SQL Server™ 2005:
Introduction to SQL Server Management Studio

Table of Contents

SQL Server 2005: Introduction to SQL Server Management Studio

Objectives

After completing this lab, you will be able to:

- Use the new SQL Server Management Studio to perform basic SQL Server administration tasks.

NOTE: This lab focuses on the concepts in this module and as a result may not comply with Microsoft® security recommendations.

NOTE: The SQL Server 2005 labs are based on beta builds of the product. The intent of these labs is to provide you with a general feel of some of the planned features for the next release of SQL Server. As with all software development projects, the final version may differ from beta builds in both features and user interface. For the latest details on SQL Server 2005, please visit http://www.microsoft.com/sql/2005/.

Scenario

SQL Server Workbench is a new tool built for the 2005 release of SQL Server. It combines the functionality of the Enterprise Manager snap-in and the Query Analyzer. Although this is the main tool for administering one or more SQL Servers, you can also use the SQL Server Workbench for executing queries and scripts, and for managing SQL Server projects. This lab will show you how to perform basic administration tasks on SQL Server 2005 using SQL Server Workbench.

Prerequisites

- Basic experience with SQL Server 2000 administration.
- An understanding of basic Microsoft Windows® networking concepts.
- A basic understanding of SQL Server security.
- Some experience with T-SQL.

Estimated Time to Complete This Lab

45 Minutes

Lab Setup

Tasks	Detailed Steps
1. Log in.	**1.** Log in using the **Administrator** user account. The password is **Pass@word1**.

Exercise 1
Exploring the Environment

Scenario

In this exercise, you will become familiar with the new SQL Server 2005 administration tool, called SQL Server Management Studio.

SQL Server Management Studio is a new tool built for SQL Server 2005. It combines the functionality of the Enterprise Manager snap-in and the Query Analyzer. Although this is the main tool for administering one or more SQL Servers, you can also use the SQL Server Management Studio for executing queries and scripts, and for managing SQL Server projects.

The SQL Server Management Studio tool is based on the Microsoft Development Environment used in Microsoft Visual Studio® 2005 to create applications. If you are not already familiar with Visual Studio 2005, using the SQL Server Management Studio tool will help you learn to use the new Microsoft Development Environment for Visual Studio 2005.

Tasks	Detailed Steps
1. Open SQL Server Management Studio and connect to your server.	**1.** From the Windows task bar, select **Start | All Programs | Microsoft SQL Server 2005 | SQL Server Management Studio**. **2.** When the **Connect to Server** dialog box opens, verify that **SQL Server** is selected as the **Server type,** verify that **Server name** is same name as the host machine or set it to **localhost**, and verify that **Windows Authentication** is selected as the authentication method. **3.** Click the **Options** button to display additional connection options. **4.** Click the **Connection Properties** tab. Note that the following options are available: ▪ You can configure the network protocol to use for this connection, which might be different than the protocol you use for other connections. ▪ You can configure a connection timeout, which controls how long to wait for the connection to be made. ▪ You can configure an execution timeout to specify how long to wait for response from a query. **5.** Examine the options available on the **Registered Servers** tab. Note that you can export your server registration information, or import registration information from another server. This facility can be valuable for large organizations with many SQL Server administrators who all want to have the same servers registered. **6.** Click **Options** again to hide the additional options tabs. **7.** Click **Connect**. Note the various areas of the SQL Server Management Studio: The lower left pane is the Object Explorer, which appears as a tree view on the left side of SQL Server Management Studio. Above that is the Registered Servers pane, containing a list of servers to which Management Studio can connect. The right side of the SQL Server Management Studio contains the tools for managing projects. On the top right is the **Solution Explorer**. Below that is the **Properties Window**. If the Solution Explorer is not visible, you can choose to display it by selecting **View | Solution Explorer**. If the **Properties** window is not visible, you can also enable that window from the View menu, or by right-clicking on any object in the **Solution Explorer** window and choosing **Properties Window**. At the moment,

