Optimización de consultas a través de índices

Conceptos básicos sobre Índices

Un índice en SQL Server mejora la velocidad de recuperación de datos en las consultas, funcionando como un "índice" en un libro para localizar información rápidamente.

Los índices son estructuras de datos que permiten acceder a los datos de forma más rápida al ordenar físicamente (índices agrupados) o mediante referencias organizadas (índices no agrupados)

Los índices más comunes son:

- **Índice Clustered** (agrupado): ordena físicamente los datos en la tabla. Solo puede haber uno por tabla, ya que define el orden físico de los datos.
- **Índice Non-Clustered** (no agrupado): crea una estructura separada que apunta a las filas de datos. Puede haber múltiples índices non-clustered en una tabla.

Para mejorar el rendimiento con índices, se deben identificar las consultas que involucran operaciones frecuentes de búsqueda y filtrado (`WHERE`, `JOIN`); y que **Ordenan o agrupan** datos frecuentemente (`ORDER BY`, `GROUP BY`). También, que seleccionan una **porción de registros** en lugar de hacer un escaneo completo de la tabla.

SQL Server proporciona herramientas como el **"Execution Plan"** para revisar cómo se ejecuta una consulta y ver qué índice se utilizó para llevarla a cabo

En este informe, solo se trabajó con los **Índices Agrupados** (Clustered).

Aplicación en SQL Server

A continuación, aplicaremos los índices a tablas relevantes de nuestra base de datos:

-Ejemplo en la Tabla `Ventas`

La tabla *Ventas* es una tabla donde seguro se harán muchas consultas de búsqueda.

Estas búsquedas podrían realizarse por columnas como FechaRegistro o MontoPago. Un índice en esos campos va a acelerar estas búsquedas.

TAREAS SOLICITADAS:

- ➢ Realizar una carga masiva de por lo menos un millón de registro sobre alguna tabla que contenga un campo fecha (sin índice). Hacerlo con un script para poder automatizarlo.

```
CREATE TABLE VentaNuevo (
    id_Venta INT PRIMARY KEY,
    MontoPago INT,
    MontoTotal INT,
    FechaRegistro DATE,
    Id_Usuario INT,
    Id_Cliente INT
);
```

Se creo una tabla nueva en base a la tabla de *Ventas.*

Luego, cargamos un millon de registros en ella con un script.

➢ *Realizar una búsqueda por periodo y registrar el plan de ejecución utilizado por el motor y los tiempos de respuesta.*

se ejecuta una consulta que busque por un período de fechas, por ejemplo, un periodo de un año:

```
69
70  SET STATISTICS TIME ON;
71  SET STATISTICS IO ON;
72
73  SELECT *
74  FROM VentaNuevo
75  WHERE FechaRegistro BETWEEN '2015-01-01' AND '2015-12-31';
76
77  SET STATISTICS TIME OFF;
78  SET STATISTICS IO OFF;
79
```

los comandos *SET STATISTICS TIME ON* y *SET STATISTICS IO ON* permiten medir y analizar el rendimiento de las consultas, en términos de tiempo y de operaciones de entrada/salida.

-Obtuvimos los siguientes resultados al realizar la consulta sin ningun indice:

```
 SQL Server Execution Times:
   CPU time = 110 ms,  elapsed time = 668 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2024-11-10T17:52:30.2381644-03:00
```

Lecturas lógicas: 4951

Se puede ver que la consulta sin ningún índice mostró un tiempo de ejecución relativamente alto y número elevado de lecturas lógicas. Esto se debe a que SQL Server tuvo que realizar una búsqueda completa en la tabla, revisando cada fila para cumplir con los criterios de búsqueda.

Un número alto de lecturas lógicas sugiere que SQL Server tuvo que leer muchas páginas de datos en memoria para obtener el resultado. Esto puede indicar que la consulta no está optimizada, ya que sevestá accediendo a más datos de los necesarios.

En la siguiente imagen se puede ver que el programa no usó ningún índice para realizar la consulta.

