Por Milos Radivojevic

El Problema del rastreo de parámetros

con procedimientos almacenados de SQL Server

Los usuarios afirman que de repente la ejecución de un procedimiento almacenado es muy lenta. Sin embargo, cuando se ejecuta dentro de SSMS con los mismos parámetros, se ejecuta con rapidez. Además, los usuarios afirman que, para algunas combinaciones de parámetros, el procedimiento almacenado funciona bien. Si estás en este escenario, están descubriendo el lado oscuro del rastreo de parámetros.

¿Qué es el rastreo de parámetros?

Siempre que se invoca un procedimiento almacenado, el optimizador de consultas trata de volver a utilizar un plan de ejecución. Si existe un plan de ejecución coincidente en caché se reutiliza "a ciegas". Si no es así, se generará un nuevo plan. Durante la generación del plan, el optimizador analiza y optimiza todas las consultas en el procedimiento almacenado. Comprueba las posibilidades de generar físicamente un conjunto de resultados y considera varios factores: existencia de índices en las columnas que participan en cláusulas JOIN y WHERE, distribución de datos y el tamaño de tabla para las tablas involucradas, etc., a fin de estimar la selectividad de la consulta. Cuando se genera un plan de ejecución para un procedimiento almacenado, además de todos estos factores, el optimizador considera también los parámetros que se envían en la invocación de procedimiento. El optimizador "huele" estos parámetros en las consideraciones del plan. Es posible que los parámetros no tengan ninguna influencia sobre la estructura y la selectividad de la consulta resultante, pero generalmente, los valores de parámetro afectan significativamente esa selectividad y el plan de ejecución generado. ¿Esto es malo? Podría ser. Si los valores de parámetro para las llamadas posteriores al procedimiento almacenado tienen valores similares (de hecho, cuando sus valores no cambian la selectividad de las consultas de procedimiento) el rastreo de parámetro es importante. De lo contrario podría reducir drásticamente rendimiento. Tomemos el siguiente procedimiento almacenado en la base de datos de *Adventure Works* como ejemplo:

Listado 1: Creación de muestra de procedimiento almacenado

```
CREATE PROCEDURE dbo.getSalesOrderHeader
@SalesOrderID int
AS
            SELECT SalesOrderID,OrderDate,ShipDate,SubTotal
            FROM Sales.SalesOrderHeader
            WHERE SalesOrderID = @SalesOrderID
```

Este procedimiento almacenado devuelve cuatro columnas de la tabla **Ventas.SalesOrderHeader** para un determinado **SalesOrderID.** Vamos a invocarlo por primera vez con el parámetro **45671**. Como la columna **SalesOrderID** es una clave principal agrupada de la tabla, el plan de ejecución para este procedimiento es simple: una búsqueda de índice agrupado. Consulte la figura 1, página 45.

¿Así pues, dónde está aquí el rastreo de parámetros? Si elige la opción *Mostrar XML... de Plan de ejecución* puede obtener una representación

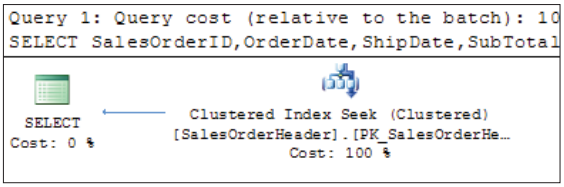

Figura 1: Plan de ejecución para el procedimiento almacenado del Listado 1

Listado 2: Crear procedimiento almacenado con el operador de rango

```
CREATE PROCEDURE dbo.getSalesOrderHeader
@OrderDate datetime
AS
          SELECT SalesOrderID,OrderDate,ShipDate,SubTotal
          FROM Sales.SalesOrderHeader
          WHERE OrderDate >= @OrderDate
```

