TEMA

Tema 5. Optimización de un SGDB

Administración de sistemas informáticos en red

Administración de sistemas gestores de bases de datos

Autor: Joan Pou

Linkia FP

Tema 5: Optimización de un SGDB

¿Qué aprenderás?

- Optimizar el servidor.
- Ejecutar las consultas en MySQL.
- Utilizar los diferentes tipos de índices.
- Comprobar el estado del sistema.

¿Sabías que…?

- Existen más de 6 millones de copias de MySQL funcionando en la actualidad.
- El tráfico web de MySQL AB superó en 2004 al del sitio de IBM.
- El 2019 fue declarado como el DBMS of the year por DB-Engines Ranking.

5. Optimización de un SGDB

En un SGDB la mayoría de problemas de rendimiento que se experimentan pueden venir provocados por los siguientes motivos.

- Los recursos hardware son insuficientes.
- Gran cantidad de conexiones a la base de datos.
- Configuraciones incorrectas de MySQL.
- El diseño de la base de datos no es correcto, está mal implementado, demasiado normalizado o desnormalizado.
- Las consultas no son eficientes por un mal diseño de los índices.

5.1. Optimización a nivel de base de datos

En este nivel nos hemos de preguntar si las tablas están estructuradas correctamente. Eso significa que:

- Cada tabla tienes las columnas adecuadas para el tipo de trabajo, las aplicaciones que se actualizan frecuentemente suelen tener muchas tablas con pocas columnas mientras que las aplicaciones que analizan grandes cantidades de datos suelen tener pocas tablas con muchas columnas.
- Existen los índices adecuados para que las consultas sean eficientes.
- Se está utilizando el motor de almacenamiento apropiado para cada tabla.

5.1.1. Optimizar el diseño de las tablas

MySQL admite diferentes motores de almacenamiento. Para cada se puede decidir qué método de indexación y almacenamiento utilizar. Una buena elección puede mejorar notablemente el rendimiento. Los motores de almacenamiento son:

- **InnoDB**. El motor de almacenamiento predeterminado en MySQL 8.0. Cumple las características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Permite transacciones y bloqueos. Apropiado para tablas donde predominan las operaciones de actualización respecto a los SELECT.
- **MyISAM**. Estas tablas ocupan poco espacio. Mayor velocidad a la hora de recuperar datos, aunque la evolución de InnoDB está alcanzando velocidades cercanas a MyIsam Es recomendable para aplicativos en los que predominan las sentencias SELECT ante las INSERT / UPDATE.

- **Memory.** Almacena todos los datos en la RAM para un acceso rápido en entornos que requieren búsquedas rápidas de datos no críticos.
- **CSV**. Guarda datos en ficheros de texto usando formato de valores separados por comas. Para intercambiar datos con scripts o aplicaciones que leen y escriben en este formato.
- **Archive**. Para guardar gran cantidad de información sin indexar ocupando poco espacio.
- **Blackhole.** No almacena datos se utiliza en configuraciones de replicación.
- **NDB (NDB cluster).** Es el motor de almacenamiento usado por MySQL Cluster para implementar tablas que se particionan en varias máquinas
- **Merge.** Permite a un desarrollador o DBA de MySQL agrupar lógicamente una serie de tablas MyISAM idénticas y hacer referencia a ellas como un solo objeto.
- **Federated.** Ofrece la capacidad de crear una base de datos lógica a partir de varios servidores MySQL físicos. Para entornos distribuidos y de Data Warehouse.
- **Example.** Es un motor de pruebas que no hace nada. Su propósito es servir como ejemplo en el código fuente MySQL para ilustrar cómo empezar a escribir nuevos motores de almacenamiento.