Tasks	Detailed Steps
	the Solution Explorer is blank because no solution is currently loaded. The location of all of these windows and browsers can be changed, and you can remove the ones you don't want or need. You can undock and move windows by double-clicking on the window's title bar. You can reset the window layout by select **Window** | **Reset Window Layout**. If you close any of the windows, you can restore them from the View menu.
2. Use TSQL to execute a stored procedure.	**1.** Click the leftmost button (New Query) on the standard toolbar, and then select **Database Engine Query** to open a SQL Query Window. When asked to connect to the server, click **Connect**. **2.** The query window appears in the center portion of the SQL Server Management Studio, and is tabbed with the Summary Page. All additional query windows will be tabbed as well. **3.** In the query window, enter the following code: `EXEC sp_who` **4.** Press **F5** or click the **Execute** button on the toolbar to execute the query. You can disconnect from SQL Server without closing the query window by using the **Query | Connection | Disconnect** menu option. Alternatively, you can add a **Disconnect** or **Disconnect All** button to the toolbar. Right-click on the toolbar and choose **Customize**. Go to the **Commands** tab and select **Query** in the leftmost list box. You can then drag any command from the rightmost list onto an existing toolbar. Click **Close** to dismiss this dialog box.
3. Open and use a script file from a project.	The scripts you'll need to run for the remainder of this lab have all been written for you. To access these scripts, click **File | Open | Project/Solution**, and select **C:\SQL Labs\Lab Projects\Administration Lab\Administration Lab.ssmssln**. If the system prompts you to save changes as you load the solution, select **No**. **1.** You should see the Administration Lab solution in the Solution Explorer window on right side of the Management Studio. There are five projects, one for each of the five exercises in this lab. Under Exercise 1, expand the Queries folder, and double-click the script called ServerProperties.sql. If prompted, reconfirm your server name and authentication mode in the **Connect to SQL Server** dialog box. `USE master` `SELECT SERVERPROPERTY('ServerName')` `SELECT SERVERPROPERTY('Edition')` `SELECT SERVERPROPERTY('ProductVersion')` `SELECT SERVERPROPERTY('ProductLevel')` **2.** To execute the batch, press **F5** or click the **Execute** button on the toolbar. Note the edition and version number of the SQL Server you are connected to. You may have to scroll down to see all of the results.

Exercise 2
Attaching a Database

Scenario

In this exercise, you will attach the AdventureWorks database using the "Attach Database" option in the SQL Server Management Studio **Object Explorer**.

Attaching a database means making all the database files available to your SQL Server, just as if you had created the database on the current server. Detaching a database allows you to move the physical files and reattach those files from a new physical location, or to make a copy of the files to use on another server (perhaps a test or development server).

Because the primary file contains the locations of all the other files in a database, if all the files are in their original location (as stored in the primary file) you only need to specify the primary file when attaching the database. SQL Server will read the information in the primary file and attach all the associated files for the database.

However, if the files are not all in the original location, the information stored in the primary file may not be sufficient for SQL Server to find and attach all the database files. You will then need to specify the exact physical location of each database file that is not in the location it was when the database was detached.

When a database is detached, SQL Server will do a checkpoint in the database, so all the committed transactions are written to the disk files.

Tasks	Detailed Steps
1. If the database is already attached, detach it.	**1.** In the Object Explorer, expand **localhost** or the name of your server (if it's not already expanded), then expand the **Databases** folders. **2.** If the **AdventureWorks** database is in the list, right-click on it. Otherwise, skip to the **Attach the AdventureWorks database** task below. **3.** Point to **Tasks**, and then click **Detach**. **4.** In the **Detach Database** dialog box, click **OK**.
2. Attach the AdventureWorks database.	**1.** In the Object Explorer, right-click on the **Databases** folder, and click **Attach.** **2.** In the Attach Database(s) dialog box, click **Add**. **3.** Locate and select the following master file for the AdventureWorks database (**C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\AdventureWorks_Data.mdf**), and click **OK**. Note that the Attach Database dialog box is not modal. This means that you can go back to the Management Studio and look for other information you might require while filling in the dialog box. **4.** Verify that there are two files listed in the **AdventureWorks database details** section in the bottom half of the **Attach Database(s)** dialog box. **5.** Click **OK** to attach the database.
3. Use the Object Explorer to Verify the AdventureWorks Database.	**1.** Expand the **Databases** folder in the **Object Explorer**. **2.** If the AdventureWorks database doesn't appear, right-click the **Databases** folder and select **Refresh**. **3.** Expand the AdventureWorks database. **4.** Expand the **Tables** folder, and verify that there are several dozen tables. **5.** In the **Solution Explorer**, open the Queries folder under Exercise 2, and double click the file DatabaseProperties.sql.. If requested, respond to the **Connect to SQL Server** dialog box. `USE AdventureWorks`

Tasks	Detailed Steps
	`` SELECT DATABASEPROPERTYEX('AdventureWorks', 'Status') SELECT DATABASEPROPERTYEX('AdventureWorks', 'Recovery') SELECT DATABASEPROPERTYEX('AdventureWorks', 'Collation') SELECT DATABASEPROPERTYEX('AdventureWorks', 'Updateability') SELECT DATABASEPROPERTYEX('AdventureWorks', 'UserAccess') SELECT DATABASEPROPERTYEX('AdventureWorks', 'IsAutoCreateStatistics') SELECT DATABASEPROPERTYEX('AdventureWorks', 'IsAutoShrink') ``` **6.** Press **F5** or click the **Execute** button on the toolbar to execute the batch **7.** Examine the values returned for the various database properties.

