Motores de Almacenamiento y Manejo de Indices en MySQL

Brenda Mariza Quintero Beltrán,

Resumen—MySQL es un sistema de administracion de base de datos relacional. Lógicamente, los datos se estructuran en tablas que se relacionan entre si por un campo comun. MySQL incorpora una característica única llamada ”motores de almacenamiento”, que nos permite seleccionar el tipo de almacenamiento interno de cada tabla, en base al que mejor se adecue a una situación particular, entre los más populares se encuentran MyISAM e InnoDB. MySQL utiliza índices para la optimización de consultas, los cuales facilitan a MySQL recuperar eficientemente los datos de una tabla.

Palabras Clave—Motor de Almacenamiento, índices, espacio en disco.

1 INTRODUCCIÓN

Esta investigación, no se concentrará en los aspectos técnicos de los diferentes motores de almacenamiento, sino que se concentrará sobre cómo y dónde estos diferentes motores pueden ser empleados de la mejor manera. También se incluye un análisis detallado de la estructura y utilización de los índices y sus beneficios.

A continuación se describe brevemente lo que es un motor de almacenamiento y la indexación:

El servidor MySQL incorpora una característica única llamada ”motores de almacenamiento”, que nos permite seleccionar el tipo de almacenamiento interno de cada tabla, en base al que mejor se adecue a una situación particular. Dicha selección, la hace el desarrollador a nivel de tabla, y no afecta a la manera en que el servidor interactúa con el cliente: los comandos SQL serán los mismos sea cual sea el motor de almacenamiento escogido. El cliente no necesita saber como se guardan los datos.

El motor de almacenamiento es quien almacenará, manejará y recuperará información de una tabla en particular. Comparando MyISAM vs InnoDB, ninguno se destaca como la solución para la mayoría de los casos. Cada uno tiene sus pros y sus contras, por lo tanto al momento de decidir que motor de almacenamiento a utilizar dependerá mucho del escenario donde se aplique.

En la optimizacion de consultas se hace uso del indexado, la indexación puede aumentar drásticamente el rendimiento al buscar y recuperar datos de la base de datos. Los diferentes motores de almacenamiento ofrecen diferentes técnicas de indización y algunos pueden ser más adecuados para el tipo de datos que se encuentren.

• *B.M. Quintero Belrán es con la Escuela de Informática, Universidad Autónoma de Sinaloa, Josefa Ortiz de S/N, Ciudad Universitaria, Culiacán, Sinaloa, 8000, México.*
E-mail: lmarizita@hotmail.com

2 MOTORES DE ALMACENAMIENTO

2.1 ¿Que es un Motor de Almacenamiento?

Los datos en MySQL se almacena en archivos o memoria usando una variedad de diferentes técnicas. Cada una de estas técnicas emplean diferentes mecanismos de almacenamiento, la indexación facilita, los niveles de bloqueo y, en definitiva, proporciona una gama de diferentes funciones y capacidades. Al elegir una técnica diferente podemos ganar más velocidad o funcionalidad de los beneficios que mejoren la funcionalidad de nuestra aplicación.

Por ejemplo, si trabajamos con una gran cantidad de datos temporales, podemos hacer uso del motor de almacenamiento MEMORY, que almacena todos los datos de las tablas en la memoria. Alternativamente, podemos querer una base de datos que soporte transacciones para garantizar la capacidad de datos.

Cada una de estas diferentes técnicas y suites de funcionalidad en el sistema de MySQL que se denomina un motor de almacenamiento también conocido como tipo de tabla. Por defecto, MySQL viene con una serie de diferentes motores de almacenamiento preconfigurado y habilitado en el servidor MySQL.

Podemos seleccionar el motor de almacenamiento a utilizar en un servidor, base de datos e incluso la tabla base, proporcionándonos mayor flexibilidad a la hora de elegir la forma en que la información se almacenará, cómo se indexa y qué combinación de rendimiento y funcionalidad deseamos utilizar con los datos.

Esta flexibilidad de elegir la forma en que los datos son almacenados y la indexación es una de las principales razones por las qué MySQL es tan popular; otros sistemas de bases de datos, incluidas la mayoría de las opciones comerciales, soportan un único tipo de almacenamiento de base de datos. Lamentablemente, el ”tamaño de un enfoque único para todos” en estas otras soluciones significa un sacrificio en el rendimiento de la funcionalidad, o tener que pasar horas o incluso días de ajuste adecuado para la base de datos. Con MySQL, podemos cambiar el motor que estamos utilizando.

2.2 Diferenciando los Motores

La mayoría de las personas que utilizan MySQL saben que MyISAM e InnoDB son los dos motores de almacenamientos más comunes en MySQL. También es sabido, que la mayoría no toma en cuenta el motor de almacenamiento al crear una tabla y acepta el que viene por default en la base de datos.