Respecto a las columnas de las tablas debemos tener en cuenta:

- Utilizar los tipos de datos más pequeños posibles, para ahorrar espacio en disco y memoria. Siempre será mejor un varchar que un char. Si utilizamos un mediumint en vez de int ahorramos el 25% de espacio.
- Declarando columnas not null, las operaciones SQL son más rápidas al permitir un mejor uso de los índices.
- Para identificadores únicos es preferible valores numéricos que valores de cadena. Es más rápido y necesita menos espacio.
- Si la información puede ser guardada en formato numérico o cadena, es preferible siempre que sea numérica.
- Con columnas inferiores a 8KB en tamaño es mejor hacer uso del tipo de datos varchar que blob. Son mejores para utilizar cláusulas GROUP BY y ORDER.
- Si en una tabla se disponen de columnas a las cuales raramente se va a acceder, por ejemplos los datos postales (dirección, código postal, población) es mejor mover estas columnas a otra tabla y relacionarlas con una clave con la tabla original.
- Los nombres de las columnas deberían tener menos de 18 caracteres para que sean portables a otros servidores SQL. Es preferible utilizar el nombre de columna “nombre” que “trabajador_nombre”.

Podemos optimizar también el número de tablas abiertas. Para saber cuántas tablas abiertas tiene el gestor debemos ejecutar la orden mysqladmin status.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqladmin -u root -p status
Enter password: ****
Uptime: 105726  Threads: 2  Questions: 88  Slow queries: 0  Opens: 170  Flush tables: 3
Open tables: 91  Queries per second avg: 0.000
```

La variable table_open_cache define el número de tablas abiertas para todos los subprocesos, y tiene un valor por defecto de 4000. La variable max_connections nos defiene el número máximo de conexiones simultáneas, por defecto 151. Por tanto el valor recomendado debería ser como mínimo 151 * el número de tablas a las que accede cualquier consulta SQL. Este valor se debería incrementar para reservar también para el uso de tablas temporales y ficheros.

Si modificamos este valor no podrá superar:

- El número de ficheros máximos que puede abrir mysql, variable open_files_limit, por defecto 5000.
- El número de ficheros abiertos que puede soportar el sistema operativo.

5.1.2. Índices

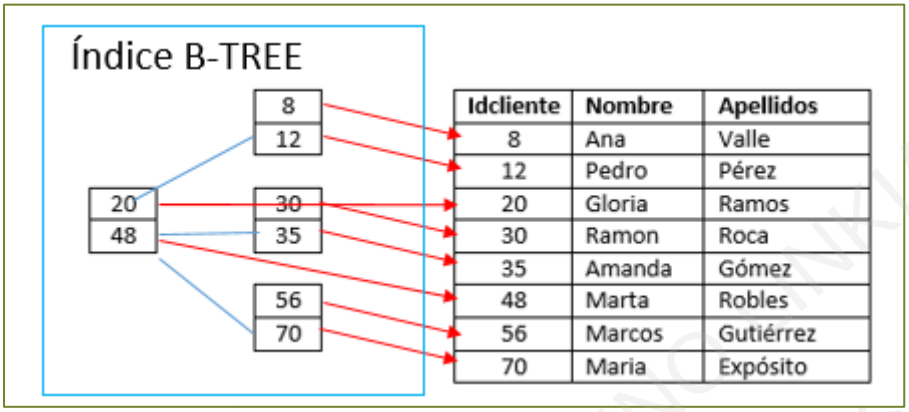
Un índice es un objeto asociado a una tabla que permite que las consultas den una respuesta más rápida. Un índice es física y lógicamente independiente de la tabla a la que está asociado, esto quiere decir que si borramos un índice no afectará ni a la tabla que está asociada ni a otros índices ni a las aplicaciones que trabajen sobre esa tabla. Hemos de tener en cuenta que un índice ocupa espacio en disco y que cuando se producen modificaciones sobre una tabla, los índices asociados a ésta deben ser actualizados. Estas actualizaciones requieren de un esfuerzo adicional de nuestro servidor, por eso es conveniente indexar sólo las columnas que son consultadas habitualmente.

MYSQL utiliza índices para estas operaciones:

- Encontrar registros que coinciden con una cláusula WHERE.
- Para eliminar registros.
- Si un índice tiene varias columnas, puede usar cualquier combinación de ellas siempre empezando por la izquierda.
- Para recuperar filas de otras tablas al realizar uniones.
- Par encontrar valores de funciones de agregado como max() o MIN().
- Para ordenar o agrupar una tabla.

Los índices son estructuras de datos. MySQL soporta las siguientes estructuras:

- **BTREE.** Es una estructura en forma de árbol balanceado (b-tree) formado por un conjunto de nodos ordenados que apuntan a registros del disco, se intenta que haya una cantidad de nodos similar a un lado y otro del árbol. Son recomandables especialmente para consultas con operaciones de comparación (>,<,=,<>) y rangos (BETWEEN).



