10 consejos prácticos sobre índices en SQL Server

Vp

Los índices impactan directamente en el desempeño de la base de datos -para bien o para mal- ya que facilitan la extracción de información, aunque añaden una sobrecarga al momento de la actualización de los datos.

La pertinencia o necesidad de crearlos dependerá de los requerimientos de las aplicaciones, pero en términos generales existen algunos con consejos prácticos sobre el uso de índices en SQL Server (aunque muchos aplican para cualquier base de datos relacional):

1. Eliminar índices duplicados.
2. Evitar índices redundantes.
3. Evitar que las tablas tengan más índices que columnas.
4. Evitar tablas que no tengan un índice Cluster.
5. Que todas las tablas tengan al menos un índice.
6. Incluir índices en campos que son llaves foráneas.
7. Mantener pequeñas las llaves de los índices.
8. Mantener actualizadas las estadísticas.
9. Contener la fragmentación de los índices.
10. Eliminar los índices hipotéticos.

1.- Eliminar índices duplicados

Un índice está duplicado cuando existe otro que tiene los mismos campos en el mismo orden, y deberá ser eliminado uno de ellos. Es equivalente a que en un libro exista el mismo índice de temas más de una vez, en realidad solo se debe tener uno de ellos y el otro -u otros- se pueden eliminar.

En una tabla llamada Alumnos, por citar un ejemplo, podríamos haber creado los siguientes índices

```
CREATE INDEX indice1 ON Alumnos (matricula);
CREATE INDEX indice2 ON Alumnos (nombre,apellidos);
CREATE INDEX indice3 ON Alumnos (matricula,nombre);
CREATE INDEX indice4 ON Alumnos (matricula);
```

Queda claro que **tanto el indice1 como el indice4 son iguales** , por lo que se puede eliminar uno de ellos y al momento de hacer un INSERT, UPDATE o DELETE que pudiera afectar a los índices ya no tendrá que actualizar 4 índices, ya que con 3 se tiene la misma funcionalidad y rendimiento.

2.- Evitar índices redundantes

Los índices redundantes son aquellos que comparten algunos campos llave en el mismo orden y que uno puede contener a otro sin problema, es decir, la llave de un índice es subconjunto de la llave de otro índice.

Siguiendo con los mismos índices de la tabla *Alumnos* usado en el punto anterior, podemos aseverar que el *indice1* es redundante (que está formado por matricula) ya que existe otro, el *indice3*, que está formado por matrícula más nombre, por lo que cualquier query que intente buscar por matrícula podría hacer uso del *indice3*.

Sin tomar en consideración otros elementos (índices cluster, fill factor, estadísticas, filegroups, etc.) los índices 1 y 4 se pueden eliminar por que son redundantes, y la tabla se quedaría tan solo con 2 índices (el de nombre+apellidos y el de matricula+nombre).

3.- Evitar que las tablas tengan más índices que columnas

Una mala práctica en el modelado de bases de datos (haciendo un razonamiento equivocado de que entre más índices cualquier consulta será más rápida), es tener un índice por cada campo en la tabla (además de los índices con llave compuesta por más de un campo).

Esto provoca que existan tablas que tengan más índices que campos. Esto debe evitarse, ya que en la práctica esto nunca resulta en un mejor rendimiento de la base de datos, si no por el contrario.

4.- Evitar tablas no tengan un índice Cluster

En SQL Server existe un tipo de índice especial que permite ordenar los datos en el disco en el orden de la llave primaria. Esto hace las búsqueda de la llave de dicho índice más eficientes (así como los ordenamientos en base a la llave), y en general es una buena práctica tener un índice cluster para cada tabla (solo es posible tener un índice cluster por tabla).

5.- Que todas las tablas tengan al menos un índice

Ya sea por considerar que todas las tablas cumplan con las formas normales (específicamente la 1FN) o por temas de *performance*, es importante que todas las tablas tengan por lo menos un índice, es decir, al menos deberían tener el que se asocia a la llave primaria. Esto hará que todas las búsquedas más comunes a la tabla (por su llave primaria) utilicen ese índice para mejorar el rendimiento de las consultas, además de que asegurará la unicidad de registros en las tablas.