XML del plan de ejecución. Bajo el nodo *ParameterList* se puede ver que tiene un valor *ParameterCompiledValue* de 45671 y este parámetro fue considerado por la generación del plan.

```
<ParameterList>
  <ColumnReference Column="@SalesOrderID" ParameterCompiledValue="(45671)" ParameterRuntimeValue="(45671)" />
</ParameterList>
```

Figura 2: Versión XML del plan de ejecución para el procedimiento almacenado del listado 1 (fragmento)

Si lo llamamos nuevamente con otros parámetros (por ejemplo 56781) podemos ver que *Parameter-CompiledValue* todavía tiene el valor 45671; sólo cambió *ParameterRuntimeValue*.

```
<ParameterList>
  <ColumnReference Column="@SalesOrderID" ParameterCompiledValue="(45671)" ParameterRuntimeValue="(56781)" />
</ParameterList>
```

Figura 3: Versión XML del plan de ejecución para el procedimiento almacenado del listado 1 con un parámetro diferente (fragmento)

El plan de ejecución generado es óptimo para el valor del parámetro 45671. ¿Es esto un problema? ¡Para nada! Este plan también es óptimo para el valor 56781. En realidad, para este procedimiento almacenado, solo tiene sentido un plan ejecución: una búsqueda de índice agrupado. El procedimiento devuelve exactamente una fila o un conjunto vacío, si no hay ningún pedido para un Id de pedido determinado. El valor del parámetro para la primera invocación del procedimiento almacenado no afecta a la selectividad de la consulta resultante. Esa es la razón por la que, en este caso, el efecto del rastreo de parámetros puede ser ignorado. Por supuesto es importante definir ***que*** fila debe devolverse pero no afecta a la manera de ***Cómo*** devolver esta fila.

Sin embargo, cuando son posibles diferentes planes de ejecución, los valores de los parámetros en la primera invocación pueden decidir qué se utilizará para todas las invocaciones posteriores. Consulte el listado de 2.

Si los valores de parámetro para las llamadas posteriores al procedimiento almacenado tienen valores similares (de hecho, cuando sus valores no cambian la selectividad de las consultas de procedimiento) el rastreo de parámetro es importante. De lo contrario podría reducir drásticamente rendimiento.

Ahora tenemos dos planes de ejecución razonable:

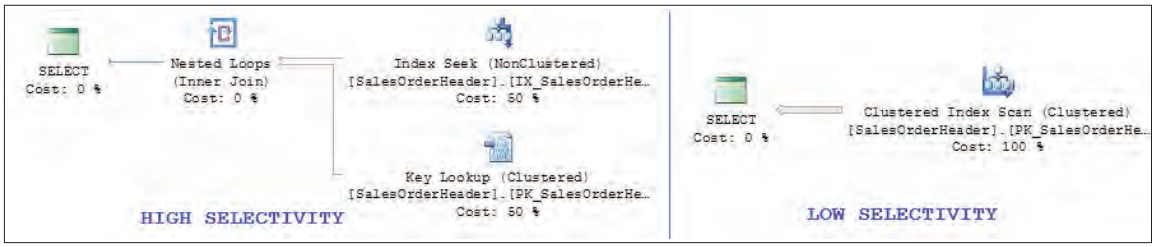

Figura 4: Plan de ejecución para el procedimiento almacenado **dbo.getSalesOrderHeader**

Qué se utilizará, es decidido por el valor del parámetro **OrderDate** para la primera invocación del procedimiento almacenado. Si el valor del parámetro es altamente selectivo (pocas filas en el conjunto de resultados) se generará el segundo plan, de lo contrario se analizará toda la tabla (análisis de índice agrupado). En este caso, no podemos ignorar cómo afecta el rastreo de parámetros. Algunas invocaciones al procedimiento almacenado terminan con un plan no óptimo. Y, por lo general, es un problema. Podría ser que el plan de ejecución sea óptimo para valores del parámetro altamente selectivos, vamos a decir, 95% de llamadas con el valor "óptimo" del parámetro. En este caso podemos decir que un 5% tiene un problema pero, esto, desde el punto de vista de la lógica comercial, no es tan importante para la aplicación. Sin embargo, el problema es que el plan de ejecución no permanece para siempre en la memoria caché. Podría ser eliminado (por muchas razones) y no tenemos control sobre cuando ocurre esto. Así, después de que el plan se elimina de la caché, es una lotería si la siguiente invocación viene con un valor de parámetro esperado.