Con el fin de tomar una decisión acerca de cual motor elegir, primero tenemos que pensar en las diferentes funcionalidades básicas en cada uno de los motores que nos permitan diferenciarlos entre ellos. En general podemos dividir la funcionalidad básica en cuatro partes, el apoyo sobre campos y tipos de datos, tipos de bloqueo, indización y transacciones.

- Campos y Tipos de Datos: Aunque todas los motores soportan los tipos comunes datos, es decir, enteros, reales y caracteres, no todos los motores apoyan otros tipos de campo, en particular la BLOB ”grandes objetos binarios” o tipos de texto. Otros motores pueden soportar anchuta de carácter y tamaños de datos limitado.
  Mientras estas limitaciones, afectan directamente a la información que tiende también a tener un efecto relacionado con los tipos de búsquedas que se realizan, o sobre los índices que se creen sobre esa información. A su vez, estas diferencias pueden afectar el rendimiento y la funcionalidad de la aplicación, es por ello que se pueden tomar decisiones a cerca de la funcionalidad del motor de almacenamiento que se elija para el tipo de datos que se esté almacenando.
- El Bloqueo: El bloqueo dentro de las bases de datos, define como es controlado el acceso y la actualización de información. Cuando un objeto en la base de datos está bloqueado para la actualización, otros procesos no pueden modificar o en algunos casos leer los datos hasta que la actualización ha terminado.
- Indexación: la indexación puede aumentar drásticamente el rendimiento al buscar y recuperar datos de la base de datos. Los diferentes motores de almacenamiento ofrecen diferentes técnicas de indización y

algunos pueden ser más adecuados para el tipo de datos que se encuentren.
Algunos motores de almacenamiento simplemente no soportan la indexación, ya sea porque todos los que utilizan la indexación de las tablas fundamentales o porque el método de almacenamiento de datos no permite la indexación.

- Transacciones: Las transacciones proporcionan fiabilidad de los datos durante la actualización o inserción de la información permitiendo añadir datos a la base de datos, pero sólo depositan estos datos cuando las condiciones y otras etapas en la ejecución de la aplicación han finalizado con éxito. Por ejemplo, cuando la transferencia de información de una cuenta a otra deberíamos usar transacciones para asegurarnos de que tanto la domiciliación bancaria de una cuenta y el crédito se ha completado correctamente. Si bien el proceso ha fallado, podríamos cancelar la transacción y los cambios se perderían. Si el proceso es completado, entonces lo confirmaríamos depositando los cambios.

2.3 Tipos de Motores de MySQL

MySQL dispone de una docena de motores de almacenamiento propios, más los motores externos desarrollados por terceras partes que se pueden incorporar al servidor. Algunos de los más conocidos son: MyISAM, InnoDB, HEAP, NDB. En éste capítulo se le dará mayor énfasis a los motores MyISAM e InnoDB, los cuales trataremos más adelante.

A continuación se describe brevemente cada uno de los motores de almacenamiento, más adelante se tratarán con más detalle los motores MyISAM e InnoDB:

- MyISAM trata tablas no transaccionales. Proporciona almacenamiento y recuperación de datos rápida, así como posibilidad de búsquedas fulltext. MyISAM se soporta en todas las configuraciones MySQL, y es el motor de almacenamiento por defecto a no ser que tenga una configuración distinta a la que viene por defecto con MySQL.
- El motor de almacenamiento MEMORY anteriormente conocido como el montón motor de almacenamiento almacena todos los datos en memoria, una vez que se ha decidido apagar el servidor MySQL, cualquier información almacenada en una base de datos MEMORY se han perdido. Sin embargo, el formato de las tablas individuales se mantiene y esto le permite crear tablas temporales que se pueden utilizar para almacenar información para un acceso rápido sin tener que recrear las tablas cada vez que el servidor de base de datos es iniciado.
A largo plazo, el uso del motor MEMORY en general, no una buena idea, porque los datos podrían ser tan perdidos fácilmente. Sin embargo, se tiene la RAM para soportar las bases de datos en las que se están trabajando, el uso de tablas MEMORY es una forma eficiente de correr consultas complejas sobre grandes conjuntos de datos y beneficiándonos de las ganancias del rendimiento.
- El motor de almacenamiento MERGE permite una colección de tablas MyISAM idénticas ser tratadas como una simple tabla. Como MyISAM, los motores de almacenamiento MEMORY y MERGE tratan tablas no transaccionales y ambos se incluyen en MySQL por defecto. Podemos ejecutar consultas que devuelven los resultados de múltiples tablas como si se tratara de una sola tabla. Cada tabla fusionada debe tener la misma definición de tabla.
Las tablas MERGE son particularmente efectivas si estamos logging directorios de datos, directa o indirectamente, en una base de datos MySQL y creando una tabla individual por día, semana o mes y queriendo ser capaces de producir consultas agregadas de múltiples tablas. Hay limitaciones para esto sin embargo, podemos fusionar tablas MyISAM y la restricción de definición de idéntica de tabla es estrictamente forzada.
- El motor de almacenamiento ISAM es el motor original disponible en las versiones de MySQL hasta que se introdujo el motor de almacenamiento MyISAM en MySQL.

