





Lesson 2: Configure permissions on database objects



docs.microsoft.com/en-us/sql/t-sql/lesson-2-configuring-permissions-on-database-objects

- 07/29/2018
- 4 minutes to read
- Contributors

APPLIES TO:  SQL Server (starting with 2008)  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

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.

Note

This lesson relies on objects created in [Lesson 1 - Create database objects](#). Complete Lesson 1 before continuing on to lesson 2.

Prerequisites

To complete this tutorial, you need SQL Server Management Studio and access to a SQL Server instance.

Install [SQL Server Management Studio](#).

If you don't have access to a SQL Server instance, select your platform from the following links. If you choose SQL Authentication, use your SQL Server login credentials.

- **Windows:** [Download SQL Server 2017 Developer Edition](#).
- **macOS:** [Download SQL Server 2017 on Docker](#).

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 will grant that new user access to SQL Server.

Create a new Windows account

1. Click **Start**, click **Run**, in the **Open** box, type %SystemRoot%\system32\compmgmt.msc /s, and then click **OK** to open the Computer Management program.
2. Under **System Tools**, expand **Local Users and Groups**, right-click **Users**, and then click **New User**.
3. In the **User name** box type **Mary**.
4. In the **Password** and **Confirm password** box, type a strong password, and then click **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 will authenticate the user. The optional `DEFAULT_DATABASE` argument connects `Mary` to the `TestData` database, unless her connection string indicates another database. This statement introduces the semicolon as an optional termination for a Transact-SQL statement.

SQL

```
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

Mary now has access to this instance of SQL Server, but does not have permission to access the databases. She does not even have access to her default database **TestData** until you authorize her as a database user.

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

To 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.

SQL

```
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 cannot. 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.

SQL

```
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.

Next steps

The next article teaches you how to remove database objects you created in the other lessons.

Go to the next article to learn more:

Next steps