El rastreo de parámetro no se limita a procedimientos almacenados sólo, sino que todos los parámetros de consultas puede ser susceptibles del rastreo de parámetros: consultas estáticas con parametrización simple, auto o forzada, o consultas dinámica con *sp_executesql*. En este artículo describiremos este fenómeno en los procedimientos almacenados.

¿Cuándo es un problema el rastreo de parámetros?

Hay dos tipos de procedimientos almacenados propensos a dar problemas con el rastreo de l parámetros:

- Procedimientos almacenados con parámetros que participan en los operadores de rango
- Procedimientos almacenados con parámetros opcionales

El procedimiento **dbo.getSalesOrderHeader** pertenece al primer grupo, y describimos en este artículo cómo resolver el problema del rastreo de parámetros en esos procedimientos almacenados. En el siguiente número de SolidQ analizaremos el problema de rastreo de parámetros en procedimientos almacenados con parámetros opcionales.

Por lo tanto, ya hemos visto que los dos planes de ejecución son posibles para este procedimiento almacenado. Si la primera invocación fue con el valor 01.07.2001 se devuelven todas las filas y de acuerdo con esto, ha sido elegido agrupado por el optimizador un análisis del índice como plan de ejecución óptima.

SQL Server debe realizar 1.406 lecturas lógicas para generar un conjunto de resultados. Sin embargo, cuando el procedimiento almacenado se ha invocado por primera vez con el valor 31.07.2005, una llamada con el parámetro 01.07.2001, ¡termina con

lecturas 94.456 lógicas! La figura 5 muestra el tiempo de ejecución para todas las combinaciones de valor del parámetro y los planes de ejecución apropiados:

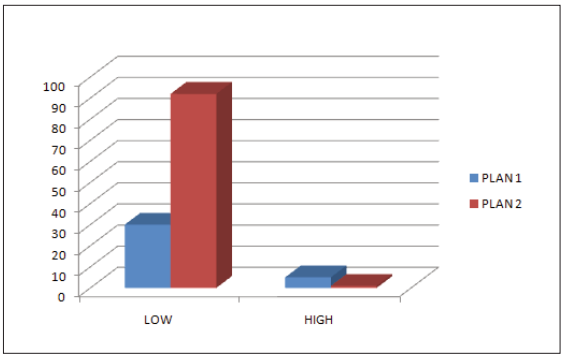

Figura 5: Tiempo de ejecución para valores de parámetros selectivos altos y bajos

Podemos ver que una llamada con un parámetro de selectividad baja se ejecuta tres veces mejor con su plan óptimo, mientras que una llamada con un parámetro de alta selectividad se ejecuta mejor con un plan de ejecución que incluya una búsqueda por índice. El problema que plantea el rastreo de parámetros pone en evidencia cuándo un procedimiento almacenado se ejecuta con un plan que no es óptimo.

Resolver el problema del rastreo de parámetros

Antes de empezar a resolver problemas de rastreo de parámetros, tenemos que definir ***lo que*** queremos hacer. A continuación veremos ***Cómo*** para lograrlo. Uno de los conceptos erróneos acerca de rastreo de parámetros es que la solución para el problema sea desactivar el efecto. Este efecto no siempre es malo: podría ser muy malo, pero al mismo tiempo es una característica y su desactivación resuelve algunos problemas, pero pueden degradar el rendimiento de consultas bien diseñadas.

