad de filas contenidas en el cursor se puede usar la función de sistema @@CURSOR_ROWS. Si el llenado del cursor se hace de manera asincrónica, el valor devuelto por @@CURSOR_ROWS es negativo y representa la cantidad aproximada de filas devueltas desde la apertura del cursor.

Para los cursores dinámicos @@CURSOR_ROWS devuelve -1, ya que no hay modo de saber si ya se devolvió todo el conjunto de resultados (debido a que otras operaciones que trabajen sobre los mismos datos podrían insertar nuevas filas).

7. @@Cursor_Rows.SQL

PRECAUCIÓN.

La función anterior devuelve la cantidad de filas del último cursor abierto en la conexión actual. Si dentro de un trigger usamos cursores, el resultado de llamar esta función desde el nivel principal de ejecución podría ser engañoso.

Para indicar a SQL Server cuándo debe llenar un cursor de forma asincrónica, se puede usar el procedimiento almacenado de sistema ***sp_configure*** y cambiar la opción de servidor "***cursor_threshold***" por el número máximo de filas que SQL Server debería leer directamente sin llenado asincrónico.

RECOMENDACIÓN.

No se debe configurar el valor de "cursor_threshold" demasiado bajo, porque tratándose de conjunto de resultados pequeños es más eficiente la apertura sincrónica.

Carga de filas.

Para desplazarnos usamos la instrucción FETCH.

FETCH FROM nombrecursor carga la siguiente fila en el cursor.

Es equivalente a FETCH NEXT FROM nombrecursor.

Posibilidades:

- FETCH PRIOR: mueve el puntero del cursor a la fila anterior.
- FETCH FIRST: mueve el puntero del cursor al principio del conjunto de resultados.
- FETCH LAST: mueve el puntero del cursor al final del conjunto de resultados.
- FETCH ABSOLUTE n: mueve el puntero del cursor a la enésima fila del conjunto de resultados. Si n es negativo, el puntero se mueve a n filas antes del final del conjunto de resultados. Si la nueva posición no existe, la instrucción devuelve una fila vacía, pero no se produce ningún mensaje de error. Si n es igual a 0 no se devuelve ninguna fila y el puntero apunta fuera del conjunto de resultados.
- FETCH RELATIVE n: mueve el puntero del cursor n filas delante de su posición actual. Si n es negativo, el puntero se mueve n filas hacia atrás desde su posición actual. Si la nueva posición no existe, la instrucción devuelve una fila vacía, pero no se produce ningún mensaje de error. Si n es igual a 0, la instrucción devuelve la fila actual.

Para verificar si después de la última instrucción FETCH el cursor quedó apuntando a una fila válida, se puede usar la función de sistema @@FETCH_STATUS, cuyo valor puede ser uno de los siguientes:

- 0, si la instrucción FETCH tuvo éxito y el cursor apunta a una fila válida.
- -1, si la instrucción no tuvo éxito el cursor apunta fuera de los límites del conjunto de resultados.

- -2, si el cursor está apuntando a una fila inexistente. Esto puede ocurrir en un cursor controlado por conjunto de claves, cuando se elimina una de las filas fuera del control del cursor.

PRECAUCIÓN

@@FETCH_STATUS es global para la conexión, por lo que refleja el estado de la última instrucción FETCH ejecutada en la conexión. Por eso es importante realizar la verificación inmediatamente después de la instrucción FETCH. ***(7. Fetch.sql)***

Al mismo tiempo que movemos el cursor con la instrucción FETCH podemos usar la cláusula **INTO** para cargar los campos del cursor directamente en variables definidas por el usuario. De esta forma podremos usar más tarde los valores almacenados en estas variables en otras instrucciones Transact-SQL.

Si el cursor es actualizable podemos modificar los valores en las tablas subyacentes mediante instrucciones UPDATE o DELETE comunes, para lo cuál indicaremos WHERE CURRENT OF nombrecursor como condición.

A través de un cursor se pueden modificar columnas que no forman parte de su definición, siempre que esas columnas sean actualizables.

8. Fetch INTO.sql

Cierre del cursor.

La instrucción CLOSE cierra el cursor y libera todos los bloqueos que usa. La estructura sigue existiendo, pero después de cerrarlo no lo podemos usar para leer sus datos.

Es importante cerrarlos una vez usados para mejorar la concurrencia de las aplicaciones.

