Optimización de consultas a través de índices

Un índice es una estructura en disco o en memoria asociada con una tabla o vista que acelera la recuperación de filas de la tabla o vista. Un índice de almacén de columnas contiene claves generadas a partir de una o varias columnas de la tabla o la vista. En el caso de los índices de almacén de columnas, dichas claves están almacenadas en una estructura de árbol (árbol B+) que permite que el motor de base de datos busque de forma rápida y eficiente la fila o las filas asociadas a los valores de cada clave.

El optimizador de consultas en el motor de base de datos elige de forma confiable el índice más eficaz en la mayoría de los casos. La estrategia general de diseño de índices debe proporcionar una variedad de índices al optimizador de consultas y confiar en que tomará la decisión correcta. Así se reduce el tiempo de análisis y se obtiene un buen rendimiento en diversas situaciones.

Tipos de índices

Índice	Descripción
Clustered	Un índice clúster ordena y almacena las filas de datos de la tabla o vista por orden en función de la clave del índice clúster. El índice clúster se implementa como una estructura de árbol b
Nonclustered	Cada fila del índice no clúster contiene un valor de clave no agrupada y un localizador de fila. Este localizador apunta a la fila de datos del índice clúster o el monton que contiene el valor de clave. Las filas del índice se almacenan en el mismo orden que los valores de la clave del índice, pero no se garantiza que las filas de datos estén en un determinado orden a menos que se cree un índice clúster en la tabla.
Hash	En un dice hash, se accede a los datos a través de una tabla hash en memoria. Los índices hash utilizan una cantidad fija de memoria, que es una función del número de cubos.
Unique	Un índice único se asegura de que la clave de índice no contenga valores duplicados y, por tanto, cada fila de la tabla o vista sea en cierta forma única.
columnstore	El índice de almacén de columnas en memoria almacena y administra los datos mediante el almacenamiento de datos basado en columnas y el procesamiento de consultas basado en columnas.
Índice con columnas incluidas	Índice no clúster que se extiende para incluir columnas sin clave además de las columnas de clave.
Índice en columnas calculadas	Índice de una columna que se deriva del valor de una o varias columnas, o algunas entradas deterministas.

Filtrado	Índice no clúster optimizado, especialmente indicado para cubrir consultas que seleccionan de un subconjunto bien definido de datos. Utiliza un predicado de filtro para indizar una parte de las filas de la tabla.
Espacial	Un índice espacial permite realizar de forma más eficaz determinadas operaciones en objetos espaciales (*datos espaciales*) en una columna del tipo de datos de **geometry** . El índice espacial reduce el número de objetos a los que es necesario aplicar las operaciones espaciales, que son relativamente costosas.

Etapa evaluativa

- Comprender las características de la propia base de datos.

  Si se usa un gran número de índices en una tabla, el rendimiento de las instrucciones INSERT, UPDATE, DELETE y MERGE se verá afectado, ya que todos los índices deben ajustarse adecuadamente a medida que cambian los datos de la tabla.

  Por lo tanto, si se usa un gran número de índices en una tabla, el rendimiento de las instrucciones INSERT, UPDATE, DELETE y MERGE se verá afectado, ya que todos los índices deben ajustarse adecuadamente a medida que cambian los datos de la tabla.

  La indización de tablas pequeñas puede no ser una solución óptima, porque puede provocar que el optimizador de consultas tarde más tiempo en realizar la búsqueda de los datos a través del índice que en realizar un sencillo recorrido de tabla

- Determinar las consultas mas frecuentes y sus características
  La utilización de índices puede mejorar el rendimiento de las consultas, ya que los datos necesarios para satisfacer las necesidades de la consulta existen en el propio índice. Es decir, solo se requieren las páginas de índice, y no las páginas de datos de la tabla o el índice clúster, para recuperar los datos solicitados; por lo tanto, se reduce la E/S de disco global.

- Entender las características de las columnas utilizadas en las consultas, tipo de datos y tipo de valores que acepten

  Las columnas de los tipos de datos ntext, text, image, varchar(max), nvarchar(max)y varbinary(max) no pueden especificarse como columnas de clave de índice. En cambio, los tipos de datos varchar(max), nvarchar(max), varbinary(max)y xml pueden participar en un índice no agrupado como columnas de índice sin clave.

