Managing the users for a MySQL server is one of the most important tasks of a MySQL database administrator. Because of the flexibility of the permission system, it is not necessarily a trivial task.

A user in MySQL is a combination of a username and host string. A host string can be an IP address, hostname, fully qualified domain name or netmask.

This means that even they share a username ‘admin’@‘192.168.2.10’ is different from ‘admin’@’192.168.2.%’, and both the users have different passwords and permissions.

mysql> GRANT USAGE ON \*.\* TO ‘admin’@‘192.168.2.10’ IDENTIFIED BY ‘secret2’;

mysql> GRANT ALL ON \*.\* TO ‘admin’@‘192.168.2.20’ IDENTIFIED BY ‘secret1’;

Note: We can create new user and grant privileges at the same time with GRANT statement. It is not recommended that yours GRANT statement to create a user, because it is too easy to forget to specify a password when using with GRANT syntax. Users should be created with CREATE USER statement first, then given permissions with GRANT.

How the server determines who a user is and what permissions are allowed for that user:-

——————————————————————————————————-

Access Control Lists (ACL)

ACL is a list of permissions that is associated with an object. This list is basic the basic for MySQL servers security model and once you understand this it helps greatly when troubleshooting problems with users not being able to connect.

MySQL keeps the ACLs (also called grant tables) cached in memory, when a user tries to authenticate or run a command, MySQL checks the authentication information and permissions against the ACLs, in predetermined order.

If you had two users:

admin@‘192.168.2.%’.

admin@’192.168.2.10’ #this user comes first in the ACLs, because it has specific host string

If you are not sure what user you are actually logged in a you can use the USER() and CURRENT\_USER() functions to determine how you are connected.

User()

This function shows which username and host the MySQL server sees the connection as coming from.

CURRENT\_USER()

This function shows which username and host the connection is actually authenticated.

mysql> select USER();

+-------------------+

| USER() |

+-------------------+

| moulali@localhost |

+-------------------+

1 row in set (0.02 sec)

mysql> select CURRENT\_USER();

+-------------------+

| CURRENT\_USER() |

+-------------------+

| moulali@localhost |

+-------------------+

1 row in set (0.00 sec)

SHOW GRANTS statement with no arguments shows the privileges for the user the connection was authenticated - the privileges for CURRENT\_USER().

mysql> SHOW GRANTS;

+------------------------------------------------------+

| Grants for moulali@localhost |

+------------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'moulali'@'localhost' |

+------------------------------------------------------+

1 row in set (0.00 sec)

Wildcards:

—————

Wildcard characters (% and \_) are allowed in host strings.

This is another source of confusion as admin@’192.168.2.10 is completely different user than admin@192.168.2.%.

How the MySQL server orders the access control list:

MySQL server orders the access control list with the least specific hosts last. This means that hostnames and IP’s without wildcards or net masks are placed before hostnames and It’s with wildcards and net masks. MySQL matches the most specific user and hostname.

admin@192.168.2.10 authenticates first because it has a specific host than the user admin@192.168.2.%

Lets create two users with same same username but different host names to illustrate above:

Here we are going to use same password to see which user it will authenticate first.

User1: admin@192.168.2.10

User2: admin@192.168.2.%

mysql> CREATE USER 'admin'@'192.168.2.10' IDENTIFIED BY 'admin';

Query OK, 0 rows affected (0.03 sec)

mysql> CREATE USER 'admin'@'192.168.2.%' IDENTIFIED BY 'admin';

Query OK, 0 rows affected (0.00 sec)

Note: The CREATE USER and GRANT USER commands can both be used to create users without passwords. This is very insecure and should be avoided. Always use the IDENTIFIED BY clause with these commands.

Moulalis-Air:~ moulali$ mysql -u admin -h 192.168.2.10 -p

Enter password:

ERROR 2003 (HY000): Can't connect to MySQL server on '192.168.2.10' (51)

The connection will authenticate as the user admin@192.168.2.10 because it has a more specific host than the user admin@192.168.2.% and, therefore, appeared earlier in MySQLs access control list.

System tables:

——————-

All the user and permission information is stored in the mysql database in a set of tables known as the grant tables. I f you execute ‘SHOW DATABASES’ on a typical default install of MySQL it will look like the following:

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------+

4 rows in set (0.01 sec)

The information\_schema database really is not a database but an interface to various system metadata.

