**Practical list Q2**

**CREATE USER**

The MySQL user is a record in the **USER** table of the MySQL server that contains the login information, account privileges, and the host information for MySQL account. It is essential to create a user in MySQL for accessing and managing the databases.

When the MySQL server installation completes, it has a **ROOT** user account only to access and manage the databases. But, sometimes, you want to give the database access to others without granting them full control. In that case, you will create a non-root user and grant them specific privileges to access and modify the database.

**Syntax**

**CREATE** USER [IF NOT EXISTS] account\_name IDENTIFIED **BY** 'password';

In the above syntax, the **account\_name** has two parts one is the **username**, and another is the **hostname**, which is separated by **@** symbol. Here, the username is the name of the user, and the hostname is the name of the host from which the user can connect with the database server.

username@hostname

The hostname is optional. If you have not given the hostname, the user can connect from any host on the server. The user account name without hostname can be written as:

username@%

**Example:**

**create** user peter@localhost identified **by** 'jtp12345';

Execute the following command to show all users in the current MySQL server.

**select** user **from** mysql.user;

**CREATE ROLE**

A role is a set of privileges with name. You can create one or more roles using the CREATE ROLE statement. It's much easier to ensure that users with the same responsibilities have the same privileges using roles than it is to grant many different privileges directly. To create a role, you need to have CREATE ROLE or CREATE USER privilege.

Syntax:

CREATE ROLE '<role>'@'<host>';

the role name and the host that the client is connecting from.

You can also create multiple roles at the same time by separating each role name with a comma:

CREATE ROLE '<role\_1>'@'<host>', '<role\_2>'@'<host>', '<role\_3>'@'<host>';

If any of the roles you specify already exist on the system, the command will fail with an error.

To avoid this and cause MySQL to only issue a warning, you can include the IF NOT EXISTS clause after the CREATE ROLE command before the role names:

CREATE ROLE IF NOT EXISTS '<role>'@'<host>';

**GRANT PRIVILEGES TO A ROLE**

After creating new roles, your next priority is typically to make them meaningful by granting them privileges.

GRANT <privileges> ON <database>.<object> TO '<role>'@'<host>';

For example, to grant the SELECT privilege to a role called **readapp** on the **appdb** database and all objects it contains, you could type:

GRANT SELECT ON appdb.\* TO 'readapp';

Similarly, you can grant **write** privileges to the same database to a role called **writeapp** by typing:

GRANT SELECT,INSERT,UPDATE,DELETE ON appdb.\* TO 'writeapp';

**REVOKING A ROLE FROM A USER**

REVOKE command can remove privileges from a role.

The basic syntax used to remove a role from a user account looks like this:

REVOKE '<role>' FROM '<user>'@'<host>';

After executing a statement like this, the user will no longer have access to the privileges that were granted through role.

As an example, we can revoke the **writeapp** role from the **'appuser'@'localhost'** user account by typing:

REVOKE 'writeapp' FROM 'appuser'@'localhost';

If the user has been granted a privilege through other means however (either directly granted or granted through membership with a different role) they will still have access to that privilege. So if the 'appuser'@'localhost' user was also a member of the readapp role we granted earlier, they would still have SELECT privileges on the **appdb** database.

**CREATE INDEX**

<https://www.javatpoint.com/how-to-create-index-in-mysql>

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries. Indexes are data structures that store a subset of the data in a table, organized in a way that allows the database management system to quickly locate and retrieve rows that match specific criteria.

When a table is created with a primary key or unique key, it automatically creates a special index named **PRIMARY**.

Generally, we create an index at the time of table creation in the database. The following statement creates a table with an index that contains two columns col2 and col3.

**CREATE** **TABLE** t\_index(

col1 **INT** **PRIMARY** **KEY**,

   col2 **INT** NOT NULL,

   col3 **INT** NOT NULL,

   col4 **VARCHAR**(20),

**INDEX** (col2,col3)   );

CREATE INDEX : Creates an index on a table. Duplicate values are allowed:

CREATE INDEX *index\_name*  
ON *table\_name* (*column1*, *column2*, ...);

CREATE UNIQUE INDEX : Creates a unique index on a table. Duplicate values are not allowed:

CREATE UNIQUE INDEX *index\_name*  
ON *table\_name* (*column1*, *column2*, ...);

Example

The SQL statement below creates an index named "idx\_lastname" on the "LastName" column in the "Persons" table:

CREATE INDEX idx\_lastname  
ON Persons (LastName);

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

CREATE INDEX idx\_pname  
ON Persons (LastName, FirstName);

**DROP INDEX**

The DROP INDEX statement is used to delete an index in a table.

ALTER TABLE *table\_name* DROP INDEX index\_name;