- **HASH.** Se basan en utiizar una función de hash que transforma los valores de la clave en valores numéricos. Indicada sobre todo para operadores de comparación (=, <>). Como inconvenientes es más lento en las búsquedas por RANGO y puede haber colisiones, dos valores que den el mismo resultado de hash. Se utiliza en los motores MEMORY y NDB. El valor recomendado estará en función del máximo de conexiones simultáneas.
- **R-TREE**. Son índices que se utilizan para bases de datos de tipo espacial.

Cuando creamos un índice en MySQL podemos indicar la estructura y el tipo de índice. Los tipos de índice que utiliza MySQL son:

- **UNIQUE:** formado por campos cuyo valor no se repite en la tabla.
- **PRIMARY:** son los que forman la clave primaria de una tabla, se refiere a un índice en el que todas las columnas deben tener un valor único (al igual que en el caso del índice UNIQUE) pero con la limitación de que sólo puede existir un índice PRIMARY en cada una de las tablas.
- **INDEX:** sobre campos no clave (sus valores se pueden repetir).
- **FULLTEXT**: formados por uno o varios campos de texto, y se suelen usar para buscar palabras dentro de un campo, buscando cadenas sobre todo en tablas MyISAM.
- **SPATIAL:** un índice parcial es el que se forma con parte de las posiciones de una columna.

Una vez decidida la estructura y el tipo de índice debemos especificar sobre que columna o columnas actúa el índice. Teniendo en cuenta este criterio tenemos los siguientes tipos de índices.

- **Parciales:** Indexan solo una parte del campo, especialmente útil para campos que contienen gran cantidad de texto. En este caso indicamos cuantos caracteres del campo forman parte del índice. Por ejemplo:

  ALTER TABLE asignaturas ADD INDEX (descripcion(50))

- **Multicolumna:** son índices sobre más de una columna. Especialmente útiles cuando se suelen hacer muchas consultas con WHERE sobre esas columnas. Por ejemplo:

  ALTER TABLE alumnos ADD INDEX (apellidos, nombre)

- **Clúster:** Es un índice que se almacena junto con los datos de la tabla. Solo puede haber un índice clúster por tabla. Normalmente es la clave primaria.

  La sintaxis a seguir para crear un índice es la siguiente:

  CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX nombre_indice [tipo_Indice] ON nombre_tabla (nombre_columna [(longitud)],..)

  Para mostrar los índices de una determinada tabla utilizamos la instrucción:

  SHOW INDEX FROM nombretabla

  Por ejemplo para mostrar los índices de la tabla film de la base de datos sakila utilizamos la siguiente instrucción:

  USE SAKILA;
  SHOW INDEX FROM film;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	C	Index_	Visible	Expression
film	0	PRIMARY	1	film_id	A	1000	NULL	NULL		BTREE			YES	NULL
film	1	idx_title	1	title	A	1000	NULL	NULL		BTREE			YES	NULL
film	1	idx_fk_language_id	1	language_id	A	1	NULL	NULL		BTREE			YES	NULL
film	1	idx_fk_original_language_id	1	original_language_id	A	1	NULL	NULL	YES	BTREE			YES	NULL

Los datos que obtenemos son los siguientes:

El nombre de la tabla. Non_Unique (0 si es primary o unique 1 en los otros casos.), el nombre del índice, el número de orden dentro de un índice multicolumna.

- Non_Unique: 0 si es primary o unique 1 en los otros casos.
- Key_name: nombre del índice.
- Seq_in_index: El número de orden dentro de un índice multicolumna.
- El nombre del campo usado para crear el índice.
- Collation: Como está ordenado. A Ascendente.
- Cardinality: El número de valores diferentes en el índice.
- Sub_part: El número de caracteres indexados en los índices parciales.
- Packed: Null la columna no está empaquetada.
- Index_type: La estructura del índice (btree, hash, …)

Merece la pena obtener estadísticas sobre el funcionamiento de los índices, porque la efectividad de estos disminuye con el paso del tiempo sobre todo por las operaciones de modificación de datos.

Con el comando OPTIMIZE TABLE podemos actualizar y reordenar los índices al tiempo que recuperamos el espacio que ya no se utiliza. Y con el comando ANALIZE TABLE reconstruye la estructura del índice, para tablas InnoDB. Por ejemplo:

```
use sakila;
analyze table film;
```

Table	Op	Msg_type	Msg_text
sakila.film	analyze	status	OK

5.1.3. Ejecución de consultas

El procesado normal de una consulta consta de 3 partes bien diferenciadas:

- Análisis: en este paso el intérprete comprueba la sintaxis de la sentencia SQL y la trocea en elementos básicos creando un árbol de análisis sintáctico.
- Procesamiento: comprueba el árbol generado en el paso anterior y comprueba que existen las tablas, las columnas y las referencias.
- Optimización: el árbol es válido y pasa a fase de ejecución. Se calcula el coste de cada operación de forma aproximada y el optimizador escoge la de coste más bajo.

El comando EXPLAIN es una herramienta que se utiliza para la optimización de las consultas. El principal objetivo de este comando será detectar si los índices se están usando correctamente y de forma óptima. Esto dependerá de la estructura de nuestras tablas, de las columnas, de los índices y de las condiciones en las clausulas WHERE o GROUP BY.

La información que nos devuelve el comando Explain es la siguiente:

- Identificador.
- Select_type: El tipo. SIMPLE (si no hay UNION ni subconsultas)
- El nombre de la tabla
- Type: Como se unen las tablas: REF,RANGE,ALL, INDEZ
- Key: El índice elegido para realizar la consulta.
- Key_len: Longitud de la clave.
- Ref: Columnas comparadas con el índice.
- Rows: Estimación de las filas a examinar.
- Filtered: Porcentaje de filas filtradas por la condición de la tabla.
- Extra: Información adicional.

Ejemplo 1:

Mostramos todos los datos de una película. Al estar indexado por el título de la película puede acceder directamente al registro indicado. En el resultado comprobamos rows=1 y que ha utilizado el índice idx_title.

```
explain select * from film where title='Casper DragonFly';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	film	NULL	ref	idx_title	idx_title	767	const	1	100.00	NULL

Ejemplo 2:

Mostramos todos los datos de las películas que su longitud es de 170. En este caso como no hay un índice asociado tiene que consultar todas las filas.

```
explain select * from film where length=170;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	film	NULL	ALL	NULL	NULL	NULL	NULL	1000	10.00	Using where

Ejemplo 3:

Creamos un índice en la columna length y volvemos a repetir la misma instrucción. Ahora observamos que utiliza el índice creado y que el número de filas que consulta solo es de 4. Por tanto mejora el rendimiento.