Sintaxis

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]
INDEX index_name ON <object> ( column [ ASC | DESC ] [ ,...n ] )
[ INCLUDE ( column_name [ ,...n ] ) ]
[ WHERE <filter_predicate> ]
[ WITH ( <relational_index_option> [ ,...n ] ) ]
```

Argumentos

- UNIQUE: se declara si el índice es único.
- CLUSTERED | NONCLUSTERED: se indica si es ordenado o no.
- Index_name: se declara el nombre representativo del índice
- column: se indica en que columna se va crear el índice
- ASC | DES: se especifica el tipo de ordenamiento.
- Incluede: especifica las columnas que no son de clave que se agregarán en el nivel hoja del índice no clúster.
- WHERE: crea un filtrado que se va a incluir en el índice.

Ensayo

- 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.
- 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

Consideraciones:

La tabla que se intervino fue Inspecciones

El lote de dato se duplico mediante un script.

4.636.000 de registros fueron los involucrados.

Script para la creación del índice:

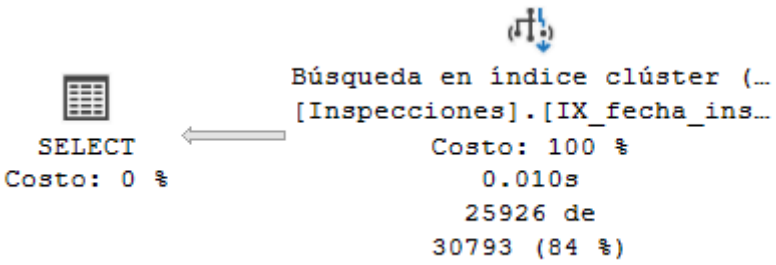
```
CREATE CLUSTERED INDEX [IX_fecha_inspeccion]
ON [dbo].[Inspecciones] ([fecha_inspeccion] ASC)
```

Consulta

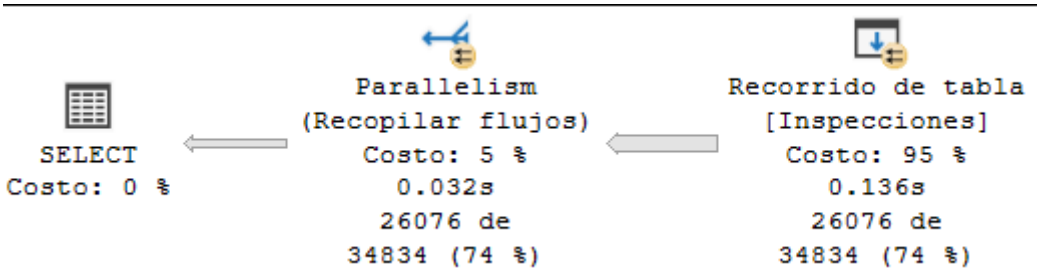
```
SELECT fecha_inspeccion AS 'Fecha inspeccion' , estado_inspeccion AS 'ESTADO'
FROM Inspecciones
WHERE estado_inspeccion = 'Aprobada'
  AND fecha_inspeccion BETWEEN '2023-03-01' AND '2023-05-31';
```

Resultados

En este primer caso, se utiliza la búsqueda en un índice clúster. Como resultado, se obtiene un costo relativo del 33% en comparación con el segundo caso, donde no se le proporciona un índice para la columna **[fecha_inscripciones]** al motor de base de datos. En este escenario, el motor debe recurrir a un **Index Scan**, lo que implica que tiene que recorrer toda la tabla para encontrar los registros correspondientes.



[CASO 1 = Consulta con Indice Clustered en la columna fecha_inscripciones]



[CASO 2 = Consulta con Indice Clustered en la columna id_inspecciones]

Estadística				
CASO	Costo	Lecturas lógicas	Tiempo transcurrido	costo de subárbol estimado
1	1%	217	0,010s	00,22
2	99%	33083	0,0136s	27,20

[4.6 millón de registros como lote de prueba]

Se propone agregar la columna [estado_inscripcion] al índice creado anteriormente. Se procede a eliminar el índice existente, solo se puede generar un índice clustered por tabla.

