## Lab Experiment 9

### Demonstration of DBA tasks in SQL server

### 

### Objectives

* Demonstration of creating views, server logins, database users and roles
* Enforcing policies, granting and revoking privileges for certain roles and users

### Introduction

Most of the following topics have already been discussed in the lectures. A basic overview of these topics and syntax is provided in the following section:

### Views

Views are virtual tables that do not store any data of their own but display data stored in other tables. In other words, VIEWS are nothing but SQL Queries. A view can contain all or a few rows from a table. A MySQL view can show data from one table or many tables.

Syntax: CREATE or ALTER VIEW [view\_name] AS SELECT statement with check option;

Example:

create or alter view [vw\_marvel]

as

select s.id , s.full\_name , s.superhero\_name from superhero s

join publisher p on s.publisher\_id=p.id

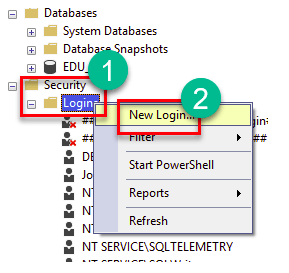
where p.publisher\_name like '%marvel%'

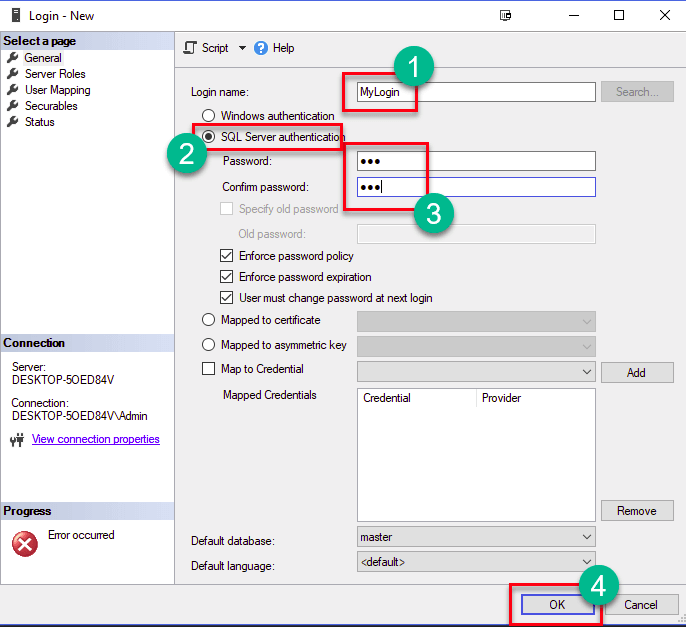
with check option;

### Create Login, User and Grant Permissions in SQL Server

There are two ways to create server’s logins, database users, roles and granting/revoking of permissions to roles/users. First way is by using SQL server management studio’s object explorer and the second method is by using transaction commands and the same tasks can be done by using T-SQL statements

Follow the below graphical tutorial to create using SQL Server management Studio:





A screenshot of a computer

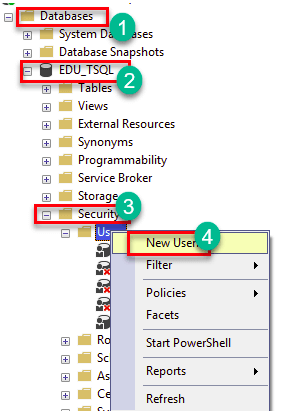
Description automatically generated

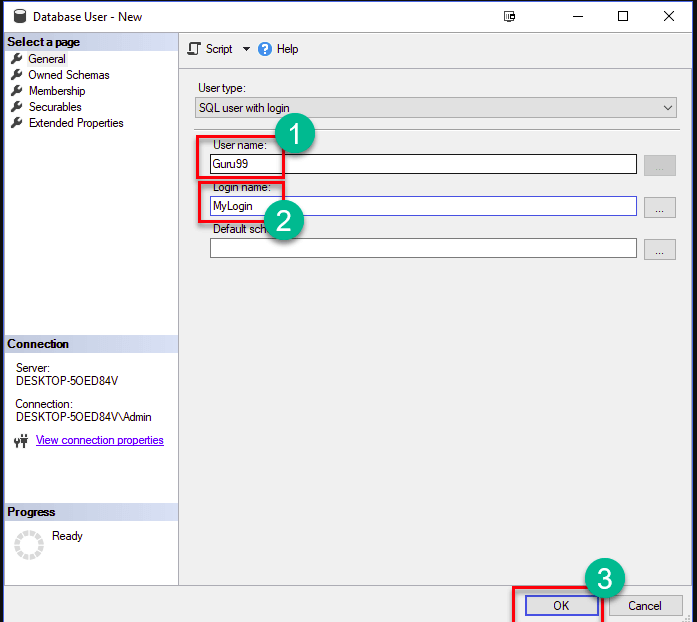
Now, by using the following T-SQL command, same thing can be done as:

*Create login testlogin with password =’123123’*

Now that login credentials for SQL server have been created, we need to create a user in database where we will be assigning roles and policies for that database user.