ISAM tiene un número de diferentes limitaciones que lo hacen poco práctico como un motor de base de datos. Estos incluyen el formato de almacenamiento, que es nativo de la plataforma y por lo tanto, no entre sistemas portátiles, un tamaño máximo de tabla de 4GB y limitado a búsquedas de sólo texto. Los índices son también son más limitados, desde que MyISAM es compatible con las mismas plataformas que ISAM, y ofrece una mejor compatibilidad, portabilidad y rendimiento.

- El motor EXAMPLE es en realidad un ejemplo de programación de un motor de almacenamiento que puede ser utilizado como base para otros motores en el sistema de MySQL. No soporta las inserciones de datos y no es un motor práctico para cualquier forma de acceso a bases de datos. Sin embargo, es una buena guía de cómo desarrollar su propio motor de almacenamiento y, por tanto, una eficaz guía para los programadores.
- NDB Cluster es el motor de almacenamiento usado por MySQL Cluster para implementar tablas que se particionan en varias máquinas. Está disponible en distribuciones binarias MySQLMax 5.0. Este motor de almacenamiento está disponible para Linux, Solaris, y Mac OS X.
- El motor de almacenamiento ARCHIVE se usa para guardar grandes cantidades de datos sin índices con una huella muy pequeña.
- El motor de almacenamiento CSV guarda datos en ficheros de texto usando formato de valores separados por comas.
- El motor de almacenamiento FEDERATED se añadió en MySQL 5.0.3. Este motor guarda datos en una base de datos remota. En esta versión sólo funciona con MySQL a través de la API MySQL C Client.

2.4 El Motor MyISAM

MyISAM es el motor de almacenamiento por defecto. Se basa en el código ISAM pero tiene muchas extensiones útiles. Cada tabla MyISAM se almacena en disco en tres ficheros. Los ficheros tienen nombres que comienzan con el nombre de tabla y tienen una extensión para indicar el tipo de fichero. Un fichero .frm almacena la definición de tabla. El fichero de datos tiene una extensión .MYD. El fichero índice tiene una extensión .MYI MYIndex.

El motor MyISAM proporciona la mejor combinación de rendimiento y funcionalidad, a pesar de que carece de la capacidad de transacción en cambio utiliza el nivel de tabla de bloqueo.

A menos que necesitemos transacciones, hay pocas bases de datos y aplicaciones que no pueden ser almacenados de manera efectiva usando el motor MyISAM. Sin embargo, en aplicaciones de alto rendimiento en las que hay un gran número de inserciones o actualizaciones de datos en comparación con el número de lecturas puede provocar problemas para el motor MyISAM. Originalmente fue diseñado con la idea de que más del 90% de los accesos de de las bases de datos una tabla MyISAM serían para lectura, más bien que para escritura.

Con el nivel de tabla de bloqueo, una base de datos con un elevado número de fila las inserciones o actualizaciones se convierten en un cuello de botella de rendimiento como la tabla está bloqueada mientras que los datos se añaden. Afortunadamente, esta limitación también funciona bien dentro de las restricciones de las bases de datos no-transaccionales.

Algunas características del motor de almacenamiento MyISAM son:

- MyISAM es el motor de almacenamiento por default y está basado en el probado ISAM, incorporando nuevas características pero conservando su fiabilidad.
- MyISAM almacena la información en tres archivos por tabla, uno para el formato de tabla, otro para los datos y un tercer archivo para los índices.
- Los campos Text y Blob pueden ser indexados completamente, lo que es de gran importancia para funciones de búsqueda.
- No transaccional.
- Bloqueos a nivel de tabla.
- Muy rápido en lectura y escritura.
- Bajo requerimiento de espacio en disco y memoria.
- Los datos se guardan en disco: diferentes

ficheros para la definición de la tabla, los datos y los índices.
- Es una buena elección cuando necesitamos velocidad, y tenemos pocas modificaciones simultáneas de la tabla.
- Ficheros grandes se soportan en sistemas de ficheros y sistemas operativos que soportan ficheros grandes.
- Registros de tamaño dinámico se fragmentan mucho menos cuando se mezclan borrados con actualizaciones e inserciones. Esto se hace combinando automáticamente bloques borrados adyacentes y extendiendo bloques si el siguiente bloque se borra.
- La longitud máxima de clave es 1000 bytes. Esto puede cambiarse recompilando. En caso de clave mayor a 250 bytes, se usa un tamaño de bloque mayor, de 1024 bytes.
- Las columnas BLOB y TEXT pueden indexarse.
- Valores NULL se permiten en columnas indexadas. Esto ocupa 0-1 bytes por clave.

2.5 El Motor InnoDB

InnoDB se diseñó para obtener el máximo rendimiento al procesar grandes volúmenes de datos. Probablemente ningún otro motor de bases de datos relacionales en disco iguale su eficiencia en el uso de CPU.

