Optimización en MySQL

Bases de datos II

3º - Grado en Ingeniería Informática

Índice de Contenidos

[Introducción 3](#_Toc5920582)

[Optimizar a nivel de base de datos 3](#_Toc5920583)

[Optimizar a nivel de hardware 3](#_Toc5920584)

[Procesos de optimización implementados por MySQL 3](#_Toc5920585)

[Sentencia EXPLAIN 3](#_Toc5920586)

[Motores de almacenamiento MyISAM e InnoDB 3](#_Toc5920587)

[Índices 5](#_Toc5920588)

[Sentencia OPTIMIZE Table 5](#_Toc5920589)

[Sentencia INSERT DELAYED 6](#_Toc5920590)

[Slow Query Log 6](#_Toc5920591)

[Variables de sistema: long-query-time 6](#_Toc5920592)

[Show ProcessList 7](#_Toc5920593)

[Conclusiones 7](#_Toc5920594)

[Bibliografía 7](#_Toc5920595)

# Introducción

El rendimiento de una base de datos se basas en diversos factores, como las tablas, sentencias y la configuración que utiliza, siendo estos factores a nivel de la propia base de datos. Además, se requiere de una optimización a nivel de hardware, minimizando y haciendo lo más eficiente posible las operaciones de la CPU y las de entrada y salida.

## Optimizar a nivel de base de datos

Las tablas deben estar estructuradas correctamente: que las columnas tengan el tipo de dato adecuado y que cada tabla tenga un número apropiado de columnas.

Uso correcto del indexado, lo que se abordará posteriormente en detalle.

Para favorecer el rendimiento y la escalabilidad se debería usar el motor de almacenamiento más adecuado para cada tabla, siendo en ocasiones mejor el uso de motores transaccionales y, en otros casos, los motores no transaccionales.

## Optimizar a nivel de hardware

A medida que la base de datos se usa con mayor asiduidad, esta podría alcanzar su límite hardware, siendo necesario modificar la aplicación de la base de datos o reconfigurar el servidor de llegar a este punto. Los cuellos de botella de los sistemas suelen ser: las búsquedas en disco, la lectura y escritura en disco, los ciclos del procesador o, aunque en menor medida, el ancho de banda de la memoria.

# Procesos de optimización implementados por MySQL

## Sentencia EXPLAIN

La sentencia EXPLAIN proporciona información sobre como MySQL ejecuta sentencias, en concreto las sentencias SELECT, DELETE, INSERT, REPLACE y UPDATE.

Cuando se utiliza EXPLAIN con una sentencia explicable, MySQL muestra información desde el optimizador explicando cómo se procesaría la sentencia, incluyendo información sobre las como se unen las tablas y en que orden.

Con la ayuda de EXPLAIN se puede ver donde se deberían añadir índices en las tablas de forma que la ejecución de la consulta se vea acelerada al utilizar índices para encontrar filas. También es útil para comprobar si el optimizador está uniendo tablas en el orden óptimo.

EXPLAIN retorna una fila de información por cada tabla utilizada en la sentencia SELECT.

## Motores de almacenamiento MyISAM e InnoDB

InnoDB es el motor de almacenamiento que usan habitualmente los usuarios de MySQL, enfocándose en la fiabilidad y concurrencia de las bases de datos. Existen ciertas consideraciones a tener en cuenta a la hora de optimizar una base de datos que usa este motor de almacenamiento:

En cuanto a la optimización de tablas:

* Si detectamos que una tabla está creciendo de forma considerable, se debería considerar la posibilidad de usar la sentencia OPTIMIZE TABLE, que trataremos en detalle posteriormente. Así reorganizaremos la tabla y compactaremos cualquier espacio usado. Se trata de una medida muy directa y que mejorará el rendimiento de forma considerable.
* Tener una clave primaria larga, tanto en una sola columna como componiendo varias columnas, malgasta mucho espacio. En su lugar se recomienda utilizar claves primarias numéricas autoincrementales.
* Para almacenar cadenas de longitud variable mejor usar varchar en vez de char, ya que char va a tener un tamaño fijo incluso cuando la cadena es más pequeña o nula.
* Para tablas grandes con mucho texto y datos numéricos repetitivos se recomienda usar el formato de tupla COMPRESSED.

En cuanto a la optimización del gestor de transacciones hay que encontrar el equilibrio entre el rendimiento de las características transaccionales y la carga de trabajo de tu servidor.

