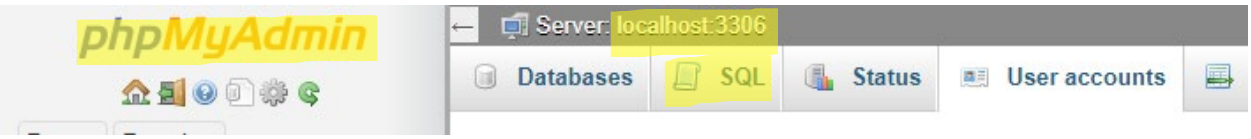
Lesson 12-User Management in PHPMyAdmin

Lesson 12 Concepts

1. Creating a user
2. Passwords in MySQL
3. Renaming a user
4. Dropping a user
5. Privileges
6. When to create a user

Click on phpMyAdmin logo on the top left:



Click on SQL tab and open the query box.

I. Creating a New User

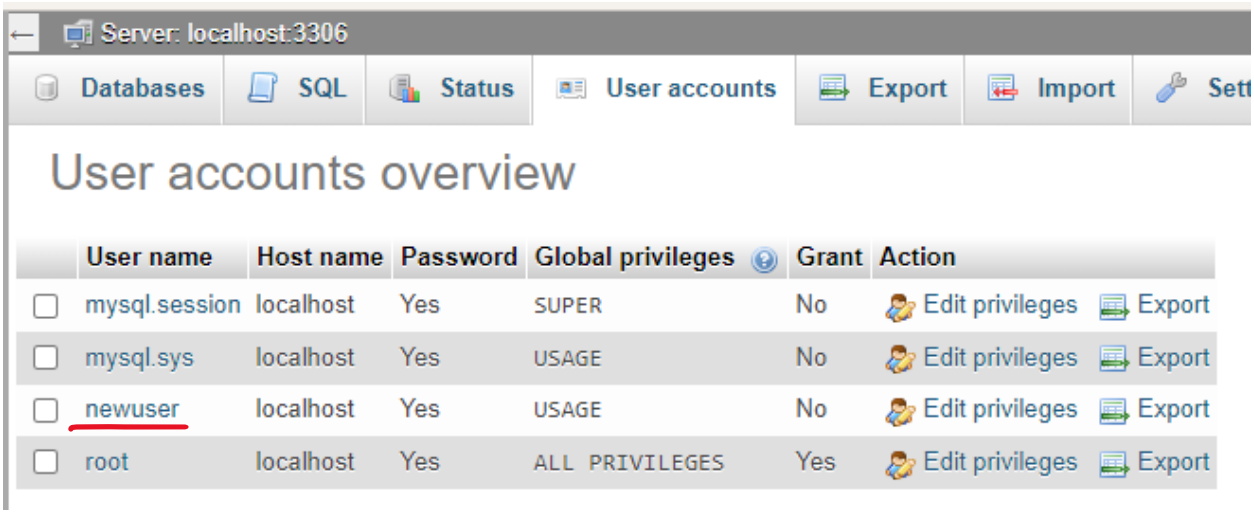
To create a new user, you can use the CREATE USER statement.

```
CREATE USER 'newuser'@'localhost' IDENTIFIED BY
'password';
```