A pesar de estar totalmente integrado con el servidor MySQL, el motor de almacenamiento InnoDB mantiene su propio pool de almacenamiento intermedio para tener un caché de datos e índices en la memoria principal. InnoDB almacena sus tablas e índices en un espacio de tablas, el cual puede consistir de varios ficheros. Las tablas InnoDB pueden ser de cualquier tamaño, aún en sistemas operativos donde el tamaño de los ficheros se limita a 2GB.

La clave para el sistema InnoDB es una base de datos, almacenando en caché e indexando la estructura donde ambos índices y los datos son cacheados en la memoria, así como para ser almacenados en el disco. Esto permite la recuperación muy rápidamente, y funciona incluso sobre conjuntos de datos muy grandes.

Mediante el soporte a nivel de bloqueo de fila, se pueden agregar datos a una tabla InnoDB sin el motor bloqueando la tabla con cada inserción y esto acelera la recuperación y el almacenamiento de información en la base de datos. Al igual que con MyISAM, hay pocos tipos de datos que no puede ser almacenados en una base de datos InnoDB. De hecho, no hay razones por las cuales no se debe utilizar siempre una base de datos InnoDB.

La gestión de los costos de InnoDB es un ligeramente más de molesto, y obtener la optimización correcta de los espacios de memoria, en disco, en los caches y archivos de base de datos puede ser compleja, en primer lugar. Sin embargo, también significa que se puede obtener mayor flexibilidad en estos valores, los beneficios superan a los primeros en tiempo. Alternativamente, se puede dejar que MySQL gestione esto automáticamente.

Características de InnoDB:
- Transaccional.
- Multiversionado: cuando múltiples transacciones modifican registros, InnoDB mantiene aisladas las transacciones guardando para cada una de ellas un versión distinta de un mismo registro, a cada transacción la versión que le corresponde.
- Bloqueos a nivel de registro.
- Restricciones en claves foráneas.
- Fácil recuperación de datos en caso de error.
- Alta concurrencia más segura en escritura.
- Deshacer transacciones a medias ”rollback”.
- Los datos se guardan en disco: un fichero para la definición de la tabla, y un ”tablespace” para guardar conjuntamente datos e índices. El tablespace puede consistir en uno o más ficheros, o incluso una partición entera en disco.
- Necesita más espacio en disco y memoria que MyISAM para guardar los datos.
- Es una buena elección cuando necesitamos transacciones, restricciones en claves foráneas, o tenemos muchas escrituras simultáneas. de InnoDB es el soporte de transacciones e integridad referencial.
- InnoDB provee bloqueo a nivel final, en contra del bloqueo a nivel tabla de My-

ISAM. Esto es, que mientras una consulta está actualizando o insertando una fila, otra consulta puede actualizar una fila diferente al mismo tiempo. Estas características incrementan la performance en concurrencia de múltiples usuarios.
- Otra de las principales características es que permite definir Foreign Key Constraints, lo que permite a los desarrolladores asegurarse que los datos insertados con referencia a otra tabla permanecerán válidos.

2.5.1 Estructura de Tablas e Indices

Cada tabla InnoDB tiene un índice especial llamado índice agrupado llamado clustered index donde se almacenan los datos de las filas. Si se define una PRIMARY KEY en una tabla, el índice de la clave primaria es el índice agrupado.

Si no se define una PRIMARY KEY para la tabla, MySQL toma como clave primaria el primer índice UNIQUE que tenga solamente columnas NOT NULL, al cual InnoDB utiliza como índice agrupado.

Si no hay en la tabla un índice con esas características, InnoDB generará internamente un índice agrupado donde las filas estarán ordenadas por el identificador de fila row ID que InnoDB asigna a las columnas en tal tabla. El identificador de fila es un campo de 6 bytes que se incrementa automáticamente a medida que se agregan nuevas filas. Por lo tanto, las filas ordenadas por este identificador están en el orden físico de inserción.

El acceso a una fila a través del índice agrupado es rápido porque la fila de datos se encuentra en la misma página a donde el índice dirige su búsqueda. Si una tabla es grande, la arquitectura del índice agrupado a menudo ahorra operaciones de E/S en disco en comparación a la solución tradicional. En muchos servidores de bases de datos, los datos se suelen almacenar en una página diferente que la entrada del índice.

En InnoDB, las entradas en índices no agrupados, también llamados índices secundarios, contienen el valor de clave primaria de la fila. InnoDB utiliza este valor de clave primaria para buscar la fila a partir del índice agrupado. Nótese que si la clave primaria es larga, los índices secundarios utilizan más espacio.

InnoDB compara las cadenas CHAR y VARCHAR de diferente longitud como si el espacio sobrante en cadena la más corta estuviera relleno con espacios.

Todos los índices en InnoDB son árboles binarios "B-trees" donde las entradas de índice se almacenan en páginas que son las hojas del árbol. El tamaño predeterminado de una página de índice es 16KB. Cuando se insertan nuevas entradas, InnoDB intenta reservar 1/16 de la página para futuras inserciones y actualizaciones en el índice.