Aunque el rastreo de parámetros es un problema de rendimiento, su solución debe incluir los requerimientos del negocio. Como se mencionó anteriormente, el proceso de optimización:

- Considera valores de parámetros en la primera invocación del procedimiento almacenado
- Usa un plan de ejecución en caché para todas las llamadas posteriores del procedimiento almacenado

Este es el comportamiento predeterminado del optimizador y si no estamos satisfechos con el resultado de la optimización tenemos que cambiar uno de estos factores. Podemos sugerir al optimizador (o forzarlo) a no a tener en cuenta los parámetros de entrada para la generación del plan o sugerir que no debe utilizar el plan generado para todas las llamadas posteriores. En el primer caso, definimos un plan de ejecución "genérico" o un plan que se genera sin conocimientos acerca de los valores de parámetro y el plan se utiliza para todas las invocaciones. O simplemente podemos forzar al optimizador a que utilice un plan de ejecución óptimo para todas las combinaciones de parámetros.

Así, antes de empezar a resolver el problema debemos definir el objetivo de la optimización, es decir, lo que significa realmente resolver el problema. En nuestro procedimiento almacenado, una solución sería utilizar un plan de ejecución genérico. Esto significa que todas las llamadas de procedimiento almacenado se ejecutarán con el plan de ejecución, basado en el índice agrupado. Esto significa que los valores del parámetro que causen una baja selectividad utilizarán el plan óptimo y los demás son elegidos como víctimas y siempre utilizarán un plan que no es óptimo. Si esto es aceptable desde la lógica del punto de vista comercial, hemos definido el objetivo: mediante un plan de ejecución óptima de selectividad baja.

Enfoque 1: Actuar sobre el parámetro de rastreo mediante un Plan para todas las combinaciones de parámetros

Al adoptar este enfoque indicamos al optimizador que use algún plan de ejecución específico (con la sustitución de parámetros de entrada) o que ignore los parámetros de entrada.

Sustitución de parámetros de entrada

Cuando tenemos una combinación de parámetros muy utilizada podemos forzar al optimizador a que utilice esa combinación, siempre que cree un plan de ejecución. En este caso, nos aseguramos de que esta combinación se realiza bien siempre, independientemente de la existencia del plan en la memoria caché. En nuestro caso, podemos decidir optimizar la ejecución de parámetros de baja selectividad y utilizar el plan con el índice agrupado (plan de azul en la figura 5). Esto significa que los valores de parámetro que causen baja selectividad utilizarán un plan óptimo y los demás son elegidos como víctimas y siempre se utilizará el plan óptimo. Esto es una decisión lógica de negocio y debe ser aceptable desde el punto de vista de la lógica de negocio. En este caso, el rastreo de parámetros no está deshabilitado; en su lugar nos referimos a los valores de parámetro de entrada. La sugerencia de consulta **OPTIMIZE** fuerza al optimizador a generar un plan de ejecución óptimo para el valor de la sugerencia de consulta en vez de para el valor enviado por la primera invocación.

```
OPTION (OPTIMIZE FOR (@OrderDate='20010101'))
```

Con la sugerencia de consulta sustituimos el valor del parámetro presentado y el rastreo de parámetros no está deshabilitado. El optimizador lo considera en el proceso de generación del plan; sólo cambiamos el valor del parámetro enviado. Se trata de una solución cuando queremos garantizar que la ejecución está bien para una combinación favorita de parámetros. Esto es útil en escenarios donde hemos identificado a la combinación de parámetros más importante, y forzamos una ejecución con el plan óptimo para esta combinación, mientras se ignoran los problemas de rendimiento con otras combinaciones. En la figura 6 podemos ver que esta solución asegura que el plan de selectividad baja se utilizará para todas las llamadas de procedimiento almacenado. Consulte la figura 6.

