¿CÓMO MEJORAR EL RENDIMIENTO DE LA EJECUCIÓN DE CONSULTAS SQL EN BASES DE DATOS RELACIONALES?

M. en C. Carlos De La Cruz Sosa
Academia de Telemática IPN – UPIITA
M. en C. Paola Nayeli Cortez Herrera
Academia de Informática IPN – UPIITA

Resumen

En muchas ocasiones el rendimiento de las consultas que se ejecutan en un servidor de base de datos no es la óptima, esto se debe generalmente a la forma en que esta estructura la consulta y al mal uso que se hace de los índices. En el presente trabajo se exponen recomendaciones para escribir consultas SQL así como el uso de planes de ejecución a fin de mejorar el rendimiento en la ejecución de las consultas.

Introducción

Una consulta tiene muchas posibles estrategias de ejecución y el proceso de elegir la más adecuada para procesarla se conoce como optimización de consulta. En los DBMS (DataBase Management Systems, por sus siglas en inglés), se incluye un módulo de optimización de consultas que se encarga de producir un plan de ejecución para ejecutar la consulta.

Un plan de ejecución es un conjunto de pasos que lleva acabo el procesador de consultas de un DBMS para ejecutar una consulta. Es decir, para cada consulta que se desea ejecutar se genera un plan de ejecución. El rendimiento de la ejecución de una consulta depende de muchos factores entre los que destacan: características del hardware del servidor donde está instalado el DBMS, infraestructura de red que se emplea para acceder al DBMS, la forma en que están escritas las consultas y los archivos de índices empleados en ellas. A continuación explicaremos como estos dos últimos factores influyen en el rendimiento de la ejecución de consultas SQL (Structured Query Language).

Optimización de consultas y planes de ejecución

El proceso de optimización de consultas se puede realizar empleando algebra relacional o analizando planes de ejecución para determinar las acciones a considerar para mejorar el rendimiento de las consultas.

Independientemente del enfoque que se utilice para optimizar la consulta debemos de tener en cuenta las siguientes recomendaciones al momento de escribir una consulta SQL:

1. Si la consulta utiliza cursores, determine si se puede escribir la consulta de cursor con un tipo de cursor más eficaz (como un cursor de sólo avance rápido) o con una única consulta. Las consultas únicas normalmente mejoran las operaciones de cursor. Debido a que un conjunto de instrucciones de cursor

suele constituir una operación de bucle externo, en la que cada fila del bucle externo se procesa una vez con una instrucción interna, considere la posibilidad de utilizar en su lugar una instrucción GROUP BY, o una subconsulta.

2. Si una aplicación ejecuta la consulta mediante un bucle, considere la posibilidad de colocar el bucle en la consulta. A menudo, una aplicación contendrá un bucle que, a su vez, contendrá una consulta con parámetros que se ejecuta muchas veces y será necesario realizar un viaje de ida y vuelta en la red entre el equipo que ejecuta la aplicación y el DBMS. En su lugar, cree una sola consulta más compleja con una tabla temporal. Sólo necesita un viaje de ida y vuelta en la red, y el optimizador de consultas puede optimizar mejor la consulta única.

3. No utilice varios alias para una sola tabla en la misma consulta para simular la intersección de índices. Ya no es necesario debido a que muchos DBMS lo realizan automáticamente la intersección de índices y se puede utilizar varios índices en la misma tabla de la misma consulta.

4. Utilice consultas parametrizadas para permitir la reutilización de los planes de ejecución almacenados en la memoria caché. Si un conjunto de consultas tiene el mismo hash de consulta y hash de plan de consulta, podría mejorar el rendimiento creando una consulta parametrizada.

A continuación describiremos la optimización de consultas empleando el enfoque de análisis de planes de ejecución.

Análisis de planes de ejecución para mejorar el rendimiento de consultas

Los DBMS incorporan herramientas para generar planes de ejecución estimados y reales.