Si las entradas del índice se insertan en un orden secuencial, las páginas de índice se llenarán en aproximadamente 15/16 de su capacidad. Si las entradas se insertan en un orden aleatorio, las páginas se llenarán entre 1/2 y 15/16 de su capacidad. Si la proporción de espacio ocupado de una página de índice cae por debajo de 1/2, InnoDB intentará reducir el árbol de índice para liberar la página. Si una tabla cabe casi completamente en la memoria principal, la manera más rápida de ejecutar consultas sobre ella es empleando índices hash. InnoDB posee un mecanismo automático que supervisa las búsquedas realizadas sobre los índices de una tabla. Si InnoDB advierte que las consultas se podrían beneficiar con la creación de un índice hash, lo hará automáticamente.

El índice hash siempre está basado en un índice B-tree existente en la tabla. InnoDB puede generar un índice hash sobre un prefijo de la clave definida para el B-tree de cualquier longitud, dependiendo del patrón de búsquedas que InnoDB observe en el índice B-tree. Un índice hash puede ser parcial: no se necesita que el índice B-tree completo sea alojado en el pool de buffer. InnoDB genera índices hash según sea necesario basándose en las páginas de índice que son utilizadas más frecuentemente.

Podría decirse que, a través del mecanismo de índices hash adaptativos, InnoDB se adapta a la memoria principal, acercándose así a la arquitectura de las bases de datos de memoria.

2.5.2 Estructura Física de los Registros

Los registros de las tablas InnoDB tienen las siguientes características:

- Cada registro de índice en InnoDB contiene un encabezado de seis bytes. El encabezado se emplea para enlazar juntos registros consecutivos, y también en el bloqueo a nivel de fila.
- Los registros en el índice agrupado contienen campos para todas las columnas definidas por el usuario. Adicionalmente, hay un campo de seis bytes para el IDentificador de transacción y un campo de siete bytes para el roll pointer.
- Si no se definió una clave primaria para la tabla, cada registro de índice agrupado contiene también un campo de IDentificación de fila de seis bytes.
- Cada registro de índice secundario contiene también todos los campos definidos para la clave del índice agrupado.
- Un registro contiene además un puntero a cada campo del mismo. Si la longitud total de los campos en un registro es menos de 128 bytes, el puntero medirá un byte, de lo contrario, tendrá dos bytes de longitud. La matriz de estos punteros se conoce como el directorio de registros. El área a donde señalan estos punteros se denomina la parte de datos del registro.
- Internamente, InnoDB almacena las columnas de caracteres de longitud fija. InnoDB trunca los espacios sobrantes de las columnas VARCHAR. Nótese que MySQL puede convertir internamente columnas CHAR a VARCHAR.
- Un valor NULL SQL reserva 1 o 2 bytes en el directorio de registros. No reservará ningún byte en la parte de datos del registro si se lo almacena en una columna de longitud variable. En una columna de longitud fija, reservará en la parte de datos la longitud asignada a dicha columna. La razón por la que se reserva este espacio fijo a pesar de tratarse de un valor NULL, es que en el futuro se podrá insertar en su lugar un valor no-NULL sin provocar la fragmentación de la página de índice.

2.6 MyISAM vs InnoDB

Las principales ventajas de InnoDB sobre MyISAM son el soporte de transacciones, restricciones foráneas y bloqueo a nivel de fila. Nos permite pues tener las características ACID, garantizando la integridad de nuestras tablas y por tanto simplificando las recuperaciones ante posibles caídas del sistema.

Además, al permitir bloqueo a nivel de registro en lugar del bloqueo de tablas que implementa MyISAM, aumenta el rendimiento si nuestra aplicación hace un uso intensivo de sentencias INSERT o UPDATE.

Por contra, las ventajas de MyISAM pasan por una mayor velocidad a la hora de recuperar datos, es decir, mayor rendimiento ante mayoría de sentencias SELECT y por permitir crear índices sobre campos de tipo text. Este mayor rendimiento se produce gracias al no comprobar la integridad referencial ni bloquear las tablas a la hora de realizar las operaciones, aunque con ello se prescinda de la atomicidad. Para aplicaciones web que suelen tener más consultas que operaciones de escritura puede ser más que suficiente.

La elección pasa, como siempre, por conseguir la mejor relación calidad / precio. Si necesitamos transacciones, claves foráneas, bloqueo a nivel de fila o cualquier característica ACID, lo mejor será InnoDB. Si por el contrario no necesitamos nada de esto, o nuestra base de datos va a ser utilizada mayoritariamente para consulta, optaríamos por MyISAM.