6.- Incluir índices en campos que son llaves foránas

Si bien una consulta muy común en las tablas es por su llave primaria, otras que son ampliamente ocupadas son aquellas que involucran las uniones o joins con tablas con las que se tiene alguna relación de 1 a muchos. Un ejemplo, es en una tabla de alumnos donde es bastante común el hacer join's con las tablas de entidades, estatus, sexo, carrera, etc.; es decir, con todas aquellas tablas con las cuales la tabla de alumnos tiene una relación de 1 a muchos.

En este caso se recomienda ampliamente el generar índices por las llaves foráneas para agilizar todos los join's que de manera normal se utilizarán en la base de datos.

7.- Mantener pequeñas las llaves de los índices

El tamaño de las llaves es un elemento importante al momento de armar una llave ya que, dependiendo de la base de datos, puede ser que se esté sobrepasando el límite permitido. En el caso de SQL Server este límite es de 900 bytes (1700 bytes para SQL Server 2016). Esto quiere decir que la suma de los campos que conforman la llave no debe sobrepasar este límite. Si la llave está formada por un solo campo CHAR, entonces la llave no debe sobrepasar los 900 caracteres.

Una llave de 900 bytes es una llave muy grande, lo ideal es formar llaves pequeñas (como de los puntos anteriores, con las llaves primarias o con las llaves foráneas que suelen ser pequeñas si se hace caso a las recomendaciones en la conformación de las llaves primarias).

7.- Mantener pequeñas las llaves de los índices

El tamaño de las llaves es un elemento importante al momento de armar una llave ya que, dependiendo de la base de datos, puede ser que se esté sobrepasando el límite permitido. En el caso de SQL Server este límite es de 900 bytes (1700 bytes para SQL Server 2016). Esto quiere decir que la suma de los campos que conforman la llave no debe sobrepasar este límite. Si la llave está formada por un solo campo CHAR, entonces la llave no debe sobrepasar los 900 caracteres.

Una llave de 900 bytes es una llave muy grande, lo ideal es formar llaves pequeñas (como de los puntos anteriores, con las llaves primarias o con las llaves foráneas que suelen ser pequeñas si se hace caso a las recomendaciones en la conformación de las llaves primarias).

8.- Mantener actualizadas las estadísticas

Conforme se van actualizando los datos, las estadísticas de las columnas, si no se actualizan, dejan de dar información adecuada para calcular la mejor forma de procesar un query. Son información sobre la distribución de los datos existentes en las columnas índice (si varia mucho, si los datos son iguales, los niveles de variación) que ayudan a determinar el mejor plan de ejecución). Es por esto que es recomendable tener actualizadas las estadísticas

9.- Contener o limitar la fragmentación de los índices

Un índice fragmentado es aquel que no está optimizado y que ya no es tan eficiente como uno recién creado. Si se piensa que un índice es como un árbol binario (como se expuso al explicar lo que es un índice) un índice fragmentado podría ser aquel que está desbalanceado y con muchos nodos vacíos, por lo que es necesario crear un plan de mantenimiento para que se incluya la regeneración de índices continuamente y así evitar la fragmentación.

10.- Eliminar los índices hipotéticos

Ocasionalmente el Database Engine Tuning Advisor (DETA) no borra índices temporales que llama hipotéticos que sirven para medir la mejora de sus recomendaciones.

Si el índice hipotético no es utilizado, entonces se debe eliminar.

Consejo Extra
11.- Usar índices filtrados cuando sea posible

Un índice filtrado es aquel que **no incluye todos los registros** en el índice, solo aquellos registros que tienen una característica relevantes. Por ejemplo, cuando se tiene una tabla con un campo Sexo (con dos posibles valores H y M) y la gran mayoría de los registros son de mujeres, un índice por Sexo no sería útil para búsquedas de mujeres, pero si en el caso de hombres (el subconjunto obtenido será menor). Un índice filtrado permite filtrar solo los registros de Sexo=H, de tal manera que si se ejecuta una consulta buscando mujeres no utilizará el índice, pero si se buscan hombres entonces si sería útil.


Gracias ...!
Víctor Portugal