Liberación del cursor.

La orden DEALLOCATE destruye el cursor. La única forma de reabrir el cursor es volver a definirlo.

Después de DEALLOCATE, el nombre del cursor queda disponible para reutilizarlo en la declaración de otro cursor.

Nota: para reutilizar el mismo cursor en diferentes ocasiones en un lote prolongado o en un procedimiento almacenado complejo conviene declarar el cursor tan pronto como sea necesario y liberarlo cuando deja de serlo.

Entre las instrucciones DECLARE y DEALLOCATE, se debe acceder a los datos usando OPEN y CLOSE tantas veces como sean necesarias, para evitar bloqueos de larga duración. Sin embargo, las aperturas pueden producir algunas sobrecargas.

Alcance de los cursores.

En la instrucción DECLARE CURSOR, se puede indicar su alcance. El alcance predeterminado es GLOBAL, pero es posible cambiar la opción predeterminada de la base de datos a cursor local.

Nota: es preferible no confiar en el alcance predeterminado de cursor de SQL Server. Es recomendable declarar explícitamente el cursor como LOCAL o GLOBAL.

Un cursor global se puede usar en cualquier sitio dentro de la misma conexión que lo creó, mientras que un cursor local solo es válido dentro del alcance del lote, procedimiento, función definida por el usuario o trigger donde ha sido creado. Cuando un cursor sale fuera del alcance, SQL Server lo libera de forma automática.

Sin embargo, también se puede asignar el cursor a un parámetro de salida de un procedimiento almacenado. En este caso, la liberación del cursor tendrá lugar cuando esté fuera de alcance la última variable de cursor que haga referencia a él.

Los cursores locales y globales tienen dos espacios de nombres diferentes, por lo que es posible tener un cursor global con el mismo nombre que uno local y definiciones completamente diferentes. Para evitar problemas potenciales, SQL Server usa cursores locales.

Cursores locales

Estos cursores representan un mecanismo de seguridad que permite la creación de cursores dentro de objetos independientes, tales como procedimientos almacenados, triggers y función definida por el usuario. Los cursores locales son más fáciles de manejar que los globales, ya que no hace falta tener en cuenta cambios potenciales al cursor en otros procedimientos almacenados o triggers usados por la aplicación.

Cursores globales

Son útiles en situaciones en las que diferentes procedimientos deben manejar un conjunto de resultados común e interactuar dinámicamente con él.

Nota:

Se recomienda el uso de cursores locales siempre que sea posible. Si se necesita compartir un cursor entre dos procedimientos se debería considerar (como alternativa) el uso de una variable de cursor.

Uso de variables de cursor.

Es posible declarar variables con el tipo de datos CURSOR.

Esto es útil cuando necesitamos enviar una referencia del cursor a otro procedimiento o función definida por el usuario.

SQL Server ofrece varios procedimientos almacenados de sistema que permiten obtener información acerca de los cursores. Estos procedimientos usan variables de cursor para comunicar su información:

- *Sp_cursor_list*: devuelve la lista de los cursores disponibles en la conexión actual.
- *Sp_describe_cursor*: lee los atributos de un cursor abierto.
- *Sp_describe_cursor_columns*: describe las columnas que lee el cursor.
- *Sp_describe_cursor_tables*: obtiene información sobre las tablas que usa el cursor.
- T-SQL - Cursores

Estos procedimientos almacenados usan variables de cursor para la lectura de los resultados. De este modo, los procedimientos o lotes que llamen a estas utilidades pueden usar el resultado fila a fila.

9. Variable Cursor.sql

Uso de cursores en triggers para resolver acciones sobre varias filas

En muchos casos, el llevar a cabo operaciones sobre varias filas dentro de un trigger no es una tarea sencilla. Los cursores pueden ser utilizados para poder llevarlas a cabo.

Supongamos el siguiente ejemplo: queremos asignar a cada cliente un límite de crédito siguiendo un proceso automatizado cuya aplicación corre por cuenta del procedimiento almacenado "*AsignarLimiteCredito*". Para automatizar el proceso podemos crear un trigger AFTER INSERT que calcule el crédito para cada nuevo cliente.

10. Trigger-Cursores.sql

Más información

C/ Miracruz, 10 (Bº de Gros) 20001 Donostia

Telf.: 943 275819

email: s e i m @ c e n t r o s e i m . c o m

S.E.I.M.