MyISAM, en la mayoría de los casos será más rápido que InnoDB en selecciones, actualizaciones e inserciones bajo circunstancias normales. InnoDB también es un motor de almacenamiento ágil, pero se destaca porque incorpora características como bloqueo a nivel filas, transacciones y diseño de tablas relacionales. Aunque la primera de las características nombradas solo se destaca en tablas que son ”martilladas” constantes, como por ejemplo una tabla de logs, para el resto de los casos, un bloqueo a nivel tabla es suficiente en condiciones normales.

InnoDB se recupera de errores o reinicios no esperados del sistema a partir de sus logs, mientras que MyISAM requiere una explo-

ración, reparación y reconstrucción de índices de los datos de las tablas que aun no habían sido volcadas a disco.

2.7 Aplicaciones en las que un motor es más eficiente que otro

- Es aconsejable utilizar MyISAM cuando hay problema en el espacio en disco o memoria RAM.
- InnoDB es mejor si preferimos o requerimos diseño relacional de bases de datos.
- Si necesitamos hacer búsquedas full-text, es mejor utilizar MyISAM Cuando las tablas va a recibir INSERTs, UPDATEs y DELETES mucho más tiempo de lo que será consultada. Es mejor utilizar InnoDB.

3 ADMINISTRACION DEL ESPACIO EN DISCO Y MEMORIA

MySQL siempre crea un fichero .frm para guardar la definición de tabla y columnas. El índice y datos de la tabla pueden estar almacenado en uno o más ficheros, en función del tipo de tabla. El servidor crea el fichero .frm por encima del nivel de almacenamiento del motor. Los motores de almacenamiento individuales crean los ficheros adicionales necesarios para las tablas que administran.

El servidor usa el directorio de datos para almacenar lo siguiente:

Los directorios de base de datos. Cada base de datos corresponde al directorio único debajo el directorio de datos, sin considerar qué tipos de tablas usted cree en la base de datos. Por ejemplo, una base de datos determinada es representada por un directorio si contiene tablas MyISAM , tablas InnoDB, o una mezcla del dos.

Las tablas de formato archivo .frm. Cada tabla tiene su propio archivo .frm, ubicado en el directorio apropiado de base de datos. La verdad es que no importa que motor de almacenamiento administra la tabla.

Los archivos de índice y datos se crean para cada tabla por algunos motores de almacenamiento. Estos archivos se ponen en el directorio apropiado de base de datos. Por ejemplo, el motor de almacenamiento MyISAM crea un archivo de datos y un archivo índice para cada tabla. El motor de almacenamiento BDB crea un archivo para la tabla que incluye ambos, datos e índice. El motor de almacenamiento InnoDB tiene sus propios espacios de tablas y archivos log. Los espacios de tablas contienen información de datos e índice para todas las tablas InnoDB, así como también de los registros diarios que se necesitan si una transacción debe ser recuparada.

4 ESTRUCTURA Y USO DE LOS INDICES EN MySQL

El indexado es la herramienta más importante que tenemos para acelera las consultas. Existen otras técnicas disponibles, pero generalmente una de las cosas que hace la mayor diferencia es el uso apropiado de los índices. En la lista electrónica de MySQL, la gente a menudo solicita ayuda para hacer que una consulta corra más rápido. En un número sorprendentemente grande de los casos, no hay índice sobre las tablas en cuestión, y agregando un índice a menudo resuelve el problema inmediatamente. No siempre es así ya que la optimización no siempre sencilla. No obstante, si no usamos un índice, en muchos casos se pierde tiempo tratando de mejorar el rendimiento por otros medios. Usamos índices primero para conseguir un mayor rendimiento y después vemos qué otras técnicas podrían ser útiles.

Los índices son estructuras de datos que ayudan a MySQL a recuperar datos eficientemente. Los índices (también llamados "llaves" en MySQL) llegan a ser tan importantes como tan grandes sean los datos. Las bases de datos pequeñas ligeramente cargadas, frecuentemente se desempeñan bien aun sin índices apropiados, pero como la base de datos crece, su desempeño puede disminuir rápidamente.

La manera más fácil de comprender como trabaja un índice en MySQL es pensar sobre el índice de un libro. Para encontrar algún tema particular en un libro, usted mira en el índice, y este indica el número de la página donde aparece el término.

MySQL usa los índices de forma similar, busca la estructura del índice de un deter-

minado valor, cuando encuentra coincidencias regresa la fila donde se encuentra el dato.

Un índice contiene valores de una columna específica o varias columnas en una tabla, cuando indexamos una columna en particular, MySQL crea otra estructura de datos que usa para almacenar información acerca de los valores en la columna indexada, a éstos valores se le denominan claves o llaves, MySQL almacena todas las claves del índice en una estructura de datos de árbol, la cual le permite a MySQL encontrar claves muy rápidamente.

Los detalles particulares de las implementaciones de índice varían en los diferentes motores de almacenamiento de MySQL. Por ejemplo, para una tabla MyISAM, las filas de datos de las tablas se guardan en archivos de datos, y los valores de índice se guardan en un archivo de índice. Podemos tener más de un de índice en una tabla, pero todos son almacenados en el mismo archivo de índice. Cada índice en el archivo de índice consiste de un arreglo ordenado de registro claves que se usan para el acceso rápido en el archivo de datos.