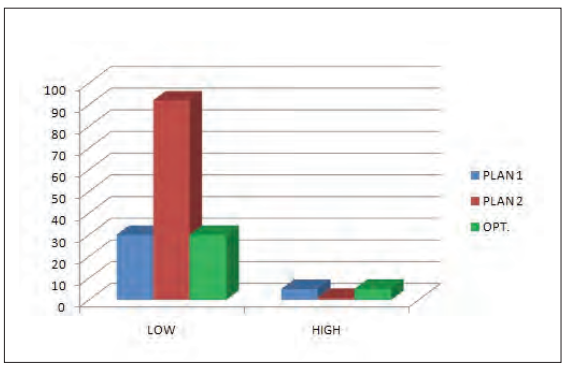

Figura 6: solución con un plan genérico de ejecución

Deshabilitar el rastreo de parámetros

Desactivación el rastreo de parámetros es útil cuando el procedimiento almacenado tiene varios parámetros y no hay una combinación favorita de parámetros. Si no se conocen los valores de parámetro en tiempo de compilación, no hay nada que se descubrir.

Así es cómo los excluimos de las consideraciones del plan de ejecución. El resultado de la desactivación del rastreo de parámetro es, generalmente, un plan de "promedio" – no óptimo, pero aceptable para todas las combinaciones de parámetros. Tenemos tres opciones para deshabilitar o neutralizar el rastreo de parámetros: sugerencias de consulta, reescritura de consultas y buscar trazas de la bandera 4136.

Con sugerencias de consulta

Para deshabilitar el rastreo de parámetros nuevamente podemos utilizar la sugerencia de consulta **OPTIMIZE**.

```
OPTION (OPTIMIZE FOR UNKNOWN)
```

Esta vez el optimizador recibe una sugerencia para omitir los valores de los parámetros enviados. Por lo tanto, se desconoce el valor y el optimizador utiliza la densidad de los valores de la tabla para estimar la cardinalidad. Con el operador de rango, eso significa que se espera aproximadamente un 30% de filas en el resultado, los que generalmente significa un plan de ejecución promedio con análisis

Listado 3: Modificación de procedimiento almacenado mediante el uso de Variables locales

```
CREATE PROCEDURE dbo.getSalesOrderHeader
@OrderDate datetime
AS
          DECLARE @OrderDateLocal AS datetime = @OrderDate
          SELECT SalesOrderID,OrderDate,ShipDate,SubTotal
          FROM Sales.SalesOrderHeader
          WHERE OrderDate >= @OrderDateLocal
```

de índice agrupado. Esta opción está disponible en SQL Server 2008.

Reescritura de consultas

El mismo efecto puede lograrse reescribiendo algo de código en el procedimiento almacenado. Consulte el listado 3.

Introducimos una variable local y le asignamos el valor del parámetro. Dado que el optimizador compila el procedimiento almacenado en un proceso por lotes, el valor del argumento en la cláusula **WHERE** no se conoce. Por lo tanto, tenemos el mismo plan que con la sugerencia **UNKNOWN**.

Bandera de seguimiento 4136

En octubre de 2010, como parte de la actualización acumulativa, Microsoft proporcionó una opción para deshabilitar el rastreo de parámetros a nivel de instancia (KB980653). Cuando la bandera de seguimiento 4136 está habilitada, el rastreo de parámetros está deshabilitado para toda la instancia. Esto tiene el mismo efecto que poner la sugerencia **OPTIMIZE FOR UNKNOWN** en todos los procedimientos almacenados y consultas enviadas al motor de base de datos. Esto suena muy restrictivo y peligroso y por lo tanto, todavía hay excepciones donde este indicador de traza no tiene ningún efecto: consulta que utilicen **OPTIMIZE FOR** o sugerencias de consulta **RECOMPILE** y procedimientos almacenados definidos con la opción **WITH RECOMPILE**.

Aunque esto aporta la flexibilidad necesaria y no rompe con nuestros esfuerzos de optimización ya implementados, el indicador de traza 4136 afecta a toda la instancia del servidor y debe considerarse como un último recurso para resolver problemas de rastreo de parámetros.