```
DROP INDEX IX_fecha_inspeccion ON Inspecciones
```

[Script para eliminar el índice creado anteriormente]

Script para la creación del nuevo índice Clustered con la columna [estado_inscripcion]:

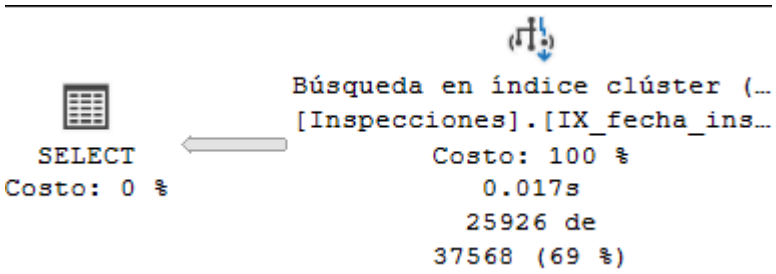
```
CREATE CLUSTERED INDEX IX_fecha_inspeccion
ON [dbo].[Inspecciones]([fecha_inspeccion],[estado_inspeccion])
```

[CASO 3.1 Ordenado por fecha]

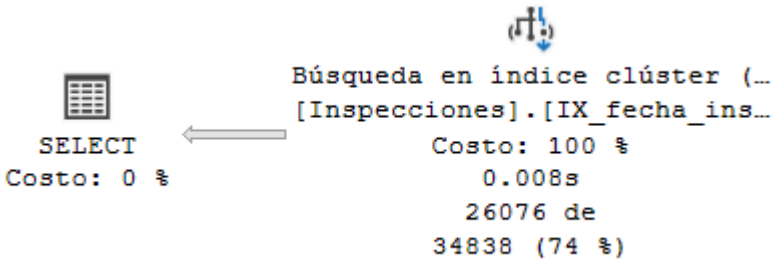
```
CREATE CLUSTERED INDEX IX_fecha_inspeccion
ON [dbo].[Inspecciones]([estado_inspeccion],[fecha_inspeccion])
```

[CASO 3.2 Ordenado por [estado_inspeccion]]

En este caso, nos preguntamos cúal es la mejor manera de ordenar el índice. Ordenar primero por fecha y luego por la columna [estado_inscripcion] o viceversa. En estos, casos siempre se va a premiar a la columna que tenga mayor selectividad, es decir, la que tenga menor variaciones de valores.



[CASO 3.1 Índice ordenado primero por [fecha_inspeccion]]



[CASO 3.2 Indeice ordenado primero por [id_inspecciones]]

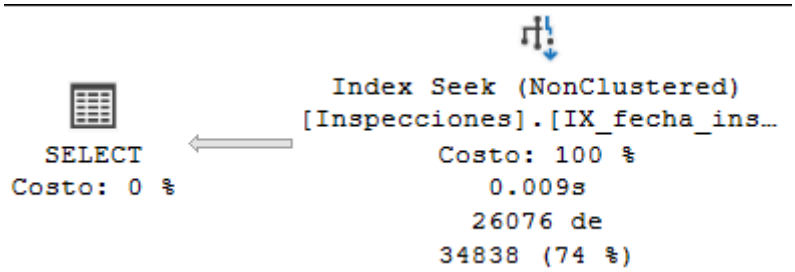
Estadística				
CASO	Costo [relativo]	Lecturas lógicas	Tiempo transcurrido	costo de subárbol estimado
3.1	60%	308	0,017 ms	0,25
3.2	40%	152	0,008 s	0,17

[4.6 millón de registros como lote de prueba]

El caso 3.2 mostró un mejor rendimiento debido a la baja cardinalidad de la columna **[id_inspecciones]**, lo que significa que la columna contenía una menor variedad de valores. Esto facilitó una búsqueda más eficiente. En contraste, la columna **[fecha_inspeccion]** presentaba una menor selectividad, es decir, una mayor variedad de valores similares, lo que dificultó la optimización de la consulta y generó un mayor costo de ejecución.

Índice NO CLUSTERED

```
CREATE NONCLUSTERED INDEX [IX_fecha_inspeccionNonclustered]
ON [dbo].[Inspecciones] ([estado_inspeccion],[fecha_inspeccion])
```



[CASO 4]

Podemos observar un buen rendimiento del índice creado para la consulta analizada. Como resultado, la duración total de la consulta fue de **0.009 segundos**, y se realizaron **97 operaciones lógicas** durante su ejecución

Uso de Memoria

Almacenamiento		
CASO	Espacio de datos	Espacio para Índice
1	210,938 MB	00,903 MB
2.1 [HEAP]	210,938 MB	00,001 MB
3.1	210,938 MB	00,920 MB
3.2	210,938 MB	00,950 MB
4	210,938 MB	137,305 MB

En el caso 4, donde se utilizó un índice **NonClustered**, se observó un aumento considerable en el espacio requerido para el índice. Aunque la consulta mejoró en términos de rendimiento, el espacio de almacenamiento utilizado por el índice casi igualó al de los propios datos. Este es un factor importante a considerar al optimizar el rendimiento de las consultas: es necesario evaluar si vale la pena invertir en más espacio físico para obtener una mejora en el tiempo de ejecución de las consultas, o si es más conveniente explorar otras estrategias que puedan reducir los costos económicos asociados con el almacenamiento.

Conclusión:

Los índices son una herramienta fundamental para mejorar el rendimiento de las consultas, ya que permiten un acceso más rápido a los datos. Sin embargo, su implementación también conlleva ciertos costos, como el uso adicional de CPU durante las actualizaciones de los índices y un aumento en el espacio de almacenamiento necesario. Por lo tanto, el uso adecuado de **índices agrupados** (Clustered) y **no agrupados** (NonClustered) dependerá de diversos factores, como el patrón de consultas, la estructura de los datos y los requisitos específicos de rendimiento. Un índice agrupado, por ejemplo, es ideal cuando las consultas requieren un acceso eficiente a rangos de datos ordenados, mientras que los índices no agrupados pueden ser más apropiados para consultas que involucren filtros sobre columnas específicas. Es crucial encontrar un balance entre el beneficio en el tiempo de respuesta y los costos asociados al almacenamiento y al uso de recursos, adaptando la estrategia de indexación a las necesidades particulares de la base de datos y sus operaciones.