Por el contrario, los motores de almacenamiento InnoDB y BDB no separan las filas de datos y los valores de índice del mismo modo, aunque ambos mantienen los índice como conjuntos de valores ordenados., el motor BDB de motor usa un archivo único por tabla para almacenar ambos datos y valores de índices. El motor InnoDB usa un solo espacio de tabla al menos en el que administra el almacenamiento de índice y datos para todas las tablas InnoDB. InnoDB puede configurarse para crear cada tabla con su propio espacio de tabla, pero aún así, los datos de una tabla y los índices se almacenan en el mismo archivo de espacio de tabla.

La discusión anterior describe el beneficio de un índice dentro del contexto de tabla única, donde el uso de un índices aceleran, donde el uso de un índice aceleran las búsquedas significativamente sin tener la necesidad de escanear completamente las tablas. Los índices son aun más valorados cuando emitimos consultas involucrando uniones en múltiples tablas. En una consulta de tabla única, el número de valores que necesitamos examinar por columna es el número de filas en la tabla. En consultas de múltiples tablas, el número de combinaciones posibles se eleva porque es el producto del número de filas en las tablas.

4.1 Tipos de Indices

Existen varios tipos de índices, cada uno diseñado para desempeñar propósitos diferentes. Los índice se implementan en la capa de motor de almacenamiento, pero cada uno de ellos trabaja de manera diferente en cada motor, y no todos los motores apoyan todos los tipos de índice. Veamos los tipos de índices que MySQL actualmente apoya, sus beneficios, y sus desventajas.

B-Tree: Es un índice con estructura de árbol, donde las entradas de índice se almacenan en páginas que son las hojas del árbol, es muy utilizado por su flexibilidad, rendimiento y ahorro de espacio. Además de agilizar las consultas directas por los campos indexados, también agilizan las consultas por rango BETWEEN.

MySQL utiliza índices B-Tree en los motores MyISAM, HEAP e InnoDB. En el caso de InnoDB, el índice está ”incrustado”, es decir, que se almacena junto a los datos y no en un fichero separado como en MyISAM.

Los índices de estructura B-TREE aceleran el acceso a los datos porque el motor de almacenamiento no tiene que recorrer completamente la tabla para encontrar los datos deseados. En vez de ello, comienza en el nodo raíz.

Hash: Si una tabla cabe casi completamente en la memoria principal, la manera más rápida de ejecutar consultas sobre ella es empleando índices hash. Estos índices están basados en algoritmos de hash que garantiza la dispersión de valores de la clave. Se utilizan sólo para las comparaciones de igualdad.

El optimizador no puede utilizar un índice hash para acelerar las operaciones ORDER BY. Este tipo de índice no puede utilizarse para la búsqueda de la próxima entrada en orden. MySQL no puede determinar aproximadamente cuántos registros hay entre dos valores es por ello que es utilizado el optimizador de consultas para decidir qué índice emplear.

R-Tree: Son índices utilizados para datos espaciales y multidimensionales. Permiten localizar rápidamente puntos contenidos en

diferentes tipos de áreas. Textuales: Los índices textuales permiten localizar rápidamente palabras completas en campos de texto varchar, text, etc. Son los utilizados habitualmente en la implementación de buscadores.

MySQL dispone de este tipo de índices para tablas MyISAM. Estos índices se implementan como una estructura de árboles B-Tree de dos niveles. Dicho de forma sencilla e imprecisa, el primer árbol B-Tree almacena cada palabra indexada, el número de filas en las que aparece y un puntero la raíz de su segundo árbol asociado. El segundo árbol B-Tree almacena la importancia por cada fila y un puntero a los datos "rowid".

4.2 Operaciones que podemos realizar con índices

- Encontrar rápidamente las filas que coinciden con una WHERE.
- Eliminar las filas de la consideración. Si hay una elección entre varios índices, MySQL normalmente utiliza el índice que se encuentra el menor número de filas.
- Recuperar las filas de otras tablas cuando se realiza una unión.
- Encontrar el minimo ó máximo de un valor específico indexado en alguna columna.

En algunos casos, una consulta puede ser optimizada para recuperar los valores sin consultar a las líneas de datos. Si una consulta utiliza sólo las columnas de una tabla numérica que se forma y que un prefijo izquierdo de algunos de los principales, los valores seleccionados podrán ser recuperados a partir del índice B-Tree para una mejor rendimiento. El índice también puede ser utilizado para comparaciones LIKE si el argumento es una constante cadena que no comienza con un carácter comodín.

4.3 Los Costos del Indexado

En general, si MySQL puede deducir como usar un índice para procesar una consulta más rápidamente, lo hará. Esto significa que, en la mayor parte, si no tenemos un índice en las tablas, nos causaríamos danos nosotros mismos. ¿ Hay desventajas? Sí, hay. Hay costos en ambos en el tiempo y en el espacio. En la práctica, estas desventajas tienden a ser superadas por las ventajas, pero deberíamos saber cuales son.

El primer lugar, los índices aceleran recuperaciones pero hacen lentas las inserciones y borrados, así como también actualizaciones de valor en el indexado de columnas. Esto es, los índices demoran la mayoría de las operaciones que involucran escritura. Esto ocurre porque la escribir un registro requiere escribir no solamente las filas de datos, requiere también hacer cambios en cualquier índice. Entre más índices tenga una tabla, más cambios necesitamos hacer, y el rendimiento se disminuye.

En segundo lugar, un índice toma más espacio en el disco, y los índices múltiples toman más del espacio correspondiente. Esto podría ocasionar enriquecer un tamaño de tabla limita más rápidamente que si no hay índices:

Para una tabla MyISAM, el indexado pesadamente puede ocasionar que el archivo índice alcance su tamaño máximo más rápidamente que el archivo de datos.

Toda tabla InnoDB que están localizadas dentro del tablespace InnoDB compiten el mismo espacio, y agregando índices reduce el almacenamiento dentro del tablespace más rápidamente. Sin embargo, los diferentes archivos usados para tablas MyISAM y BDB, el tablespace compartido de InnoDB no están comprometidos por el tamaño límite del archivo del sistema operativo porque puede configurarse para usar archivos múltiples. Tan grande como sea espacio adicional de disco, podemos expandir el espacio de tablas agregándole nuevos componentes.

Las tablas InnoDB que usan espacios de tablas individuales se limitan la misma manera como las tablas BDB porque los valores de índice y datos se almacenan juntos en un archivo único.

La implicación práctica de ambos factores es que si no necesitamos de un índice en particular que ayude a la consulta se desempeña mejor, no debemos crearlo.

5 CONCLUSIONES

Como conclusión de los diferentes motores de almacenamiento disponibles, hay pocas ra-

zones para no utilizar los motores MyISAM o InnoDB. MyISAM se usará en la mayoría de las situaciones, pero si se tiene un número elevado de actualizaciones o inserciones en comparación con los registros y selecciones entonces se conseguirá un mejor rendimiento con el motor InnoDB. Para obtener el mejor rendimiento de InnoDB es necesario ajustar los parámetros del servidor, de lo contrario, no hay ninguna razón para no utilizarlo.

Pero si ambos motores MyISAM e InnoDB son tan grandes, ¿por qué incluso no consideramos la posibilidad de utilizar los demás tipos de motores? Simplemente porque proporcionan funcionalidad específica que no está disponible por otros medios. MERGE es un motor muy eficaz de consulta para datos múltiples, sus tablas se definen del mismo modo.

El motor de MEMORY es la mejor manera para llevar a cabo un gran número de consultas complejas sobre los datos que sería ineficiente para buscar en un disco. El CSV es un motor de gran manera para la exportación datos que podrían utilizarse en otras aplicaciones. BDB es excelente para los datos que tiene una clave única que es frecuentemente accesada.

Algunos de estos casos son posibles de hacer con InnoDB, pero es la flexibilidad de elegir un tipo de motor que se adapte nuestras condiciones y a los datos que están trabajando, esto es lo que diferencia a MySQL de otras soluciones.

Dado que los índices hacen que las consultas se ejecuten más rápido, podemos estar incitados a indexar todas las columnas de nuestras tablas. Sin embargo, lo que tenemos que saber es que el usar índices tiene un precio. Cada vez que hacemos un INSERT, UPDATE, REPLACE, o DELETE sobre una tabla, MySQL tiene que actualizar cualquier índice en la tabla para reflejar los cambios en los datos.

La razón para tener un índice en una columna es para permitirle a MySQL que ejecute las búsquedas tan rápido como sea posible y evitar los escaneos completos de tablas. Podemos pensar que un índice contiene una entrada por cada valor único en la columna. En el índice, MySQL debe contar cualquier valor duplicado. Estos valores duplicados disminuyen la eficiencia y la utilidad del índice.

Así que antes de indexar una columna, debemos considerar que porcentaje de entradas en la tabla son duplicadas. Si el porcentaje es demasiado alto, seguramente no veremos alguna mejora con el uso de un índice.

Otra cosa a considerar es qué tan frecuentemente los índices serán usados. MySQL puede usar un índice para una columna en particular únicamente si dicha columna aparece en la cláusula WHERE en una consulta.

Si muy rara vez usamos una columna en una cláusula WHERE, seguramente no tiene mucha sentido indexar dicha columna. De esta manera, probablemente sea más eficiente sufrir el escaneo completo de la tabla las raras ocasiones en que se use esta columna en una consulta, que estar actualizando el índice cada vez que cambien los datos de la tabla.

APPENDIX A
PROOF OF THE FIRST ZONKLAR EQUATION

Appendix one text goes here.

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENTS

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.