* Con una base de datos sobrecargada quizás interese establecer autocommit a 0 para poder fusionar varias operaciones de modificación relacionadas en una sola transacción y después hacer commit.
* Se debería evitar realizar un rollback después de insertar, modificar o eliminar una gran cantidad de tuplas.
* Cuando una o varias columnas son eliminadas o modificadas, las propias columnas y sus archivos de recuperación asociados no son físicamente eliminados, si no que se guardan hasta que las transacciones que empezaron antes y todavía no han acabado, finalicen.

En cuanto a transacciones de solo lectura, las transacciones de escritura o las de bloqueo de escritura tienen un id, por lo tanto, eliminar los id de transacción restantes reducirá el tamaño interno de las estructuras consultadas. InnoDB detecta las transacciones de solo lectura cuando empiezan por START TRNSACTION READ ONLY, cuando autocommit está activo o cuando la transacción no empieza por READ ONLY, pero no realiza modificaciones ni bloqueos de lectura.

Hay que optimizar también los logs de recuperación, siguiendo algunas pautas:

* Hacer los fichero de logs de recuperación grande, al menos tanto como el búfer compartido. De no ser así se realizarían demasiadas escrituras en disco innecesarias.
* Aumentar el tamaño del búfer de log. Con ello se podrán ejecutar transacciones mayores sin la necesidad de escribir en el log antes de que la transacción realice el commit.

Para mejorar las entradas y salidas de disco:

* Aumentar el tamaño del búfer compartido.
* Ajustar el método de flush.
* Configurar un tamaño umbral para escritura en el búfer.

El motor de almacenamiento MyISAM tiene un mejor rendimiento con operaciones de leer la mayoría de los datos o aquellas con un grado de concurrencia bajo, porque las tablas tiene un bloqueo de que limita la capacidad de realizar varias modificaciones simultáneas.

Consejos para optimizar tablas MyISAM para mejorar el rendimiento de las consultas:

* Usar la sentencia ANALYZE TABLE después de cargar datos a la tabla.
* Evitar sentencias SELECT en tablas que son modificadas con frecuencia para evitar problemas de bloqueos de tabla.
* Realizar insercciones múltiples y simultáneas de tuplas y evitar eliminar tuplas, o usar OPTIMIZE TABLE después de borrar una tupla.
* En tablas que son modificadas habitualmente hay que evitar tener columnas con un tipo de datos de longitud variable.
* No suele ser útil dividir una tabla en varias por el hecho de que las tuplas se están volviendo grandes.
* Si se necesita hacer cálculos habituales con una tabla con muchas tuplas, podría ser preferible introducir una nueva tabla que see encargue exclusivamente de ello.
* Usar periódicamente OPTIMIZE TABLE para evitar la fragmentación en las tablas con formato dinámico.

En cuanto a inserciones de datos:

* A una tabla MyISAM se le puede realizar inserciones concurrentes para añadir múltiplas tuplas al mismo tiempo que se están ejecutando sentencias select.
* Es posible conseguir que LOAD DATA vaya más rápido cuando tiene muchos índices: realizando un flush a la tabla, eliminando todo uso de índices, insertar los datos con LOAD DATA, si solo pretendes leer de la tabla en el futuro convendría comprimirla, volver a crear los índices y realizar otro flush.
* Para incrementar la velocidad de ejecución de las inserciones que se realizan con múltiples sentencias para tablas no transaccionales hay que bloquear estas.
* Para mejorar el rendimiento de INSERT y de LOAD DATA conviene aumentar el tamaño de la caché de claves.

REPAIR TABLE en MyISAM es similar a usar myisamchk con el fin de reparar operaciones.

## Índices

La creación de índices en una o varias de las columnas objetivo de una consulta es la mejor manera de mejorar el rendimiento de las operaciones SELECT. Estos índices actúan como puntero a las filas de la tabla, permitiendo a la consulta determinar rápidamente que filas cumplen cierta condición en la cláusula WHERE y la recuperación del resto de información de dichas filas. Tampoco es bueno crear índices sin control en todas las columnas de una consulta ya hacen a MySQL gastar espacio y tiempo en determinar que índices se utilizarán.

## Sentencia OPTIMIZE Table

La sentencia OPTIMIZE TABLE reorganiza el almacenamiento físico de los datos de una tabla y los índices asociados con el fin de reducir el espacio de almacenamiento y mejorar la eficiencia de las operaciones de entrada y salida que acceden a la tabla. Los cambios producidos por esta sentencia dependen del motor de almacenamiento que use la tabla optimizada.