Exercise 3
Exploring an Existing Database

Scenario

In this exercise, you will use the options available in the SQL Server Management Studio to discover table and procedure definitions. You will also use the Management Studio's query tool to execute basic SQL commands to examine table data.

Tasks	Detailed Steps
1. Examine the properties of the AdventureWorks database.	**1.** Expand the **Databases** folder in the Object Explorer window and right-click on the AdventureWorks database. **2.** Click on **Properties**. Note that the **Database Properties** dialog box is non-modal and has its own button in the taskbar. You saw the value for several properties in Exercise 2 when you executed the DatabaseProperties.sql script. **3.** Click on each of the categories in the left pane to see what properties are read-only and which are updateable. **4.** Click **Cancel** to close the **Database Properties** window.
2. Write a query to examine some data. NOTE: Even though the GO is not necessary for SQL Server to execute the queries, or for you to select and run individual batches, using GO is the only way that the query tool knows how to separate the batches.	**1.** In the Solution Explorer window, expand the Exercise 3 node, and double-click the **GroupBy.sql** file. If requested, respond to the **Connect to SQL Server** dialog box. This adds the following SQL to the query window: `USE AdventureWorks` `GO` `SELECT  City, StateProvinceID, CustomerCount = count(*)` `FROM Person.Address` `GROUP BY City, StateProvinceID` `HAVING count(*) > 1` `ORDER BY count(*) DESC` `GO` **2.** Select the query you just entered, and then press **F5** or click the **Execute** button on the toolbar to execute the query.

Exercise 4
Creating a New Database

Scenario

In this exercise, you will create a new database using a SQL Server Management Studio **template**. You will then investigate the default properties of a new database.

Tasks	Detailed Steps
1. Create a new database using a TSQL Template.	**1.** Choose **File | New | File.** **2.** In the **New File** dialog box, in the **Categories** list, within the **SQL Server Query** node, select **Database**. In the **Templates** list, click **create database**, and then click **Open**. If prompted, connect to the database. A new query window will open, and you should see the following template: `-- ============================================` `-- Create database template` `-- ============================================` `IF EXISTS (` `  SELECT *` `  FROM  sys.databases` `  WHERE  name = N'<database_name, sysname, sample_database>'` `)` `  DROP DATABASE <database_name, sysname, sample_database>` `GO` `CREATE DATABASE <database_name, sysname, sample_database>` `GO` **3.** Above the line that starts with IF EXISTS, add the following: `USE MASTER` `GO` **4.** Select the **Edit** | **Find and Replace | Quick Replace** menu command. **5.** In the **Find what** textbox, enter **<database_name, sysname, sample_database>** (including the angle brackets). Note that the Find and Replace dialog is not modal, so you can copy this text from the Query window and paste it into the Find what textbox. **6.** In the **Replace with** textbox, enter **NewDB**. **7.** Click **Replace All**. A dialog box should show **3 occurrence(s) replaced**. Click **OK**, and then close the dialog box. **8.** Press **F5** or click the **Execute** button on the toolbar**.** Click **Connect** if necessary.
2. Explore your new database.	**1.** In the Object Browser, expand **Databases** if necessary. **2.** Right-click on **Databases** and click **Refresh**. A NewDB folder should appear.

Tasks	Detailed Steps
	3. Right-click on the **NewDB** folder and click **Properties.** **4.** Explore the properties dialog box to see what properties exist for a newly created database. Because the **Properties** window is not modal, you can open a separate Properties window for each database, making it easier to compare the two. **5.** Close the **Properties** window(s) when you're done.

Exercise 5
Setting up Database Access

Scenario

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

SQL Server 2005 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 2005 includes a new 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.

Tasks	Detailed Steps
1. Create two new logins.	**1.** Open a new query window and enter the following code. Alternatively, you can open the SQL script **Logins.sql** under Exercise 5 in the **Solution Explorer.** `USE master` `GO` `EXEC sp_addlogin  NewUser1, pa$$word, NewDB` `EXEC sp_addlogin NewUser2, pa$$word, 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.** You should now have two new logins, each with a password of 'pa$$word' and a default database of NewDB.
2. 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**. This code is in the SQL script DBUsers.sql under Exercise 5 in the Solution Explorer. Load the **DbUsers.sql** script. `USE NewDB` `GO` `CREATE USER NewUser1` `WITH DEFAULT_SCHEMA = DemoSchema` `GO` `CREATE USER NewUser2` `GO`

Tasks	Detailed Steps
	2. Press **F5** or click the **Execute** button on the toolbar to execute the query. In SQL Server 2005, you use CREATE USER to map a login to a database user instead of sp_grantdbaccess. Optionally, you can specify a login name using this syntax: `CREATE USER <user_name>` `[FOR LOGIN <Login_name>]` `[WITH DEFAULT_SCHEMA schema_name]` If no login name is specified, then the user is associated with the login of the same name as that of the <user_name>. If no such login exists the CREATE USER fails. (However, if the name specified were interpreted to be a Windows login in the form DOMAIN\loginname, the CREATE USER would succeed.) The <login_name> can be a Windows login, a Windows group or a SQL Login. 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.
3. 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. This code is in the SQL script **RolesAndSchema.sql** under Exercise 5 in the Solution Explorer. Load this code. `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.** Press **F5,** or click the **Execute** button on the toolbar**,** to execute the query. 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.

Tasks	Detailed Steps
4. Create a schema and a table owned by the schema.	**1.** In this step, you will add a schema owned by the role DemoUsers. Verify that the schema was created by looking in the schemas table. This code is in the SQL script **Schemas.sql** under Exercise 5 in the Solution Explorer. Load this code: `USE NewDB` `GO` `CREATE SCHEMA DemoSchema AUTHORIZATION DemoUsers` `GO` `SELECT * FROM sys.schemas` `GO` **2.** Press **F5,** or click the **Execute** button on the toolbar**,** to execute the query. 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 2005, 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 **3.** 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. This code is in the SQL script **NewTable.sql** under Exercise 5 in the Solution Explorer. Load this code: `USE NewDB` `GO` `CREATE TABLE DemoSchema.DemoTable` `(version varchar(20) )` `INSERT INTO DemoSchema.DemoTable SELECT 'DemoSchema schema'` `GO` `CREATE TABLE dbo.DemoTable` `(version varchar(20) )` `INSERT INTO dbo.DemoTable SELECT 'DBO schema'` `GO` **4.** Press **F5,** or click the **Execute** button on the toolbar**,** to execute the query..
5. Grant permissions to a role.	**1.** In this step, you'll grant permission to the DemoUsers role to create new tables. This code is in the SQL script **Permission.sql** under Exercise 5 in the Solution Explorer. Load this code: `USE NewDB` `GO` `GRANT CREATE TABLE to DemoUsers` `GO`

Tasks	Detailed Steps
	2. Press **F5,** or click the **Execute** button on the toolbar**,** to execute the query..
6. Test access after logging in as different users.	**1.** To see what happens when a user has a default schema defined, open the following script, which can be found in the SQL script **UserDemo1.sql** under Exercise 5 in the Solution Explorer. `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` **2.** Disconnect from the database, if necessary: select the **Query | Connection | Disconnect** menu item. (If you haven't connected yet, this item will be unavailable.) **3.** Press **F5** or click the Execute button on the toolbar to execute the script. When you are prompted to supply your connection information, choose **SQL Server Authentication**. Enter **NewUser1** for the user name, and **pa$$word** for the password. Click **Connect**. Because the user NewUser1 was given the default schema of DemoSchema, SQL Server will automatically look first for an object called DemoTable in DemoSchema when resolving the SELECT statement. SQL Server will also use DemoSchema as the owner of the new table DemoTable1, as you can see when you execute sp_help DemoTable1. You can change the connection information for any open query using the **Query \| Connection \| Change Connection** menu item. **4.** Disconnect again: select the **Query \| Connection \| Disconnect** menu item. **5.** To see what happens when a default schema was not specified when the user was created, open the following script. It can be found in the SQL script **UserDemo2.sql** under Exercise 5 in the Solution Explorer. `USE NewDB` `SELECT * FROM DemoTable` `GO` `CREATE TABLE DemoTable2` `(message varchar(30))` `INSERT INTO DemoTable2 SELECT 'Created by NewUser2'` `GO` **6.** Press **F5** or click the Execute button on the toolbar to execute the script. When you are prompted to supply your connection information, choose **SQL Server Authentication**. Enter **NewUser2** for the user name, and "**pa$$word**" for the password. Click **Connect**. 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

Tasks	Detailed Steps
	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. In previous versions of SQL Server, users and schemas were treated as synonymous. For backward compatibility, if you use the system stored procedure sp_grantdbaccess, SQL Server 2005 will create both a user and a schema of the same name, and the schema will be a user's default schema. This means that sp_ grantdbaccess NewUser is equivalent to the following code. `CREATE USER NewUser` `WITH DEFAULT SCHEMA NewUser` `CREATE SCHEMA NewUser AUTHORIZATION NewUser` In SQL Server 2005, 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.