**Reference:** <https://msdn.microsoft.com/en-us/library/ms173463.aspx>

**Part I: This first portion will walk you through the SQL commands to setup users, schemas and permissions.**

**Scenario:** SQL Server simplifies the administration of SQL Server security by separating the implicit link between users and the database objects that they own. Earlier versions of SQL Server required that you first drop or reassign all database objects that the users owned, which significantly complicated the process. SQL Server includes an object permission model that addresses this issue.

By breaking the link between users and database objects, administration is simplified as follows:

* Users can be dropped without having to drop or reassign database objects.
* Users are associated with a default schema that is the owner of objects that the user creates.
* Schemas can be owned by roles, allowing multiple users to administer database objects without requiring database-wide permissions.

So, in SQL Server, users and schemas are two different things.A user is an identifier for the person using a database. A user can belong to roles, and can be granted permissions, either directly or through roles they belong to. A schema is a namespace that contains a set of objects and is owned by a user or a role. A user is never added to a schema. Schemas contain objects, not users. Users can be assigned a default schema that may or may not exist. A user’s default schema is used for name resolution.

In order to create an object in a schema, the following conditions must be satisfied:

* The schema must exist.
* The user creating the object must have CREATE TABLE permission, either directly or through role membership.
* The user creating the object must be the owner of the schema, must be a member of the role that owns the schema, must have ALTER rights on the schema, or must have the ALTER ANY SCHEMA permission in the database.

**Exercise:** In this exercise, you will add two new logins to SQL Server and add those logins to a role in your new database.

1. Create a new database named NewDB
2. Create two new logins.   
   * 1. Open a new query window and enter the following code.   
          
        USE master   
        GO   
        EXEC sp\_addlogin NewUser1, pa7$$wordabvvK, NewDB   
        EXEC sp\_addlogin NewUser2, pa7$$wordabvvK, NewDB   
        GO   
        EXEC sp\_helplogins   
        GO
     2. Press **F5**, or click the **Execute** button on the toolbar, to execute the query. If prompted, respond to the **Connect to SQL Server** dialog box.
     3. In the report created by sp\_helplogins, you should now have two new logins, each with a password of **pa7$$wordabvvK** and a default database of **NewDB**.
3. Add the new logins as database users.   
   * 1. This step will add both of your new logins to the NewDB database. The login NewUser1 will be given a default schema of DemoSchema. The login NewUser2 will not be given a specific default schema, so its default schema will be **dbo**.   
          
        USE NewDB   
        GO   
        CREATE USER NewUser1   
        WITH DEFAULT\_SCHEMA = DemoSchema   
        GO   
        CREATE USER NewUser2   
        GO

Select the code you just entered, and press **F5** or click the **Execute** button on the toolbar to execute the selected statements.   
  
Note that you can give a user a default schema, even though the schema has not been created yet. The default schema is the schema name that will automatically be assumed when a query is run, if a schema is not explicitly specified. The default schema applies to all DML and DDL statements: SELECT, INSERT, UPDATE and DELETE, as well as CREATE TABLE and ALTER TABLE.

1. Create a database role and add users to it.   
   * 1. In this step, you will create a database role named **DemoUsers** and you will create a schema of the same name, owned by that role. The two commands you will use are the equivalent of sp\_addrole. You’ll add both new users to that role.   
          
        USE NewDB   
        GO   
        CREATE ROLE DemoUsers   
        GO   
        CREATE SCHEMA DemoUsers AUTHORIZATION DemoUsers   
        GO   
        EXEC sp\_addrolemember DemoUsers, NewUser1   
        EXEC sp\_addrolemember DemoUsers, NewUser2   
        EXEC sp\_addrolemember db\_datareader, DemoUsers   
        GO
     2. Select the code you just entered, and press **F5**, or click the **Execute** button on the toolbar, to execute the selected statements.   
          
        Note that you can make a role a member of another role. In this case, you made the user-defined role DemoUsers a member of the predefined database role db\_datareader.
2. Create a schema and a table owned by the schema.  
   * 1. In this step, you will add a schema owned by the role **DemoUsers**. It will verify that the schema was created by looking in the schemas table.