```
create index idx_length on film(length);
explain select * from film where length=170;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	film	NULL	ref	idx_length	idx_length	3	const	4	100.00	NULL

Ejemplo 4:

En operaciones de ordenación que no utilicen la cláusula WHERE muchas veces MYSQL no utiliza los índices ya que tiene que recorrer toda la estructura de datos. Pero podemos mejorar el rendimiento de la consulta obligando a utilizar el índice si existe con la cláusula FORCE INDEX. En las dos siguientes consultas vemos el análisis sin índice y forzando el índice.

```
explain select * from film order by length asc;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	film	NULL	ALL	NULL	NULL	NULL	NULL	1000	100.00	Using filesort

```
explain select * from film  force index (idx_length) order by length asc;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	film	NULL	index	NULL	idx_length	3	NULL	1000	100.00	NULL

5.2. Optimización del servidor

Son muchos los parámetros que se pueden optimizar en el servidor. En un SGDB básicamente son dos: disco y memoria.

Los problemas de disco suelen venir provocados por las búsquedas, las lecturas y escrituras de datos. Aunque los discos modernos tienen tiempos de acceso muy buenos la mejor forma de optimizar el rendimiento es distribuir los datos en más de un disco.

Dividir los datos en tablas más pequeñas también mejora el rendimiento sobre todo de la CPU del servidor.

Cuando el procesador necesita procesar más datos de los que caben en su caché también provoca un cuello de botella, es poco frecuente pero también debe tenerse en cuenta. Si en una tabla hay columnas que se utilizan con poca frecuencia, es mejor dividir la tabla en dos y relacionarlas. De esta forma cargamos menos datos en la memoria.

La memoria a pesar de ser más lenta que el procesador, es más rápida que el disco duro; por lo que podemos evitar accesos al disco si utilizamos memoria caché para almacenar consultas realizadas.

Los motores InnoDB y MyISAM cuentan con buffers para almacenar datos y claves. Si modificamos estos parámetros podemos aumentar el rendimiento. En los motores InnoDB son las siguientes variables:

- **Innodb_buffer_pool_size:** Tamaño del buffer.
- **Innodb_buffer_pool_instances:** Número de divisiones del buffer para asignar una parte a cada hilo o proceso.
- **Innodb_buffer_pool_chunk_size:** Tamaño del fragmento del buffer asignado a cada proceso o hilo.

```
mysql> select @@innodb_buffer_pool_size, @@innodb_buffer_pool_instances,
    -> @@innodb_buffer_pool_chunk_size;
+---------------------------+---------------------------------+---------------------------------+
| @@innodb_buffer_pool_size | @@innodb_buffer_pool_instances | @@innodb_buffer_pool_chunk_size |
+---------------------------+---------------------------------+---------------------------------+
|                   8388608 |                               1 |                         8388608 |
+---------------------------+---------------------------------+---------------------------------+
```

En el caso de motores MyISAM se pueden almacenar los bloques de índices en una estructura especial llamada caché de claves. Para los bloques de datos se utiliza la caché del sistema operativo.

Para controlar el tamaño de la cache se utiliza la variable **key_buffer_size.** Si la establecemos a valor 0 no utiliza esta caché. En el siguiente ejemplo vemos como crear un buffer de caché de claves y asignar las claves de una determinada tabla. Para asignar los índices de una tabla a un determinado caché utilizamos la instrucción **cache index**. En el siguiente ejemplo vemos el funcionamiento.

```
create database pruebas;
use pruebas;
create table test (
id smallint auto_increment primary key,
nombre varchar(20))
engine=MyISAM;
set global cache1.key_buffer_size=128*1024;
SHOW VARIABLES LIKE 'key_buffer_size';
cache index test in cache1;
set global cache1.key_buffer_size=0;
```

Variable_name	Value
key_buffer_size	8388608

Table	Op	Msg_type	Msg_text
pruebas.test	assign_to_keycache	status	OK

También debemos considerar el uso de tablas de memoria (MEMORY) para datos no críticos a los que se accede con frecuencia y que son solo de lectura o que rara vez se actualizan. El tiempo invertido al inicio en copiar estos datos a memoria es muy inferior al rendimiento obtenido en vez de utilizar tablas equivalentes MyISAM o InnoDB.

