Lab 2: SQL DDL For DBA

Granting a user access to a database involves three steps. First, you create a login. The login lets the user connect to the SQL Server Database Engine. Then you configure the login as a user in the specified database. And finally, you grant that user permission to database objects. This lesson shows you these three steps, and shows you how to create a view and a stored procedure as the object.

This lab relies on Database Objects created in Lab 1

**NOTE : All your tasks are highlighted in Yellow**

## Create a login

To access the Database Engine, users require a login. The login can represent the user's identity as a Windows account or as a member of a Windows group, or the login can be a SQL Server login that exists only in SQL Server. Whenever possible you should use Windows Authentication.

By default, administrators on your computer have full access to SQL Server. For this lesson, we want to have a less privileged user; therefore, you will create a new local Windows Authentication account on your computer. To do this, you must be an administrator on your computer. Then you grant that new user access to SQL Server.

### Create a new Windows account

1. Select **Start**, select **Run**, in the **Open** box, type %SystemRoot%\system32\compmgmt.msc /s and then select **OK** to open the Computer Management program.
2. Under **System Tools**, expand **Local Users and Groups**, right-click **Users**, and then select **New User**.
3. In the **User name** box type **Mary**.
4. In the **Password** and **Confirm password** box, type a strong password, and then select **Create** to create a new local Windows user.

### Create a SQL login

In a Query Editor window of SQL Server Management Studio, type and execute the following code replacing computer\_name with the name of your computer. FROM WINDOWS indicates that Windows authenticates the user. The optional DEFAULT\_DATABASE argument connects Mary to the TestData database, unless the connection string indicates another database. This statement introduces the semicolon as an optional termination for a Transact-SQL statement.

CREATE LOGIN [computer\_name\Mary]

FROM WINDOWS

WITH DEFAULT\_DATABASE = [TestData];

GO

This authorizes a user name Mary, authenticated by your computer, to access this instance of SQL Server. If there is more than one instance of SQL Server on the computer, you must create the login on each instance that Mary must access.

**Note**

***Because Mary is not a domain account, this user name can only be authenticated on this computer.***

## Grant access to a database

The Mary account now has access to this instance of SQL Server, but doesn't have permission to access the databases. The account doesn't even have access to the default database TestData until you authorize the account as a database user.

To grant Mary access, switch to the TestData database, and then use the CREATE USER statement to map the login to a user named Mary.

### Create a user in a database

Type and execute the following statements (replacing computer\_name with the name of your computer) to grant Mary access to the TestData database.

USE [TestData];

GO

CREATE USER [Mary] FOR LOGIN [computer\_name\Mary];

GO

Now, Mary has access to both SQL Server and the TestData database.

## Create views and stored procedures

As an administrator, you can execute the SELECT from the Products table and the vw\_Names view, and execute the pr\_Names procedure; however, Mary can't. To grant Mary the necessary permissions, use the GRANT statement.

### Grant permission to stored procedure

Execute the following statement to give Mary the EXECUTE permission for the pr\_Names stored procedure.

GRANT EXECUTE ON pr\_Names TO Mary;

GO

In this scenario, Mary can only access the Products table by using the stored procedure. If you want Mary to be able to execute a SELECT statement against the view, then you must also execute GRANT SELECT ON vw\_Names TO Mary. To remove access to database objects, use the REVOKE statement.

**Note**

If the table, the view, and the stored procedure are not owned by the same schema, granting permissions becomes more complex.

### About GRANT

You must have EXECUTE permission to execute a stored procedure. You must have SELECT, INSERT, UPDATE, and DELETE permissions to access and change data. The GRANT statement is also used for other permissions, such as permission to create tables.

# Delete database objects

This short lesson removes the objects that you created in Lab 1 an Lab 2, and then drops the database.

Before you delete objects, make sure you are in the correct database:

USE TestData;

GO

## Revoke stored procedure permissions

Use the REVOKE statement to remove execute permission for Mary on the stored procedure:

REVOKE EXECUTE ON pr\_Names FROM Mary;

GO

## Drop permissions

1. Use the DROP statement to remove permission for Mary to access the TestData database:

DROP USER Mary;

GO

1. Use the DROP statement to remove permission for Mary to access this instance of SQL Server 2005 (9.x):

DROP LOGIN [<computer\_name>\Mary];

GO

1. Use the DROP statement to remove the store procedure pr\_Names:

DROP PROC pr\_Names;

GO

1. Use the DROP statement to remove the view vw\_Names:

DROP VIEW vw\_Names;

GO

## Delete table

1. Use the DELETE statement to remove all rows from the Products table:

DELETE FROM Products;

GO

1. Use the DROP statement to remove the Products table:

DROP TABLE Products;

GO

## Remove database

You can't remove the TestData database while you are in the database; therefore, first switch context to another database, and then use the DROP statement to remove the TestData database:

USE MASTER;

GO

DROP DATABASE TestData;

GO