Un plan de ejecución estimado consiste en mostrar los posibles operadores que el procesador de consultas utilizará en la ejecución, así como los costos probables de entrada/salida, uso de procesador y porcentajes de procesamiento empleado por cada operador. En este proceso no se ejecuta la consulta, simplemente se analiza y determina el conjunto de operadores que se pueden emplear de acuerdo a como está estructurada la consulta, este plan de ejecución permite observar el comportamiento que tendrá la ejecución y así poder determinar cambios en la estructura de la consulta, desde el hecho de reescribirla o añadir archivos índices de forma estratégica a fin de optimizar su ejecución.

Un plan de ejecución real consiste en mostrar los costos reales de ejecución de una consulta, en este proceso si se ejecuta la consulta y al final de su ejecución se muestra el plan de ejecución empleado.

Los planes de ejecución muestran el conjunto de operadores empleado por el procesador de consultas para resolver la consulta, así como la interacción entre los operadores. Los operadores a emplear varían según la estructura de la consulta y si la(s) tabla(s) involucradas poseen o no índices (agrupados y no agrupados).

Los operadores que se pueden emplear en un plan de ejecución cambian de nombre entre un DBMS y otro, pero tienen funcionalidad equivalente.

A continuación explicaremos los operadores más comunes en el DBMS SQL Server:

1.**TABLE SCAN.** Esto indica que el motor necesita leer completamente la tabla sin utilizar un índice. En la mayoría de los casos su aparición quiere decir que estamos haciendo mal las cosas, y necesitamos urgente crear un índice o reestructurarlo si ya existe alguno. Aunque no siempre es así, el motor siempre intenta predecir los costos de ejecución basados en las estadísticas que va almacenando, si él estima que va ser más rápido leer toda la tabla en vez de leer un índice, usará ese método. Esto suele suceder con tablas poco pobladas y si la consulta en cuestión no incluye ningún filtro.

2.**CLUSTERED INDEX SCAN ó INDEX SCAN.** Es semejante al Table Scan, ya que recorre completamente la tabla pero utilizando ésta vez alguno de los índices clustered o non clustered. Aparece normalmente en tablas que están pobladas considerablemente.

3.**CLUSTERED INDEX SEEK ó INDEX SEEK.** Este operador realiza una búsqueda específica empleando un valor del archivo índice.

4.**BOOKMARK LOOKUP.** Este aparece cuando se requiere hacer un salto del apuntador del índice non-clustered a la páginas de datos real. Una de las maneras para evitar su aparición excesiva es limitar los campos requeridos en la consulta, sólo solicitar los que están incluidos en el índice, ésta es la principal razón por la que habrán escuchado más de una vez, "no escribas consultas SELECT * FROM tabla".

5. JOIN. El operador JOIN se divide en tres tipos de JOIN nuevamente. Entran en acción cuando justamente se hacen uso de las cláusulas JOIN para unir dos o más tablas en la consulta, el tipo a emplear está determinado normalmente por el volumen de datos con el que se trabajará para que SQL Server elija usar uno u otro. NESTED LOOP JOIN (volumen de datos relativamente pequeño), MERGE JOIN (volumen de datos considerablemente grande), HASH JOIN (para grandes volúmenes de datos, especialmente si carecen de índice).

6. HASH MATCH. Este operador se presenta cuando el DBMS está comparando contenido, puede aparecer en un JOIN, WHERE y son lugares donde no deberían estar, lo hacen por falta de índices principalmente. Donde sí son muy útiles es cuando incluimos la cláusula DISTINCT, UNION, UNION ALL, en donde no solo se compara el valor de un campo, sino de todo un conjunto de columnas o incluso filas y columnas.

Indistintamente del operador del que se trate, se debe de poner especial atención en los detalles que se generan en el plan de ejecución en cuantos a los costos y porcentaje de procesamiento que consume cada uno de los operadores, para así determinar las acciones a realizar para mejorar la consulta. El siguiente ejemplo ilustra un plan de ejecución estimado de una consulta con los detalles de costo.