Mysql incorpora la función **Benchmark()** que podemos utilizar para medir la velocidad de una función o una expresión específica. El valor de retorno es siempre 0 pero mysql nos muestra aproximadamente cuanto tiempo tardó en ejecutar la instrucción.

En el siguiente ejemplo realiza 10 millones de sumas y en un Core i5 a 2,4 Ghz tarda 0,13 segundos.

```
mysql> select benchmark (10000000,1+1);
+--------------------------+
| benchmark (10000000,1+1) |
+--------------------------+
|                        0 |
+--------------------------+
1 row in set (0.13 sec)
```

5.3. Monitorización del sistema

Una de las tareas principales del DBA cuando el sistema está en explotación, es realizar el seguimiento del servidor y del SGBD para comprobar su correcto funcionamiento. Aunque no es recomendable monitorizar la BD constantemente, es preciso hacerlo de forma regular y automatizada.

Deberemos escoger de las herramientas con las que cuentas los SGDB las menos intrusivas, es decir aquellas que menos interfieran en el sistema y nos puedan proporcionar la información de la forma más fiable.

Las herramientas gráficas siempre consumen más recursos y debemos tener en cuenta que si queremos controlar el rendimiento no lo debemos hacer cuando el sistema está trabajando a plena carga.

Conviene recordar que todas las herramientas que sean de tipo gráfico consumirán más recursos del sistema y de la base de datos. No debemos olvidar que observar conlleva un consumo de recursos y que si pretendemos controlar el rendimiento no es lógico hacerlo en un momento en el que la BD está realizando consultas pesadas.

Cuando detectemos que el sistema no funciona correctamente, lo primero que deberemos hacer es determinar la causa de ese funcionamiento no deseado. Los motivos pueden ser diversos. Los parámetros más típicos que podemos controlar son:

- Transacciones por unidad de tiempo: es una de las medidas que se aplica para obtener el rendimiento de las aplicaciones de BD.
- Tiempo de respuesta: mide el tiempo total requerido por una tarea.
- Escalabilidad: resultan medidas útiles para sistemas que deben mantener el rendimiento bajo una carga de trabajo variable.
- Concurrencia: una medida que permite obtener el número de solicitudes por segundo que se generan en el servidor en el momento de máxima actividad.

También disponemos de ficheros log para registrar los eventos que se van produciendo durante la utilización del SGDB. Son los siguientes:

Tipo de registro	Información escrita
Error log	Problemas encontrados al iniciar, ejecutar o detener mysqld
General query log	Establecimiento de conexiones con los clientes y instrucciones recibidas de los clientes
Binary log	Instrucciones que cambian los datos (también se utilizan para la replicación)
Relay log	Cambios de datos recibidos de un servidor de origen de replicación
Slow query log	Consultas que tardaron más de long_query_time segundos en ejecutarse
DDL log (metadata log)	Operaciones de metadatos realizadas por instrucciones DDL

5.3.1. Mysqladmin y la sentencia Show

Mysqladmin es una herramienta instalada junto con el servidor MySQL es muy usada por los administradores para llevar a cabo tareas básicas de MySQL como especificar o cambiar la contraseña del usuario root, supervisar los procesos, comprobar el estado del servidor, etc. Algunas de las operaciones que podemos realizar son las siguientes:

- Cambiar la contraseña del usuario root.

mysqladmin –u root –p 'password antiguo' password 'password nuevo'

- Comprobar si está funcionando el servidor.

mysqladmin –u root –p ping

- Conocer la versión instalada del servidor MySQL.

mysqladmin –u root –p version

- Estado actual del servidor.

mysqladmin –u root –p status

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqladmin -u root -p status
Enter password: ****
Uptime: 9652  Threads: 4  Questions: 100  Slow queries: 0  Opens: 160  Flush tables: 3
Open tables: 81  Queries per second avg: 0.010
```

Nos muestra el tiempo de funcionamiento del servidor, el número de subprocesos, el número de consultas, el número de consultas lentas, el número de tablas que ha abierto el servidor, los reloads de tablas, las tablas abiertas actualmente y la media de consultas por segundo.

- Mostrar las variables de servidor con su valor.

mysqladmin –u root –p extended-status

- Mostrar las variables de sistema del servidor con su valor.

mysqladmin –u root –p variables

- Mostrar los procesos que se están ejecutando.

mysqladmin –u root –p processlist

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqladmin -u root -p processlist
Enter password: ****
+----+-----------------+-----------------+----+---------+-------+-------------------------+------------------+
| Id | User            | Host            | db | Command | Time  | State                   | Info             |
+----+-----------------+-----------------+----+---------+-------+-------------------------+------------------+
| 5  | event_scheduler | localhost       |    | Daemon  | 10223 | Waiting on empty queue  |                  |
| 9  | root            | localhost:55660 |    | Sleep   | 101   |                         |                  |
| 10 | root            | localhost:55661 |    | Sleep   | 101   |                         |                  |
| 20 | root            | localhost:56334 |    | Query   | 0     | starting                | show processlist |
+----+-----------------+-----------------+----+---------+-------+-------------------------+------------------+
```

- Matar un proceso que está dormido. Como por ejemplo el 9 de la captura anterior.

mysqladmin –u root –p kíll número.

- Comandos flush.
  - flush-tables: Limpia todas las tablas.
  - flush-threads: Limpia todos los hilos caché.
  - flush-logs: Limpia la información de logs.
  - flush-privileges: Recarga las tablas grant.
  - flush-status: Limpia las variables de estado.

mysqladmin –u root –p commando-flush

La instrucción show nos da información sobre todos los objetos que tenemos en nuestra base de datos Algunas de los comandos son equivalentes a las instrucciones mysqladmin vistas anteriormente. Algunos comandos a destacar:

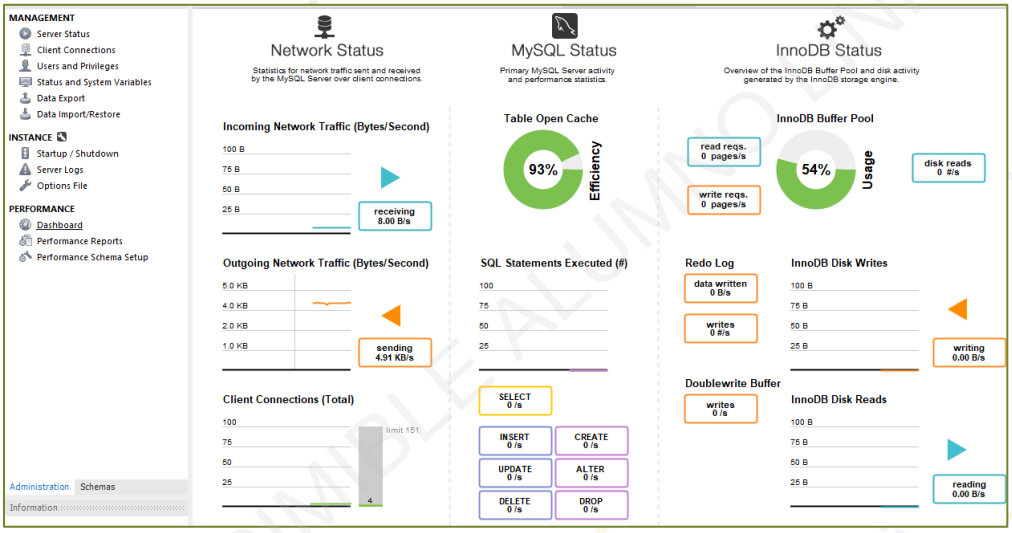
- SHOW STATUS Estado del servidor.
- SHOW VARIABLES. Variables del servidor.
- SHOW PROCESSLIST. Procesos que se están ejecutando.
- SHOW ENGINE INNODB STATUS. Estado del motor de almacenamiento InnoDB.
- SHOW ERRORS y SHOW WARNINGS. Muestran los errores y advertencias.
- SHOW TABLE STATUS FROM database. Muestra mucha información sobre cada tabla.