Following figures depict the method using SQL Server management studio:





A screenshot of a computer

Description automatically generated

By following the instructions in above figures, user will be created. The other method is by using T-SQL commands as follows:

*Create user testuser for login test*

### Roles:

In SQL, roles are security principals that group other principals and are used to manage permissions for a database or server:

* **Server-level roles**: Manage permissions for the entire SQL Server instance. You can create your own roles, but some fixed roles are built in and cannot be changed.
* **Database-level roles**: Manage permissions for the entire database. You can create your own roles, or use fixed roles that are predefined in the database.
* **Application-level roles**: Restrict access to data in your database.

Roles are useful when a database has many users because they provide a more efficient way to grant privileges than granting them to each user individually.

Here are some things you can do with roles in SQL:

* **Create roles**: Use the CREATE ROLE command.
* **Grant roles**: Use the GRANT command to assign roles to users. You can grant multiple roles to the same user.
* **Add and remove users to a role**: Use the ADD MEMBER and DROP MEMBER options of the ALTER ROLE statement.
* **Delete or modify a role**: In Object Explorer, expand the Security folder, then expand the Roles folder.
* **Check user roles**: Get a list of every object a user has access to, including permissions and how they were granted.

### Syntax:

*create role role\_name;*

*alter role role\_name add member db\_user;*

### Grant & Revoke Statements

To grant permissions to roles in SQL Server, you can use the GRANT command. The syntax for granting system permissions is:

* **GRANT system\_permission, ...**
* **TO ROLE 'role\_name', ...**
* **WITH GRANT OPTION/**

The grantor must have the permission being granted or a higher permission that implies it. Members of the sysadmin fixed server role can grant any permission.

Here are some other things to know about granting permissions in SQL Server:

* **Object permissions**: Control access to individual objects, such as tables.
* **System permissions**: Control the commands that can be run in the ObjectServer.
* **Grant With**: Indicates that the grantee will also be able to grant the specified permission to other principals.
* **REVOKE**: Can be used to remove granted permissions.
* **DENY**: Can be used to prevent a principal from gaining a specific permission through a GRANT.

Syntax:

GRANT privilege\_name

ON object\_name

TO {user\_name |PUBLIC |role\_name}

[WITH GRANT OPTION];

* ***privilege\_name*** is the access right or privilege granted to the user. Some of the access rights are select, update, and delete.
* ***object\_name*** is the name of an database object like TABLE or a VIEW
* ***user\_name*** is the name of the user to whom an access right is being granted.
* ***PUBLIC*** is used to grant access rights to all users.
* ***ROLES*** are a set of privileges grouped together.
* ***WITH GRANT OPTION*** - allows a user to grant access rights to other users.

Revoke statement is used to take the given privileges back from a user or a role and follows similar syntax.

### Privileges and Roles:

Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

**1) System privileges** - This allows the user to CREATE, ALTER, or DROP database objects.  
**2) Object privileges** - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

|  |  |
| --- | --- |
| **System Privileges** | **Description** |
| CREATE object | allows users to create the specified object in their own schema. |
| CREATE ANY object | allows users to create the specified object in any schema. |

**The above rules also apply for ALTER and DROP system privileges.**

Few of the object privileges are listed below:

|  |  |
| --- | --- |
| **Object Privileges** | **Description** |
| INSERT | allows users to insert rows into a table. |
| SELECT | allows users to select data from a database object. |
| UPDATE | allows user to update data in a table. |
| EXECUTE | allows user to execute a stored procedure or a function. |

### Task

Remember the database “Superhero” that we used earlier and create the following users/roles and assign appropriate privileges accordingly:

Create a user **JeanPaulJohn** who accesses the database as a MarvelSuperheroManager and should have rights to

* look at all the features/characteristics of superheroes whose publisher is Marvel
* add super heroes and all their features in the relevant tables whose comics are published by Marvel
* delete superheros of DC comics
* delegate the read privileges to other users

Create another user **RajnikanthPyarelal** who is a DC comics Manager and should have rights to

* look at all the features/characteristics of superheroes whose publisher is Marvel
* add super heroes and all their features in the relevant tables whose comics are published by Marvel
* delete superheros of DC comics

Create a user ‘**Alphonso’** who can see all the superheros and can only change the colour of their hair, skin and eyes. He should not have access to anything else.

### 

**Rubric for Lab Assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| **The student performance for the assigned task during the lab session was:** | | | |
| Excellent | The student completed assigned tasks without any help from the instructor and showed the results appropriately. | 4 |  |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 |  |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 |  |
| Worst | The student did not complete assigned tasks. | 1 |  |

**Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**