Consejos para el diseño de bases de datos de SQL Server

El diseño de bases de datos es una fase que muchos programadores toman a la ligera. Un sistema puede funcionar, pero sin eficiencia y con la posibilidad siempre presente de la duplicidad innecesaria e involuntaria en información.

Si pertenece al grupo que utiliza el Primer enfoque de base de datos, cualquier decisión que tome en fase puede tener un gran impacto.

Interior de un Data Center

Me gustaría destacar algunos puntos importantes que pueden ser dignos de considerar para un mejor diseño y rendimiento de la base de datos.

- Siempre trate de usar una convención de nomenclatura y algunos estándares como el tamaño del campo, los tipos

de datos y los supuestos generales antes de comenzar a diseñar la base de datos. Estos pueden ahorrar mucho tiempo y esfuerzos. Estos incluso pueden desempeñar un papel vital en el rendimiento.

- Considere la desnormalización, particularmente para datos muy grandes. Pero como regla general, primero normalice su base de datos. Es una buena idea usar la Vista indizada para la desmoralización. La **desnormalización** es el proceso de procurar optimizar el funcionamiento de una base de datos por medio de agregar datos redundantes. A veces es necesario porque los actuales sistemas de gestión de bases de datos (SGBD) implementan el modelo relacional de manera limitada. Un SGBD relacional que cumpla las recomendaciones ISO debe permitir una base de datos completamente normalizada a nivel lógico, soportado por un almacenamiento físico de los datos afinado para los requisitos necesarios del sistema.
- Un diseño normalizado a menudo almacenará diferentes, pero relacionadas, piezas de información en tablas lógicas separadas (denominada relaciones). Si estas relaciones están almacenadas físicamente en dos archivos distintos en disco, su recuperación para realizar una consulta a la base de datos que se base en información de varias relaciones (una operación unión) puede ser muy lenta, llegando a no satisfacer determinados requisitos no funcionales del sistema. Hay dos estrategias para tratar de solventar esta situación:
    - El método preferido es mantener normalizado el diseño lógico, pero indicar al SGBD que almacene en el disco información redundante para optimizar la respuesta a la consulta. En este caso, es responsabilidad del software del SGBD proveer soporte para gestionar la confidencialidad de los datos en todos los niveles de la arquitectura. Este método es

implementado de diversa manera según el desarrollador: en Microsoft SQL Server se usan vistas indexadas, en lote productos de Oracle vistas materializadas, etc. Una vista representa la información en un formato conveniente para consultar, y el índice asegura que las consultas contra la vista estén optimizadas.

- Sin embargo, la aproximación más usual es desnormalizar el diseño de datos lógico. Realizado con cuidado, esto puede alcanzar una mejora similar en respuesta de consulta, pero complica la tarea de los usuarios que modifiquen contenido del sistema: ahora es la responsabilidad del diseñador de la base de datos el asegurarse de que la base de datos desnormalizada no llegue a ser inconsistente. Esto se realiza mediante reglas en la base de datos llamadas restricciones, que especifican cómo las copias redundantes de información se deben mantener sincronizadas. Es el aumento en la complejidad lógica del diseño de la base de datos y la complejidad añadida de las restricciones adicionales lo que hacen de esta una alternativa delicada. Por otra parte, debido a los gastos indirectos de evaluación de restricciones al insertar, actualizar, o eliminar datos, una base de datos desnormalizada puede llegar a ofrecer en la práctica un rendimiento inferior al que proporcionaba su versión equivalente normalizada. Por contra, cuando se está seleccionado o leyendo datos a menudo el funcionamiento es más probable que sí se mejore.

- Será mejor definir las relaciones de clave principal y clave externa.
- Finalice claramente el nivel de aislamiento de la base

de datos.

- Defina claramente la secuencia de participación de las tablas en la transacción para evitar puntos muertos.
- Es una buena práctica tener una columna de identidad en una tabla, ya sea clave principal o no.
- Planifique con precisión la indexación y tenga en cuenta que es un proceso continuo. Sea específico para elegir la columna para índices agrupados y no agrupados. En general, el uso de columnas en join, where, order by y group by afecta directamente sus requisitos de índice.
- Necesita un buen plan para la fragmentación del índice, especialmente cuando hay muchas operaciones de inserción, eliminación y actualización.
- Considere las nuevas tecnologías y la solución lista para usar. En general, es posible que prefiera las funciones listas para usar en lugar de usar metodologías personalizadas antiguas hasta que, y a menos que exista alguna razón específica para hacerlo, como la compatibilidad con versiones anteriores.
    - Considere la función integrada de Particionamiento de tablas en lugar de la partición manual de tablas para tablas de datos grandes.
    - Considere File Stream introducido en SQL Server 2008 para File Storage.
    - Considere nuevos DataTypes sobre viejos siempre que sea posible, como Varchar (max) sobre Text, Date over DateTime y etc.
    - Considere las tablas optimizadas en memoria. Ahora la RAM es muy barata, y tales técnicas de caché de datos pueden hacer una gran diferencia en el rendimiento.
- Considere dividir la base de datos en varios archivos según el tamaño de su base de datos.
- Considere la separación de archivos de base de datos y archivos de registro en diferentes discos duros.

- Siempre use el tipo de datos apropiado, por ejemplo, si tiene que almacenar datos en inglés solo entonces no necesita usar nvarchar, o nchar, si necesita almacenar un pequeño conjunto de valores, entonces no es necesario usar bigint.
- Considere la tabla de resumen o los datos calculados previamente para datos muy grandes.
- Planifique una estrategia clara para el rendimiento y el mantenimiento. En muchos casos, debe abordar el rendimiento frente al mantenimiento. Idealmente, debe elegir una combinación aceptable de ambos, pero esta decisión depende directamente de sus necesidades y presupuesto.
- Será mejor mantener el mismo estándar para los objetos de la base de datos (tabla y columna) y el modelo de datos de la aplicación. Aunque puede no proporcionar ninguna ganancia de rendimiento, pero puede facilitar a los desarrolladores.
- Si planea utilizar disparadores o cursores, reevalúe los enfoques alternativos.