El enfoque con sugerencias de consulta tiene ventajas sobre la opción con variables locales porque las sugerencias de consulta no afectan a la lógica empresarial en un procedimiento almacenado y pueden aplicarse fuera del procedimiento. Combinando la sugerencia de consulta con otra gran característica de SQL Server: -las guías de planificación - podemos resolver problemas de rastreo de parámetros sin cambiar el código de la aplicación. Esto es muy importante para los sistemas en los que no se permiten cambios en el código o cuando un cambio de código pequeño es inaceptablemente caro (proyecto-recompilación, pruebas, implementación etc...).

Método 2: Un Plan óptimo para cada combinación de parámetro

Cuando se requiere que cada combinación debe tener un plan óptimo, la desactivación de rastreo de parámetros no nos facilita la tarea. Tenemos dos opciones para implementarlo: indicar al optimizador que genere (recompile) un nuevo plan para cada procedimiento almacenado o reescribir el procedimiento almacenado mediante el uso de otro procedimiento almacenado (procedimiento almacenado de árbol de decisión).

Recompilar el Plan de ejecución

Esto puede hacerse con la sugerencia de consulta **OPTION (RECOMPILE)** o mediante la inclusión

de la opción **WITH RECOMPILE** en la definición del procedimiento almacenado.

Listado 4: Procedimiento almacenado modificado cambiando la definición de procedimiento

```
CREATE PROCEDURE dbo.getSalesOrderHeader
@OrderDate datetime
WITH RECOMPILE
AS
          SELECT SalesOrderID,OrderDate,ShipDate,SubTotal
          FROM Sales.SalesOrderHeader WHERE OrderDate >= @OrderDate
```

Listado 5: Modificación de procedimiento almacenado mediante el uso de la sugerencia de consulta

```
CREATE PROCEDURE dbo.getSalesOrderHeader
@OrderDate datetime
AS
          SELECT SalesOrderID,OrderDate,ShipDate,SubTotal
          FROM Sales.SalesOrderHeader WHERE OrderDate >= @OrderDate
          OPTION (RECOMPILE)
```

En ambos casos se genera un nuevo plan para cada invocación de procedimiento. La diferencia es que **OPTION (RECOMPILE)** se realiza a nivel de instrucción y no se regeneran todas las sentencias del procedimiento almacenado, sino sólo las marcadas con esta opción. En nuestro caso es lo mismo, tenemos sólo una declaración. En el próximo número de SolidQ veremos las ventajas de compilar a nivel de instrucción. La figura 7 muestra que el plan de recompilar combina los planes iniciales para parámetros selectivos bajos y altos y es óptimo para todos los parámetros.

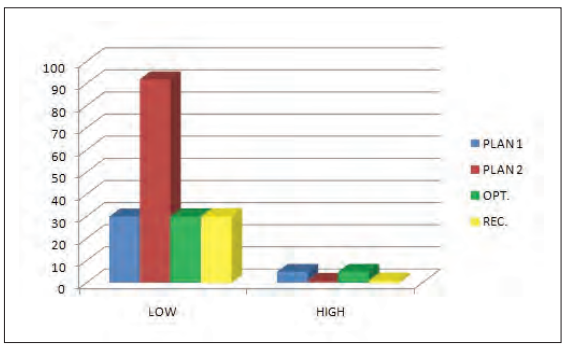

Figura 7: Solución con un plan óptimo para cada combinación de parámetros

Árbol de decisiones del procedimiento almacenado

Un procedimiento almacenado de árbol de decisiones *decide* que sub-procedimiento debería llamar basándose en los parámetros enviados. Vamos a mostrar esta solución. Consulte el listado de 6, página 51.

El procedimiento determina si la fecha enviada es de más de 10 días atrás. Si es así, llama a un procedimiento almacenado para parámetros de selectividad baja, de lo contrario se selecciona la versión para fechas con alta selectividad. Los sub-procedimientos tienen el mismo cuerpo de procedimiento y son idénticos al procedimiento almacenado inicial **dbo.GetSalesOrder-Header**. La figura 8 muestra los planes de ejecución para parámetros de alta y baja selectividad.