As mentioned earlier the mysql database stores the user information. In addition to the grant tables, the mysql database has tables containing other system information. For, example a table called event is used by the event scheduler.

mysql> show tables;

+---------------------------+

| Tables\_in\_mysql |

+---------------------------+

| columns\_priv |

| db |

| engine\_cost |

| event |

| func |

| general\_log |

| gtid\_executed |

| help\_category |

| help\_keyword |

| help\_relation |

| help\_topic |

| innodb\_index\_stats |

| innodb\_table\_stats |

| ndb\_binlog\_index |

| plugin |

| proc |

| procs\_priv |

| proxies\_priv |

| server\_cost |

| servers |

| slave\_master\_info |

| slave\_relay\_log\_info |

| slave\_worker\_info |

| slow\_log |

| tables\_priv |

| time\_zone |

| time\_zone\_leap\_second |

| time\_zone\_name |

| time\_zone\_transition |

| time\_zone\_transition\_type |

| user |

+---------------------------+

31 rows in set (0.00 sec)

The tables that of interest when it comes user management:

Columns-priv

db

procs\_priv

tables\_priv

user

You often manipulate these tables indirectly through the statements such as CREATE USER, GRANT, REVOKE and DROP USER. These commands designed for user management.

One of the more common problems of a database administrator is seeing what users are already exist and what privileges they have. I you logged in to the server with appropriate privileges, the following will show all usernames, hosts and password hashes on the system.

mysql> use mysql;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select user, host, password from user;

ERROR 1054 (42S22): Unknown column 'password' in 'field list'

mysql> select user, host, authentication\_string from user;

+---------------+-----------+-------------------------------------------+

| user | host | authentication\_string |

+---------------+-----------+-------------------------------------------+

| root | localhost | \*F13E2AEBE3248683768586F66B74455B420085FD |

| mysql.session | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| mysql.sys | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| moulali | localhost | \*C0733B6410237EEF08CEC43464599CFB15C5C01A |

+---------------+-----------+-------------------------------------------+

4 rows in set (0.00 sec)

Note: If user has a blank password, the password field will be empty.

Managing User Accounts:

——————————————————————-

MySQL server provides a number of commands used for managing users.

CREATE USER - to create new user

DROP USER - to delete user from system

mysql> CREATE USER 'dummy'@'localhost' IDENTIFIED BY 'dummy';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'dummy'@'localhost';

+-------------------------------------------+

| Grants for dummy@localhost |

+-------------------------------------------+

| GRANT USAGE ON \*.\* TO 'dummy'@'localhost' |

+-------------------------------------------+

1 row in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON \*.\* to 'dummy'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'dummy'@'localhost';

+----------------------------------------------------+

| Grants for dummy@localhost |

+----------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'dummy'@'localhost' |

+----------------------------------------------------+

1 row in set (0.00 sec)

The above user ‘dummy’ have all privileges except GRANT privilege to grant privileges to other users.

mysql> CREATE USER 'dummy2'@'localhost' IDENTIFIED BY 'dummy2';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS for dummy2@localhost;

+--------------------------------------------+

| Grants for dummy2@localhost |

+--------------------------------------------+

| GRANT USAGE ON \*.\* TO 'dummy2'@'localhost' |

+--------------------------------------------+

1 row in set (0.00 sec)

mysql> GRANT ALL ON \*.\* to dummy2@localhost WITH GRANT OPTION;

Query OK, 0 rows affected (0.01 sec)

mysql> SHOW GRANTS for dummy2@localhost;

+-----------------------------------------------------------------------+

| Grants for dummy2@localhost |

+-----------------------------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'dummy2'@'localhost' WITH GRANT OPTION |

+-----------------------------------------------------------------------+

1 row in set (0.00 sec)

The above user ‘dummy2’ all the privileges including RANT privilege.

WITH GRANT OPTION should be left out if the user need not able to grant other users privileges.

mysql> DROP USER 'dummy'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT user, host, authentication\_string from user;

+---------------+--------------+-------------------------------------------+

| user | host | authentication\_string |

+---------------+--------------+-------------------------------------------+

| root | localhost | \*F13E2AEBE3248683768586F66B74455B420085FD |

| mysql.session | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| mysql.sys | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| moulali | localhost | \*F13E2AEBE3248683768586F66B74455B420085FD |

| admin | 192.168.2.10 | \*4ACFE3202A5FF5CF467898FC58AAB1D615029441 |

| admin | 192.168.2.% | \*4ACFE3202A5FF5CF467898FC58AAB1D615029441 |

| dummy2 | localhost | \*952A3EB45B46BAC89B417BE792C7649BDBF87632 |

+---------------+--------------+-------------------------------------------+

7 rows in set (0.00 sec)

Dropping the user removes all their privileges, even if you create the exact same username and host the new user does not retain the privileges of the previous user. You are staring from scratch.