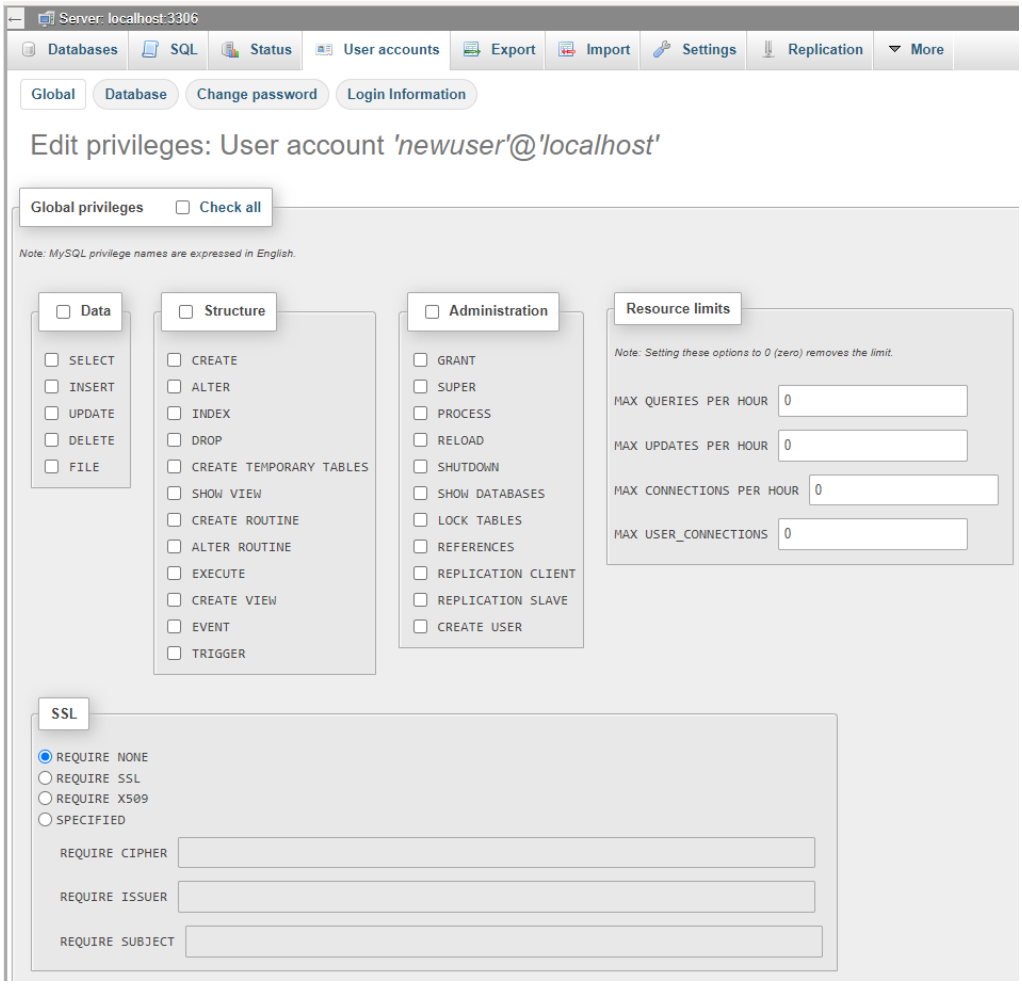
The user name and password can be set to whatever you want, so long as there isn't an existing user with the same user name.
If you didn't want to throw an error while running this statement, use the command **`CREATE USER IF NOT EXISTS.`** If you use this command, be sure check whether the new user is created!
The domain (i.e. the 'host name') is the IP address of a computer with a valid database connection. You can use the % wildcard symbol (just like in our LIKE clauses!) to set the domain to "any IP address", or you could specify the IP of a specific machine.
Most often the domain is set to localhost

Using phpMyAdmin interface for user management :

Confirm the new user by clicking on the 'User accounts' tab at the top navigation bar.



It is also possible to use the PHPMyadmin interface to manage user accounts. In the screenshot above, user privileges can be edited by clicking on the "edit privileges" link. Clicking on the "edit privileges" link will show:



What is localhost?
Localhost is shorthand for "**this machine**". If you've got your web server (i.e. your server-side application, be it PHP etc) and your database server running on the same machine, then you can give your application (i.e. SQL statements in your PHP or whatever) permission to access the database. This is by far the most common way to set up your technology stack.
Using your admin account, also set to the localhost domain, you can still access your database directly by using the SSH protocolOpens in a new window to log in to the server.
Specifying localhost as the user domain is **the most secure way** to set up your database user accounts. You can't access the database without server access, and if a malicious user has direct server access, then there will be trouble.