- También dispone de una serie de comandos vistos en otros temas para ver cómo se han diseñado diferentes objetos: SHOW CREATE DATABASE, SHOW CREATE TABLE, SHOW CREATE USER, SHOW CREATE VIEW, etc.

Las **alertas** nos permiten enviar notificaciones cuando ciertos parámetros han alcanzado o superado unos valores determinados.

Existen diversas herramientas para monitorizar el servidor, algunas en entorno gráfico y otras en entorno de comandos. Entre ellas podemos destacar MySQL Enterprise Monitor, Monyog o InnoTop.

Con el programa MySQL Workbench también podemos obtener información relativa al funcionamiento del servidor.



Video Optimización

Recursos y enlaces

- Optimizar MySQL

- Lista de comandos SHOW

- Motor de almacenamiento INNODB

- Ficheros log

- Instalación de la base de datos employees

Conceptos clave

- **B-TREE y HASH**: Estructuras para almacenar índices.
- **EXPLAIN:** Nos da información sobre la ejecución de una consulta.
- **MYSQLADMIN**: Para realizar tareas de administración desde la consola.
- **SHOW:** Para consultar información sobre diversos objetos de la base de datos.

Test de autoevaluación

¿Los índices más eficientes para casos de comparación y BETWEEN son?

a) B-Tree
b) InnoDB
c) Hash
d) R-Tree

En el fichero slow query log se almacenan:

a) Las consultas que afectan a muchos registros.
b) Las subconsultas.
c) Las consultas que tardan mucho tiempo en ejecutarse.
d) Las consultas de agrupamiento.

Para comprobar la ejecución de una consulta utilizamos la instrucción:

a) FLUSH QUERY
b) EXPLAIN
c) SHOW QUERY
d) ANALYZE

Ponlo en práctica

Actividad 1

Visualiza los procesos que están durmiendo en tu servidor mysql. Posteriormente los eliminas.

SOLUCIONARIOS

Test de autoevaluación

¿Los índices más eficientes para casos de comparación y BETWEEN son?

e) B-Tree

f) InnoDB

g) Hash

h) R-Tree

En el fichero slow query log se almacenan:

e) Las consultas que afectan a muchos registros.

f) Las subconsultas.

g) Las consultas que tardan mucho tiempo en ejecutarse.

h) Las consultas de agrupamiento.

Para comprobar la ejecución de una consulta utilizamos la instrucción:

e) FLUSH QUERY

f) EXPLAIN

g) SHOW QUERY

h) ANALYZE

Ponlo en práctica

Actividad 1

Visualiza los procesos que están durmiendo en tu servidor mysql. Posteriormente los eliminas.

Solución:

Con la instrucción:

mysqladmin –u root –p processlist

visualizamos los que están durmiendo.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqladmin -u root -p processlist
Enter password: ****
+----+-----------------+-----------------+----+---------+-------+-----------------------+------------------+
| Id | User            | Host            | db | Command | Time  | State                 | Info             |
+----+-----------------+-----------------+----+---------+-------+-----------------------+------------------+
| 5  | event_scheduler | localhost       |    | Daemon  | 10223 | Waiting on empty queue |                  |
| 9  | root            | localhost:55660 |    | Sleep   | 101   |                       |                  |
| 10 | root            | localhost:55661 |    | Sleep   | 101   |                       |                  |
| 20 | root            | localhost:56334 |    | Query   | 0     | starting              | show processlist |
+----+-----------------+-----------------+----+---------+-------+-----------------------+------------------+
```

Para matar los procesos 9 y 10 utilizamos la orden:

mysqladmin –u root –p kill 9

mysqladmin –u root –p kill 10