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	1000000
Actual Number of Rows for All Executions	100140
Actual Number of Batches	0
Estimated I/O Cost	3.66987
Estimated Operator Cost	4.76995 (100%)
Estimated Subtree Cost	4.76995
Estimated CPU Cost	1.10008
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	104564
Estimated Number of Rows Per Execution	104564
Estimated Number of Rows to be Read	1000000
Estimated Row Size	30 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0

➢ *Definir un índice agrupado sobre la columna fecha y repetir la consulta anterior. Registrar el plan de ejecución utilizado por el motor y los tiempos de respuesta.*

Declaramos el **Índice Agrupado** (Clustered) en el Campo de `FechaRegistro` para mejorar el rendimiento de la consulta:

```
85
86  CREATE CLUSTERED INDEX IDX_VentaNuevo_FechaRegistro
87  ON VentaNuevo(FechaRegistro);
88
```

Se vuelve a ejecutar la consulta de búsqueda por un periodo de un año. y se obtiene estos resultados:

```
 SQL Server Execution Times:
   CPU time = 31 ms,  elapsed time = 575 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2024-11-10T18:05:50.4392483-03:00
```

Lecturas lógicas: 501

Al aplicar un índice agrupado en la columna `FechaRegistro`, el rendimiento mejoró de forma notable. La reducción en las lecturas lógicas (de 4951 a 501) muestra que SQL Server pudo localizar de manera más eficiente los registros dentro del rango de fechas especificado, sin necesidad de escanear toda la tabla.

En la siguiente imagen se puede ver el uso del índice agrupado para efectuar la consulta:

Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	100140
Actual Number of Rows for All Executions	100140
Actual Number of Batches	0
Estimated Operator Cost	0.48089 (100%)
Estimated I/O Cost	0.370532
Estimated Subtree Cost	0.48089
Estimated CPU Cost	0.110357
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	100182
Estimated Number of Rows to be Read	100182
Estimated Number of Rows Per Execution	100182
Estimated Row Size	30 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

➢ *Definir otro índice agrupado sobre la columna fecha pero que además incluya las columnas seleccionadas y repetir la consulta anterior. Registrar el plan de ejecución utilizado por el motor y los tiempos de respuesta.*

Primero, eliminamos el índice anterior para evaluar el nuevo *índice agrupado*.

```
90
91  DROP INDEX IDX_VentaNuevo_FechaRegistro ON VentaNuevo;
92
```

Ahora creamos de un **Índice Agrupado** sobre la misma columna `FechaRegistro,` que ahora incluye otras columnas en la consulta para mejorar la optimización en accesos específicos:

```
98
99  CREATE CLUSTERED INDEX [IDX_VentaNuevo_FechaRegistro]
100     ON [dbo].[VentaNuevo] ([FechaRegistro],[MontoPago], [MontoTotal])
101
```

Ejecutamos de nuevo la consulta de búsqueda por un periodo de un año, y obtenemos estos resultados:

```
 SQL Server Execution Times:
   CPU time = 94 ms,  elapsed time = 555 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.

Completion time: 2024-11-12T19:55:00.2518647-03:00
```

Lecturas lógicas: 401

Al añadir las columnas `MontoPago` y `MontoTotal` al índice agrupado, el rendimiento mostró una pequeña mejora en el tiempo de ejecución y las lecturas lógicas. Lo que hizo la inclusión de estas columnas es que SQL Server pudo satisfacer la consulta utilizando únicamente el índice, ya que se evita hacer búsquedas adicionales en la tabla para recuperar estos datos.

En la siguiente imagen se puede ver el uso del índice agrupado con las columnas agregadas para efectuar la consulta:

Clustered Index Seek (Clustered)

Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows Read	100140
Actual Number of Rows for All Executions	100140
Actual Number of Batches	0
Estimated Operator Cost	0.406815 (100%)
Estimated I/O Cost	0.296458
Estimated Subtree Cost	0.406815
Estimated CPU Cost	0.110357
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows for All Executions	100182
Estimated Number of Rows to be Read	100182
Estimated Number of Rows Per Execution	100182
Estimated Row Size	30 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

➢ *Expresar las conclusiones en base a las pruebas realizadas.*

En conclusión, el uso de índices agrupados en la columna `FechaRegistro`, y en particular la inclusión de las columnas necesarias en el índice, optimizó significativamente las consultas en términos de tiempo de respuesta y eficiencia de lecturas.

La estructura del índice agrupado ordena físicamente los datos según `FechaRegistro`, esto hizo que las búsquedas de rangos sean mucho más eficientes en comparación con una consulta sin índices. Tambien, el incluir las columnas seleccionadas en el índice agrupado evitó que se hagan accesos innecesarios a la tabla, lo que mejoró más el rendimiento de la consulta y redujo las lecturas lógicas.