II. Passwords in MySQL

Setting a password for a new user
When creating a new user, you can specify a password using the IDENTIFIED BY clause in the CREATE USER statement. See above.

Default Password Expiration
MySQL allows you to set a default password expiration policy. This can be done by modifying the default_password_lifetime system variable.

A. Setting Passwords to Never Expire

```
SET GLOBAL default_password_lifetime = 0;
```

B. Setting Passwords to Expire After a Certain Number of Days
If you want user passwords to expire after a certain number of days, you can set default_password_lifetime to that number. If the password lifetime is 90 days:

```
SET GLOBAL default_password_lifetime = 90;
```

Note: changing the default_password_lifetime system variable only affects the passwords of new accounts and the passwords of existing accounts when they are next changed. It does not cause existing passwords to immediately expire.

Show Global Variables
SHOW GLOBAL VARIABLES statement displays the values of all global system variables.

```
SHOW GLOBAL VARIABLES;
```

This statement will return a list of all global system variables along with their current values.

You can also filter the list to show only certain variables by using the LIKE clause with a pattern.

```
SHOW GLOBAL VARIABLES LIKE 'max_connections';
```

This statement will return the current global setting for the maximum number of simultaneous client connections allowed by MySQL.

To change the value of a global variable, you can use the SET GLOBAL statement:

```
SET GLOBAL max_connections = 200;
```

Resetting an expired password

If MySQL has the following error message:

ERROR 1820 (HY000): You must reset your password
using ALTER USER statement before executing
this statement.

Then you can update your password as follows:

```
ALTER USER USER() IDENTIFIED BY 'newPassword';
```

The USER() function **returns the current user** in the format userName@domain.

III. Renaming a User

To rename a user, you can use the RENAME USER statement.

```
RENAME USER 'olduser'@'localhost' TO
'newuser'@'localhost';
```

This statement renames the user olduser to newuser. The 'localhost' part specifies the host from which the user can connect to the MySQL server. If the user can connect from any host, use '%' instead of 'localhost'

IV. Dropping a User

To drop a user, you can use the **DROP USER** statement.

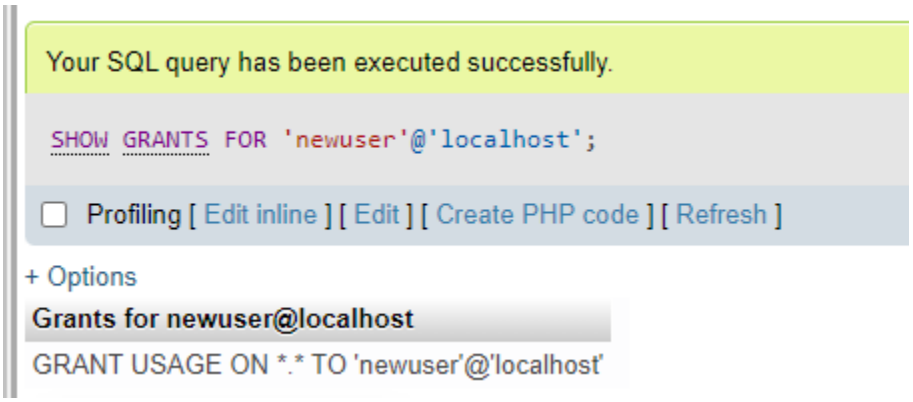
```
DROP USER 'user'@'localhost';
```

V. Database Privileges

To see user's existing privileges:
Syntax:

```
SHOW GRANTS FOR 'userName'@'domain';
```

By default, users have no privileges.
If you run the `SHOW GRANTS` command for a user with no privileges, the database will give you a response like this:



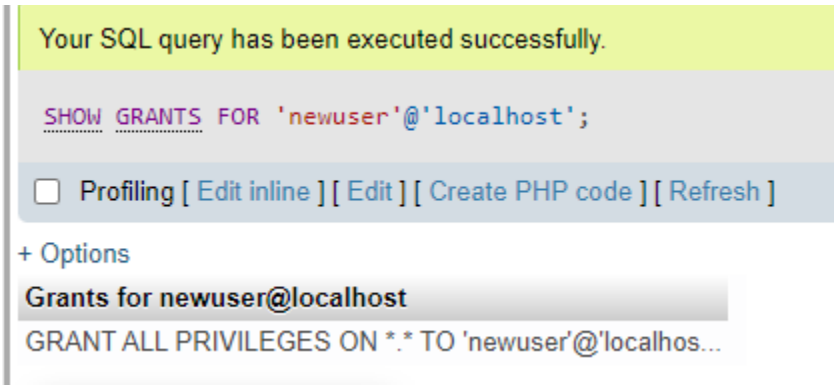
Granting privileges:
The only user who has privileges on a mysql database by default is root (the user who creates the MySQL installation).

To grant all privileges to a user on a database, you can use the following statement:

```
GRANT ALL PRIVILEGES ON *.* TO 'username'@'domain';
```

The *.* refers to multiple databases running under a single MySQL installation. Specifically, it means "**all tables, views and procedures within all databases within this server**". The first asterisk means "all databases", and the second asterisk "all tables, views and procedures".

The results of the `GRANT ALL PRIVILEGES` statement in phpMyadmin :



Granting specific privileges:
If you want to give a user to have limited privileges then the syntax would look like this:

Read and insert:

```
GRANT SELECT, INSERT ON databaseName.tableName
  TO 'userName'@'domain';
```

Call procedure:

```
GRANT EXECUTE
  ON PROCEDURE databaseName.procedureName
  TO 'userName'@'domain';
```

For more different types of user permissions, check the mysql documentation.

Granting Multiple Privileges
To grant multiple privileges to a user, you can list the privileges separated by commas.

```
GRANT SELECT, INSERT, DELETE
  ON database_name.table_name
  TO 'user1'@'domain';
```

Granting Privileges to Multiple Users
To grant privileges to multiple users, you can execute multiple GRANT statements.

```
GRANT SELECT, INSERT, DELETE
  ON database_name.table_name
  TO 'user1'@'domain',
     'user2'@'domain',
     'user3'@'domain',
```

```
    'user4'@'domain',
    'user4'@'domain';
```

Revoking Privileges
To revoke privileges from a user, you can use the REVOKE statement.

```
REVOKE SELECT ON database.* FROM 'user'@'localhost';
```

To revoke INSERT privileges:

```
REVOKE INSERT ON {databaseName}.{tableName}
  FROM 'userName'@'domain';
```

To erase all their privileges (without dropping them as a user):

```
REVOKE ALL PRIVILEGES, GRANT OPTION
 FROM 'userName'@'domain';
```

Flush Privileges
FLUSH PRIVILEGES statement tells MySQL to re-check user privileges. Referring to the MySQL documentation, when you use "account management" statements such as GRANT, REVOKE, or SET PASSWORD, the server notices these changes and automatically updates the privileges. However, if you modify the grant tables directly using statements such as INSERT, UPDATE, or DELETE, the server does not notice these changes. In this case, you need to use the FLUSH PRIVILEGES statement to tell the server to reload the grant tables.

```
FLUSH PRIVILEGES;
```

This statement doesn't require any parameters. After running this statement, any changes you've made to the grant tables will take effect.

When to Create a User

1. **New User Access**: When a new person or application needs access to certain databases on your MySQL server, you should create a new user. This allows you to control the access permissions for each user separately.
2. **Different Access Levels**: If different people or applications need different levels of access to your MySQL databases (some might only need to read data (SELECT), while others might need to modify data (INSERT, UPDATE, DELETE)), then creating separate users with appropriate privileges is a good approach.

3. **Security**: To follow the principle of least privilege, it's a good practice to create separate users for different tasks and grant them only the privileges they need to perform their tasks. This minimizes the potential damage if a user's credentials are compromised.
4. **Organization**: If you have many people or applications accessing your MySQL server, creating separate users can help keep track of who is doing what.