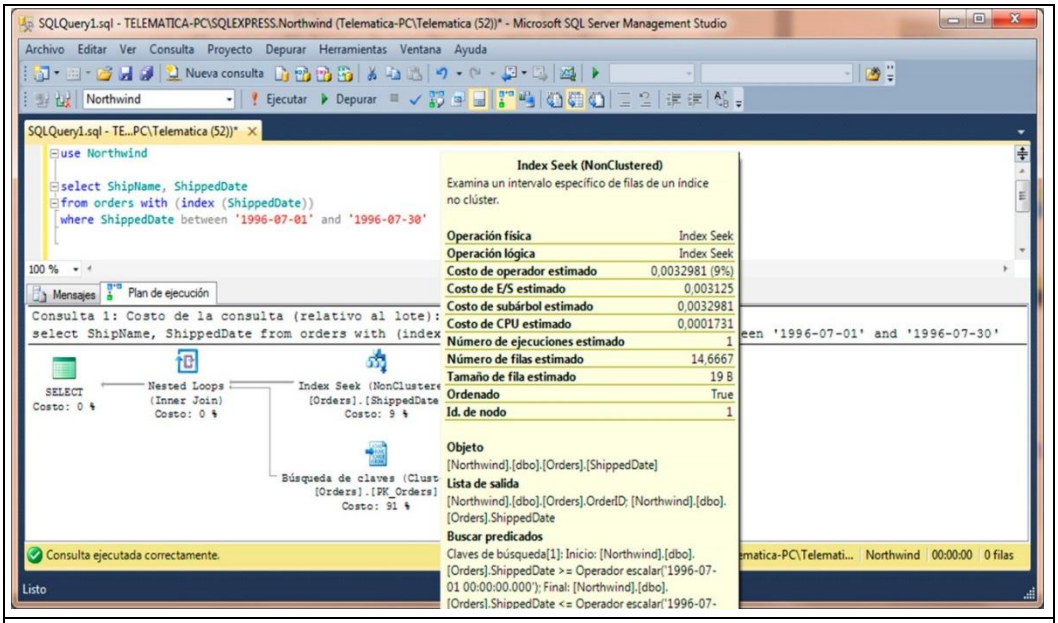

Figura 1. Consulta SQL con ejecución mediante un índice específico y plan de ejecución

En la figura 1 se puede observar la forma en que SQL Server muestra un plan de ejecución. En cada plan de ejecución hay que observar los costos que consume cada operador, en especial los costos de E/S , de CPU y de subárbol. Estos costos son determinantes para decidir añadir un archivo índice en caso de que no se esté empleando.

Conclusiones

La optimización de consultas es un proceso importante en la ejecución de una consulta, ya que este proceso es el encargado de seleccionar las operaciones que convienen aplicar para resolver la consulta consumiendo la menor cantidad de recursos posibles. En la optimización basada en el análisis de planes de ejecución es importante observar el comportamiento de cada operador en relación a los costos estimados o reales que implica la ejecución ya que estos valores en conjunto con el tipo de operador permite decidir si se añade y/o elimina un índice en particular o si se reescribe la consulta empleando otros índices de forma específica.

Referencias

1. Groff, James R. (2002). Manual de referencia SQL. Mc Graw Hill ISBN 007-222559-9

2. Elmasri, A. Ramez, Navathe, Shamkant B. (2007). Fundamentos de sistemas de bases de datos (5th ed.). Addison Wesley. ISBN 978-84-7829-085-7.

3. Grimpi IT Blog. (2008). Entender el plan de ejecución de SQL Server 2005/2008. Accedido el 15 de octubre de 2013, de http://grimpidev.wordpress.com/2008/10/28/entender-el-plan-de-ejecucion-en-sql-server-20052008/.

4. Tech Microsoft. (2013). Mostrar planes de ejecución gráficos (SQL Server Management Studio. Accedido el 15 de octubre de 2013, de http://technet.microsoft.com/es-es/library/ms178071(v=sql.105).aspx

5. Tech Microsoft. (2013). Rendimiento de las consultas. Accedido el 11 de octubre de 2013, de http://technet.microsoft.com/es-es/library/ms190610(v=sql.105).aspx 6. MSDN Blog. (2013). http://blogs.msdn.com/b/apinedo/archive/2007/01/24/mejorar-el-rendimiento-de-queries-en-sql-server.aspx. Accedido el 11 de octubre de 2013.