The RENAME USER command renames an existing account, the RENAME command will return an error if the new user already exists. It is same like creating new user that already exists.

mysql> CREATE USER 'dummy2'@'localhost' IDENTIFIED BY 'dummy2';

ERROR 1396 (HY000): Operation CREATE USER failed for 'dummy2'@'localhost'

mysql> RENAME USER 'admin'@'192.168.2.10' TO 'dummy2'@'localhost';

ERROR 1396 (HY000): Operation RENAME USER failed for 'admin'@'192.168.2.10'

mysql> rename user moulali@localhost to moulali2@localhost;

Query OK, 0 rows affected (0.00 sec)

mysql> select user, host, authentication\_string from user;

+---------------+-----------+-------------------------------------------+

| user | host | authentication\_string |

+---------------+-----------+-------------------------------------------+

| root | localhost | \*F13E2AEBE3248683768586F66B74455B420085FD |

| mysql.session | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| mysql.sys | localhost | \*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |

| moulali2 | localhost | \*C0733B6410237EEF08CEC43464599CFB15C5C01A |

+---------------+-----------+-------------------------------------------+

4 rows in set (0.00 sec)

mysql> show grants for moulali2@localhost;

+-------------------------------------------------------+

| Grants for moulali2@localhost |

+-------------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'moulali2'@'localhost' |

+-------------------------------------------------------+

1 row in set (0.00 sec)

When a user is renamed, the password and the privileges retained by the new user. Also RENAME USER does not change any database object properties(tables, views, stored routines, and triggers) that the user created.

GRANT and REVOKE commands:-

———————————————-

These are the two commands are used to control a users privileges.

GRANT - is used to give an existing user privileges

REVOKE - is used to remove privileges.

If user does not exist, GRANT will create a new user at the same time you are giving them privileges.

Note: It is not recommended that you use GRANT to create a user, because it is too easy to forget to specify a password when using the GRANT syntax. Users should be created with CREATE USER first, then given permissions with GRANT.

There are five levels that privileges can be granted.

Global

Database

Table

Column

Routine

Global Privileges:

Global privileges apply to all databases on a MySQL server. These privileges stored in the mysql.user table.

GRANT privilege\_list ON \*.\* - to grant only global level privileges.

REVOKE privilege\_list ON \*.\*. - to revoke only global level privileges.

mysql> CREATE USER 'test'@'localhost' IDENTIFIED BY 'test';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test'@'localhost';

+------------------------------------------+

| Grants for test@localhost |

+------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test'@'localhost' |

+------------------------------------------+

1 row in set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'test'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test'@'localhost';

+---------------------------------------------------+

| Grants for test@localhost |

+---------------------------------------------------+

| GRANT ALL PRIVILEGES ON \*.\* TO 'test'@'localhost' |

+---------------------------------------------------+

1 row in set (0.00 sec)

The user test granted ALL (Except GRANT) privilege on global level, that mean he can access and read/write/update/delete the data from all the databases.

mysql> CREATE USER 'test2'@'localhost' IDENTIFIED BY 'test2';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test2'@'localhost';

+-------------------------------------------+

| Grants for test2@localhost |

+-------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test2'@'localhost' |

+-------------------------------------------+

1 row in set (0.00 sec)

mysql> GRANT SELECT ON \*.\* TO 'test2'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test2'@'localhost';

+--------------------------------------------+

| Grants for test2@localhost |

+--------------------------------------------+

| GRANT SELECT ON \*.\* TO 'test2'@'localhost' |

+--------------------------------------------+

1 row in set (0.00 sec)

The user test2 granted only SELECT privilege on global level, that mean he can access and read the data from all the databases.

Note:

The user with username root created by default is only specific because of the permission it has. The root username has no significance and can be deleted from a fresh installation with no issues(servers currently in use may be depending on the root user for backups or some other important tasks). Create new user with all privileges and also with grant option then delete root user.

Database level:

Database privileges apply to all objects of a specific database. The privileges are stored in the mysql.db database.

