DBMS

MySQL

Temas básicos de Administración

MyISAM

- Bloqueo de tabla
- Aumento del rendimiento si nuestra aplicación realiza un elevado número de consultas “Select”.
- Las tablas pueden llegar a dar problemas en la recuperación de datos.
- Permite hacer búsquedas full-text
- Menor consumo memoria RAM
- Mayor velocidad en general a la hora de recuperar datos.
- Ausencia de características de atomicidad ya que no tiene que hacer comprobaciones de la integridad referencial, ni bloquear las tablas para realizar las operaciones, esto nos lleva como los anteriores puntos a una mayor velocidad.

InnoDB

- Bloqueo de registros
- Soporte de transacciones
- Rendimiento
- Concurrencia
- Confiabilidad
- Permite hacer búsquedas full-text (mysql >= 5.6)
- Permite tener las características ACID (Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español), garantizando la integridad de nuestras tablas.
- Integridad de datos, cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes.

InnoDB se recupera de errores o reinicios no esperados del sistema a partir de sus logs, mientras que MyISAM requiere una exploración, reparación y reconstrucción de índices de los datos de las tablas que aún no habían sido volcadas a disco.

Además es probable que si nuestra aplicación hace un uso elevado de INSERT y UPDATE notemos un aumento de rendimiento con respecto a MyISAM.

ADMINISTRACIÓN DE USUARIOS EN MYSQL

*** Nombre de usuario:**
Cuando creamos un nuevo usuario en MySQL, éste queda identificado por su nombre de usuario más el nombre o IP del ordenador desde el cual hemos dicho que accederá (podemos utilizar el carácter comodín '%' para representar varios ordenadores).

La sintaxis es: usuario@ordenador

Ejemplos de usuarios:
pepito
pepito@'%'
pepito@localhost
pepito@'192.168.0.%'
pepito@'%.midominio.org'

Por ejemplo, el usuario **'*pepito@localhost*'** se considera diferente del usuario '*pepito@192.168.0.%',* aunque tengan el mismo nombre 'pepito', y por lo tanto pueden tener permisos diferentes.

Ver los usuarios:
SELECT User,Host,Password FROM mysql.user;

- **Crear un usuario:**
  CREATE USER usuario [IDENTIFIED BY ‘contraseña’] [, usuario [IDENTIFIED BY ‘contraseña’]] …

Ejemplos:
CREATE USER Pepito IDENTIFIED BY ‘Griyo’;
CREATE USER Anonimo@localhost;
CREATE USER Alumno@’192.168.0.%’ IDENTIFIED BY ‘Alumno’;

- **Borrar un usuario:**
  DROP USER usuario [, usuario] …

Ejemplos:
DROP USER Anonimo;

- **Cambiar el nombre de un usuario:**
  RENAME USER viejo_usuario TO nuevo_usuario [, viejo_usuario TO nuevo_usuario] …

Ejemplos:
RENAME *USER Pepito* TO *Pepito@127.0.0.1*;

- **Cambiar la contraseña de un usuario:**
  SET PASSWORD FOR usuario = PASSWORD(‘contraseña’)

Ejemplos:
SET PASSWORD FOR Pepito = PASSWORD(‘Grillo’)

- Ver los privilegios de un usuario:
  SHOW GRANTS FOR usuario

Ejemplos:
SHOW GRANTS FOR root;
SHOW GRANTS FOR Pepito;

- **Otorga privilegios a un usuario:**

GRANT privilegios ON base_datos.tabla(*columnas*) TO usuario;

Sintaxis ampliada:

GRANT privilegios [(columnas)] [, privilegios [(columnas)]] …
ON [objecto] {tabla | * | *.* | basedatos.*}
TO usuario [IDENTIFIED BY [PASSWORD] ‘contraseña’] [, usuario [IDENTIFIED BY [PASSWORD] ‘contraseña’]] …
[REQUIRE NONE | [{SSL| X509}] [CIPHER ‘cipher’ [AND]] [ISSUER ‘issuer’ [AND]] [SUBJECT ‘subject’]]
[WITH opcion [opcion] …]

privilegios = ALL, ALTER, CREATE, CREATE USER, CREATE VIEW, DELETE, DROP, EXECUTE, INDEX, INSERT, LOCK TABLES, RELOAD, SELECT, SUPER, UPDATE, **GRANT OPTION**, …

objecto = TABLE | FUNCTION | PROCEDURE

opción = GRANT OPTION

| MAX_QUERIES_PER_HOUR count

| MAX_UPDATES_PER_HOUR count

| MAX_CONNECTIONS_PER_HOUR count

| MAX_USER_CONNECTIONS count

Ejemplos:

GRANT UPDATE, INSERT, SELECT ON world.City TO pepito@localhost;

GRANT UPDATE, INSERT, SELECT ON world.City TO fulanito@localhost IDENTIFIED BY ‘nuevapasswd’, tu@equipo.remoto.com;

GRANT UPDATE, INSERT, SELECT ON world.Country TO pepito@’%.empresa.com’, fulanito@’%’, menganito;

GRANT UPDATE, INSERT, SELECT ON world.Country TO pepito@192.168.10.111, fulanito@’192.168.10.%’, menganito;

GRANT UPDATE(Population), SELECT(Name, Population) ON world.Country TO pepito@localhost;

GRANT SELECT ON world.* TO pepito@’%’ REQUIRE ssl;

GRANT SELECT ON world.* TO pepito@’%’ WITH MAX_CONECTIONS_PER_HOUR 3 MAX_QUERIES_PER_HOUR 300 MAX_UPDATES_PER_HOUR 30;

GRANT ALL ON *.* TO operador@localhost

GRANT ALL ON *.* TO operador@localhost WITH GRANT OPTION;

- **Elimina privilegios de un usuario:**

REVOKE privilegios ON base_datos.tabla(columnas) FROM usuario;

Sintaxis ampliada:

REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] …
ON [object_type] {tbl_name | * | *.* | db_name.*} FROM user [, user] …

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] …

Ejemplos:
REVOKE ALL ON *.* FROM pepito@localhost;

* Todos los privilegios se guardan en las tablas ‘user’, ‘db’, ‘tables-priv’, ‘columns_priv’ y ‘host’ de la base de datos ‘mysql’. Se pueden realizar las modificaciones directamente sobre estas tablas, para obtener los mismos resultados que con GRANT, REVOKE, DROP o SET PASSWORD:

```
USE mysql;
SHOW TABLES;
DESCRIBE user;
DESCRIBE db;
DESCRIBE tables_priv;

SELECT User, Host, Select_priv FROM user WHERE User = ‘pepito’;

UPDATE user SET Password = PASSWORD(‘nuevapasswd’) WHERE User = ‘pepito’ AND Host = ‘localhost’;
FLUSH PRIVILEGES;
DELETE FROM user WHERE User = ‘pepito’ AND Host = ‘localhost’;

FLUSH PRIVILEGES;
DELETE FROM user WHERE Password = ”;
FLUSH PRIVILEGES;
```

* Podemos iniciar el servidor pidiéndole que no tenga en cuenta los privilegios otorgados a los usuarios. Por ejemplo, si hemos olvidado la contraseña del administrador de la base de datos y necesitamos poner una nueva:

mysqld –skip-grant-tables –skip-networking
mysql -e “UPDATE mysql.user SET Password = PASSWORD(‘nuevo’) WHERE User = ‘root’”

- De nada sirve controlar los privilegios de los usuarios dentro del servidor de bases de datos si fuera del servidor, en el entorno del sistema operativo, estos usuarios tienen libre acceso al sistema de ficheros. Para evitarlo debemos vigilar los permisos de acceso de los siguientes archivos:

– Las bases de datos y sus tablas, para que usuarios no autorizados no puedan acceder a ellas directamente.
– Ficheros de logs y de estado, para que usuarios no autorizados no puedan acceder a ella directamente.
– Ficheros de configuración, para que usuarios no autorizados no puedan reemplazarlos o modificarlos.

– Programas y scripts que manejan y acceden a bases de datos, para que los usuarios no puedan reemplazarlos o modificarlos.

* Una capa adicional de seguridad nos la da encriptar los datos que escribimos y leemos de la base de datos, mediante las funciones ENCODE(), DECODE(), DES_ENCRYPT(), DES_DECRYPT(), AES_ENCRYPT(), AES_DECRYPT (), y PASSWORD();

* Lecturas para profundizar:
– http://dev.mysql.com/doc/refman/5.0/es/user-account-management.html
– http://dev.mysql.com/doc/refman/5.0/es/account-management-sql.html
– http://dev.mysql.com/doc/refman/5.0/es/privilege-system.html
– http://dev.mysql.com/doc/refman/5.0/es/encryption-functions.html

Ejemplo creamos un usuario

```
CREATE USER 'clasemiercoles'@'localhost' IDENTIFIED VIA
mysql_native_password USING '***';
GRANT SELECT, UPDATE, CREATE VIEW, SHOW VIEW ON *.* TO
'clasemiercoles'@'localhost'
REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0;

GRANT ALL PRIVILEGES ON `facturacion`.* TO 'clasemiercoles'@'localhost';
```