USE NewDB   
GO   
CREATE SCHEMA DemoSchema AUTHORIZATION DemoUsers  
GO   
SELECT \* FROM sys.schemas   
GO

* + 1. Enter the code above, and press **F5** or click the **Execute** button on the toolbar to execute the selected statements. In your results, you should see all of the predefined database roles, as well as any user-defined roles created with sp\_addrole, which automatically creates a corresponding schema. You’ll see **guest** and **INFORMATION\_SCHEMA,** which were users in SQL Server 2000. You should see your newly defined schemas **DemoUsers** and **DemoSchema** and a system schema called **sys**. There is also a schema dbo, as well as a user dbo that you can see if you run sp\_helpuser.   
         
       In SQL Server, all system tables are in a hidden resource database but are visible through the sys schema, which is a logical schema available in every database.

1. In this step, you create two tables and insert a row of identifying data into each one. The tables will have the same name, but one will be in the DemoSchema schema and the other will be in the dbo (built-in) schema. Enter the code below, and press **F5** or click the **Execute** button on the toolbar to execute the selected statements.  
     
   USE NewDB   
   GO   
   CREATE TABLE DemoSchema.DemoTable   
     (version varchar(20) )   
   GO   
     
   INSERT INTO DemoSchema.DemoTable SELECT 'DemoSchema schema'   
   GO   
     
   CREATE TABLE dbo.DemoTable   
     (version varchar(20) )   
   GO   
     
   INSERT INTO dbo.DemoTable SELECT 'DBO schema'   
   GO
2. Grant permissions to a role.   
   * 1. In this step, you’ll grant permission to the **DemoUsers** role to create new tables. Enter and execute the following:

USE NewDB   
GO  
GRANT CREATE TABLE to DemoUsers   
GO

1. Test access after logging in as different users.   
   * 1. Here, we see what happens when a user has a default schema defined.
     2. Disconnect from the database, if necessary: select the **Query** | **Connection** | **Disconnect** menu item.
     3. Now reconnect to the database using SQL Server Authentication (not Windows Authentication), and login as **NewUser1** for the user name, and pa7$$wordabvvK for the password.
     4. Now run this code:

USE NewDB   
GO   
SELECT \* FROM DemoTable   
GO   
CREATE TABLE DemoTable1 (message varchar(30))   
INSERT INTO DemoTable1 SELECT 'Created by NewUser1'   
GO   
EXEC sp\_help DemoTable1   
GO  
  
Because the user NewUser1 was given the default schema of DemoSchema, SQL Server will automatically look first for an object called DemoTable in DemoSchema when resolving the SELECT statement. SQL Server will also use DemoSchema as the owner of the new table DemoTable1, as you can see when you execute sp\_help DemoTable1.

* + 1. Disconnect again: select the **Query** | **Connection** | **Disconnect** menu item.
    2. We will see what happens when a default schema was not specified when the user was created. Make sure you are not connected to the database.
    3. Using **SQL Server Authentication login**. Enter **NewUser2** for the user name, and pa7$$wordabvvK for the password. Click **Connect**.
    4. Now run this script:

USE NewDB   
SELECT \* FROM DemoTable   
GO   
CREATE TABLE DemoTable2   
(message varchar(30))   
INSERT INTO DemoTable2 SELECT 'Created by NewUser2'   
GO

You did not specify a default schema when you created the NewUser2 user, so dbo is used as the default schema. Because NewUser2 is a member of the DemoUsers role, which in turn is a member of the db\_datareader role, NewUser2 can read from the DemoTable in the dbo schema. However, NewUser2 does not have permission to create a table in the dbo schema, so an error message is generated.SQL Server will always first check the default schema defined for a user when accessing an object, and then it will check for an object in the dbo schema.  
**Part 2:** In an MS Word document, describe how you would apply the concepts covered above to the database you are creating for the MRP project. Explain why you believe applying specific schemas beyond the default “dbo” could be of a benefit to the physical implementation of a database. What are some roles (potential types of users) you might define for your MRP database? Explain why? Explain the permissions each role would have. Then on your own, conduct research using our text or Google to answer the following:

1. For an existing table, how would you change its schema using SQL?
2. Can you Revoke a Deny?
3. How is it possible for a user to create or drop tables among multiple schemas? Provide an example using SQL commands.