GRANT ALL ON db\_name.\*

REVOKE ALL ON db\_name.\*

mysql> GRANT ALL ON mysql.\* TO 'test2'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test2'@'localhost';

+----------------------------------------------------------+

| Grants for test2@localhost |

+----------------------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test2'@'localhost' |

| GRANT ALL PRIVILEGES ON `mysql`.\* TO 'test2'@'localhost' |

+----------------------------------------------------------+

2 rows in set (0.00 sec)

Table level:

Table privileges to all columns in a given table. These privileges are stored in the mysql.tables\_priv table.

GRANT ALL ON db\_name.table\_name

REVOKE ALL ON db\_name.table\_name

mysql> GRANT ALL ON mysql.user TO 'test2'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS for 'test2'@'localhost';

+---------------------------------------------------------------+

| Grants for test2@localhost |

+---------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test2'@'localhost' |

| GRANT ALL PRIVILEGES ON `mysql`.\* TO 'test2'@'localhost' |

| GRANT ALL PRIVILEGES ON `mysql`.`user` TO 'test2'@'localhost' |

+---------------------------------------------------------------+

3 rows in set (0.00 sec)

If you had only specified table\_name rather than db\_name.table\_name, the GRANT or REVOKE statement applies to the table table\_name in the default database. To keep from having unexpected results, we would recommend you use the full database\_name.table\_name format instead.

Column level:

Column level privileges apply to one or more columns in a given table. These privileges are stored in the mysql.columns\_priv table.

When using the REVOKE command to remove column level privileges, you must specify the same that are granted.

The column or columns for which the privileges are to be granted are enclosed within parentheses.

mysql> CREATE USER 'test3'@'localhost' IDENTIFIED BY 'test3';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test3'@'localhost';

+-------------------------------------------+

| Grants for test3@localhost |

+-------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test3'@'localhost' |

+-------------------------------------------+

1 row in set (0.00 sec)

mysql> GRANT SELECT(user,host) ON mysql.user TO 'test3'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test3'@'localhost';

+--------------------------------------------------------------------+

| Grants for test3@localhost |

+--------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test3'@'localhost' |

| GRANT SELECT (host, user) ON `mysql`.`user` TO 'test3'@'localhost' |

+--------------------------------------------------------------------+

2 rows in set (0.00 sec)

mysql> SELECT CURRENT\_USER();

+-----------------+

| CURRENT\_USER() |

+-----------------+

| test3@localhost |

+-----------------+

1 row in set (0.00 sec)

mysql> USE mysql;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> SELECT user, host from mysql.user;

+---------------+--------------+

| user | host |

+---------------+--------------+

| admin | 192.168.2.% |

| admin | 192.168.2.10 |

| dummy2 | localhost |

| moulali | localhost |

| mysql.session | localhost |

| mysql.sys | localhost |

| root | localhost |

| test | localhost |

| test2 | localhost |

| test3 | localhost |

+---------------+--------------+

10 rows in set (0.00 sec)

mysql> SELECT user, host, authentication\_string from mysql.user;

ERROR 1143 (42000): SELECT command denied to user 'test3'@'localhost' for column 'authentication\_string' in table 'user'

Routine level:

The CREATE ROUTINE, ALTER ROUTINE,EXECUTE and GRANT privileges apply to stored routines( functions and procedures). They can be granted at the global and database levels. Also except for CREATE ROUTINE, these privileges can be granted at the routine level for individual routines. The privileges are stored in the mysql.procs\_priv table

GRANT CREATE ROUTINE ON database.\* T0 ‘admin’@‘localhost’;

GRANT EXECUTE ON PROCEDURE database.backup TO ‘admin’@‘localhost’;

In combination with the five privileges levels(global, database, table, column and routine) allow for any level of granularity needed by a database administrator. This granularity creates complexity, but the end results is a more controllable and secure system.

Note:

Privileges are checked until either is allowed or the end of the ACL is reached.

If you want to query the table production.employee, then MySQL server first checks to see if you have global access privileges. If so, the query is executed.

If you do not have privileges at the global level, then MySQL checks for privileges at database(production) level. If you do not have privileges at database level , then table(employee) level privileges are checked. If this fails the column level privileges are checked and if this fails the user is denied access.

If a check returns positive at any levelmysqld stops checking privileges.

REVOKE statement:

The REVOKE statement is used to remove privileges from a user account. Just as the GRANT statement there are five levels that you can revoke privileges from global, database, table, column and routine.

mysql> REVOKE ALL ON \*.\* FROM 'test2'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'test2'@'localhost';

+-------------------------------------------+

| Grants for test2@localhost |

+-------------------------------------------+

| GRANT USAGE ON \*.\* TO 'test2'@'localhost' |

+-------------------------------------------+

1 row in set (0.00 sec)

After revoking test2 user have only USAGE privilege, that mean no privileges.