Se debería ejecutar después de realizar un gran número de operaciones insert, update o delete en una tabla InnoDB creada a partir de un archivo .ibd, que es un fichero que contiene una única tabla; después de realizar un gran número de operaciones insert, update o delete sobre columnas que son parte de índice FULLTEXT en una tabla InnoDB; o después de eliminar gran parte de una tabla de MyISAM o hacer muchos cambios a una tabla con columnas de longitud variable, como es el caso de varchar, varbinary, blob…

Optimize table funciona de tal manera que, si la tabla tiene columnas eliminadas o partidas, repara la tabla; si los índices no están ordenados, los ordena; y si las estadísticas de la tabla no están actualizadas, las actualiza.

## Sentencia INSERT DELAYED

A la hora de optimizar sentencias INSERT se debería fusionar múltiples operaciones pequeñas en una sola; realizar el menor número de conexiones con la tabla donde se van a insertar las nuevas tuplas y mandar el mayor de estas de una sola vez, retrasando la actualización de índices y la comprobación de consistencia.

Para mejorar la velocidad de las sentencias INSERT hay que usar sentencias INSERT con múltiples listas de VALUE cuando se desee insertar varias tuplas a la vez, usar la sentencia LOAD DATA cuando se cargue una tabla de un archivo de texto o insertar valores explícitamente solo cuando el valor a insertar es distinto del valor por defecto de la columna.

La opción DELAYED es una extensión de MySQL al estándar SQL, que se encuentra actualmente obsoleta desde MySQL 5.6. Anteriormente, cuando un cliente usaba esta sentencia sobre una tabla, la tupla se almacenaba en una cola para insertarse cuando esta tabla no estuviera en uso.

En MySQL 8.0 el servidor reconoce la opción DELAYED, pero la ignora y trata la sentencia como un simple INSERT. Se planea eliminar esta palabra reservada en futuras versiones.

## Slow Query Log

La variable slow query log, de ser activada, sirve para encontrar las sentencias que mayor tiempo de ejecución requieren y que son las principales candidatas para ser optimizadas. Para ello busca las sentencias que llevan más de una cantidad de segundos y que requieren de al menos un número concreto de tuplas para ser revisadas.

Estas sentencias lentas se guardarán para poder ser revisadas en un archivo de log que, de no ser especificada su ubicación, se situará en *nombre\_host*-slow.log en la carpeta de datos del servidor. Además, la variable de estado slow queries se incrementa por cada sentencia lenta.

En el log encontraremos la sentencia ejecutada, seguida de una # y en una línea el tiempo de ejecución de la sentencia, el tiempo de bloqueo, las tuplas examinadas y las que se han mandado al cliente.

Si activáramos la variable log slow extra nos aparecerían en el log muchos más datos relacionados con la sentencia ejecutada, como el número de error que produce, los bytes recibidos, o la hora en la que empezó y acabó, entre otros.

## Variables de sistema: long-query-time

Recientemente explicábamos la variable slow query log, relacionada íntimamente con long query time, debido a que esta última es el número mínimo de segundos que tardarán en ejecutarse las sentencias que se guardarán en el log.

El valor por defecto que recibe es de 10 segundos y el mínimo es 0. Por otro lado, se puede especificar en microsegundos si se requiere de ello.

Este valor se compara en tiempo de ejecución, no de procesado, por lo que una sentencia que está por debajo del umbral en un sistema con poca carga y que, por lo tanto, no entraría en el log, podría superar ese umbral en un sistema con una carga mayor.

## Show ProcessList

Esta sentencia muestra los hilos actualmente en ejecución. Se requiere de permisos para poder ver todos los hilos, de no ser así solo se pueden ver los relacionados con tu usuario.

Se trata de una sentencia muy útil para ver qué ocurre cuando se dan ciertos errores, como el de *demasiadas conexiones*.

Para cada hilo se muestran una serie de datos, como son el id del hilo, el nombre del usuario, que puede ser de un cliente o de un usuario del sistema; el nombre del host y el puerto del cliente que ejecuta una sentencia, la base de datos por defecto, el comando que está ejecutando el hilo, el tiempo en segundos que lleva el hilo en su actual estado, el estado en el que está y, por último, la sentencia que está llevando a cabo el hilo o, de no ejecutar ninguna NULL.

# Conclusiones

# Bibliografía

manual de referencia de MySQL - <https://dev.mysql.com/doc/>