Esta solución permite la reutilización de planes de ejecución y desde un punto de vista del rendimiento es mejor que la versión utilizando la opción

Listado 6: Modificar procedimientos almacenados mediante un procedimiento almacenado de árbol de decisión

```
CREATE PROCEDURE dbo.getSalesOrderHeaderDT
@OrderDate datetime
AS
        IF @OrderDate > DATEADD(day,-10,CURRENT_TIMESTAMP)
                EXEC dbo.getSalesOrderHeaderHighSelectivity @OrderDate
        ELSE
                EXEC dbo.getSalesOrderHeaderLowSelectivity @OrderDate
```

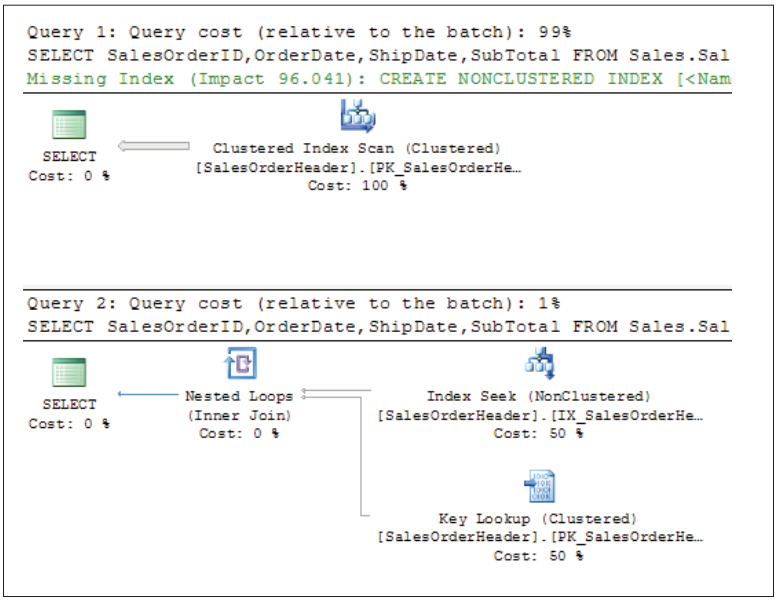

Figura 8: Tiempo de ejecución para parámetros selectivos altos y bajos para el procedimiento almacenado de árbol de decisión.

binaciones. Aún así, el número de procedimientos para cubrir los planes más comunes puede hacerse rápidamente inmanejable.

En este artículo hemos visto lo que es el rastreo de parámetro y cuándo y por qué resulta un problema. Hemos hablado de diferentes soluciones para problemas de rastreo de parámetros con procedimientos almacenados y del uso de operadores de rango. El mes que viene veremos por qué los procedimientos almacenados con parámetros opcionales son propensos al problema del rastreo de parámetros y ofreceremos nuevamente varias soluciones. ■

recompile. Sin embargo, este enfoque está abierto a problemas de mantenimiento. Para cada combinación donde queremos reutilizar un plan existente necesitaríamos un sub-procedimiento. Por lo tanto, la solución sería inmanejable. Otro problema de mantenimiento es que las decisiones sobre lo que es altamente selectivo no son tan fáciles y deberían ser evaluadas y comprobadas regularmente para ver si siguen siendo apropiadas. Podemos combinar ambos enfoques creando suficiente sub-procedimientos para cubrir los planes más comunes y uno con la opción **recompile** para todas las demás com-

Sobre al Autor

Milos Radivojevic *es Arquitecto de Plataformas de Datos con SolidQ CEE, ubicada en Viena, Austria. Su principal foco es el desarrollo de bases de datos y la optimización del rendimiento en sistemas OLTP.*