Note:

Even if you revoke all privileges, the user is not dropped(they are still visible in the mysql.user system table). At this point, the user has the USAGE privilege, which means that they can still connect to the server and execute a few commands such as SHOW VARIABLES and SELCT NOW(). To drop a user, you must use the DROP USER. It is a best practice to always drop users after revoking all their privileges.

The REVOKE command very similar to the GRANT command.

SHOW GRANTS:

The SHOW GRANTS command is used to show a users privileges.

If the user has the GRANT PRIVILEGES privilege, then the user can also view grants of other users.

mysql> SHOW GRANTS FOR 'test2'@'localhost';

+--------------------------------------------+

| Grants for test2@localhost |

+--------------------------------------------+

| GRANT SELECT ON \*.\* TO 'test2'@'localhost' |

+--------------------------------------------+

1 row in set (0.00 sec)

This was done with the root user who have all privileges, including the GRANT OPTION. Because this user has the GRANT OPTION, it can grant privileges to other users, and use the SHOW GRANTS command to display grants for other users.

Resetting the ROOT password:

——————————————

There are times when the password for the root user is lost. It is not trivial mater to reset the password and requires a server restart. However, there are times when this proves necessary.

There are two methods for recovering the password. Both have their benefits and drawbacks.

Method1:

—skip-grants-table option

When the server is restarted with this option it starts “wide open” with anyone able to log in with all privileges without even specifying a username. This is huge security risk and must be carefully considered production system.

We would recommend that when you start MySQL server with —skip-grant-table option, that you also add the —bind-address=127.0.0.1 option, which does not allow remote network connections to the MySQL server. This minimises the risk somewhat.

Procedure:

1. Edit the configuration file and add the skip-grant-tables and optionally bind-address option to the mysql section

2. Restart the MySQL server.

3. Connect to the mysql server using the mysql client. NO password or username needs to be specified. If you so used the skip-networking option, you must run the mysql client from the server itself

4. Now change the password

5. Shutdown server and remove skip-grant-tables and other settings

6. Now restart the server normally.

You can’t create new user when MySQL server running with —skip-grant-tables mode

mysql> create user dummy@localhost identified by 'dummy';

ERROR 1290 (HY000): The MySQL server is running with the --skip-grant-tables option so it cannot execute this statement

mysql> exit

Bye

This is straightforward procedure and in an emergency might be the only method you have time to perform. However, as pointed out, it is not inherently secure.

The second method of resetting the root password is more secure. The basis for this recovery method is using an initialisation file at server startup to execute the same UPDATE and FLUSH PRIVILEGES commands we used in the previous example.

UPDATE mysql.user SET authentication\_string=PASSWORD(“new\_password”) WHERE user=‘root’;

FLUSH PRIVILEGES;

Save above in a text file

> mysql —console —init-file=<location\_of\_the\_file>

Debugging User Account Problems:-

————————————————

There are times when users will come to the database administrator with complaints that a newly created account isn’t working, When this happens, there are some common issues you can look for to help when troubleshooting.

BAD PASSWORD:

A common problem is that the account does not work because of an improperly keyed password or a miscommunication in what the password should be Here is the GRANT statement for a user.

There must be issue with one of the below:

Username, host, password

You need to reset the password

ACCESS ISSUES:

A more subtle issue is that of access. The user has the right username and password, but the host is either set incorrectly or there are multiple hosts listed with the same username and the wrong host is being used in authentication.

Client does not support authentication protocol:

Some client supports only older protocols.

SET PASSWORD FOR ‘admin’@‘localhost’= OLD\_PASSWORD(<password>);

This sets the password to format of the older authentication protocols.

If at all possible, you should use the newer authentication protocol as it is much more secure.

Can’t connect to local mysql through socket:

This shows up quite frequently.

Moulalis-Air:~ moulali$ mysqladmin -u root shutdown

Moulalis-Air:~ moulali$ mysql

ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)

Check to make sure that the MySQL server is actually running.

Ps aux | grep mysql

Issue the above command in command prompt to see the mysql server process is running or not. If you don’t get any results then you should start the My SQL server.

Check mysqld file permissions

You use the chown(change owner) command as root

Chown -R mysql:mysql <location of mysql file>

I do not have right permissions: