Chapter 3

SQL Server Management Studio

In This Chapter

- Introduction to SQL Server Management Studio
- Using SQL Server Management Studio with the Database Engine
- Authoring Activities Using SQL Server Management Studio
This chapter first introduces SQL Server Management Studio, including how to connect it to a server, its Registered Servers and Object Explorer components, and its various user interface panes. After that, the chapter explains in detail the SQL Server Management Studio functions related to the Database Engine, including administering and managing databases, which you need to understand to be able to create and execute any Transact-SQL statements. Finally, the chapter covers using Query Editor, Solution Explorer, and the debugging tool to perform authoring activities in SQL Server Management Studio.

Introduction to SQL Server Management Studio

SQL Server 2012 provides various tools that are used for different purposes, such as system installation, configuration, auditing, and performance tuning. (All these tools will be discussed in different chapters of this book.) The administrator's primary tool for interacting with the system is SQL Server Management Studio. Both administrators and end users can use this tool to administer multiple servers, develop databases, and replicate data, among other things.

**NOTE**

*This chapter is dedicated to the activities of the end user. Therefore, only the functionality of SQL Server Management Studio with respect to the creation of database objects using the Database Engine is described in detail. All administrative tasks and all tasks related to Analysis Services and other components that this tool supports are discussed beginning in Part III.*

To open SQL Server Management Studio, choose Start | All Programs | Microsoft SQL Server 2012 | SQL Server Management Studio.

SQL Server Management Studio comprises several different components that are used for the authoring, administration, and management of the overall system. The following are the main components used for these tasks:

- Registered Servers
- Object Explorer
- Query Editor
- Solution Explorer

The first two components in the list are discussed in this section. Query Editor, and Solution Explorer are explained later in this chapter, in the section “Authoring Activities Using SQL Server Management Studio.”
To get to the main SQL Server Management Studio interface, you first must connect to a server, as described next.

**Connecting to a Server**

When you open SQL Server Management Studio, it displays the Connect to Server dialog box (see Figure 3-1), which allows you to specify the necessary parameters to connect to a server:

- **Server Type**  For purposes of this chapter, choose Database Engine.

**NOTE**

With SQL Server Management Studio, you can manage objects of the Database Engine and Analysis Server, among others. This chapter demonstrates the use of SQL Server Management Studio only with the Database Engine.

- **Server Name**  Select or type the name of the server that you want to use. (Generally, you can connect SQL Server Management Studio to any of the installed products on a particular server.)

![Figure 3-1  The Connect to Server dialog box](image)
Authentication  Choose between the two authentication types:

- **Windows Authentication**  Connect to SQL Server using your Windows account. This option is much simpler and is recommended by Microsoft.

- **SQL Server Authentication**  The Database Engine uses its own authentication.

**NOTE**
For more information concerning SQL Server Authentication, see Chapter 12.

When you click Connect, the Database Engine connects to the specified server. After connecting to the database server, the default SQL Server Management Studio window appears. The default appearance is similar to Visual Studio, so users can leverage their experience of developing in Visual Studio to use SQL Server Management Studio more easily. Figure 3-2 shows the SQL Server Management Window with several panes.

**NOTE**
SQL Server Management Studio gives you a unique interface to manage servers and create queries across all SQL Server components. This means that SQL Server Management Studio offers one interface for the Database Engine, Analysis Services, Integration Services, and Reporting Services.

**Registered Servers**
Registered Servers is represented as a pane that allows you to maintain connections to already used servers (see Figure 3-2). (If the Registered Servers pane isn’t visible, select its name from the View menu.) You can use these connections to check a server’s status or to manage its objects. Each user has a separate list of registered servers, which is stored locally.

You can add new servers to the list of all servers, or remove one or more existing servers from the list. You also can group existing servers into server groups. Each group should contain the servers that belong together logically. You can also group servers by server type, such as Database Engine, Analysis Services, Reporting Services, and Integration Services.
Chapter 3: SQL Server Management Studio

Object Explorer

The Object Explorer pane contains a tree view of all the database objects in a server. (If the Object Explorer pane isn’t visible, select View | Object Explorer.) The tree view shows you a hierarchy of the objects on a server. Hence, if you expand a tree, the logical structure of a corresponding server will be shown.

Object Explorer allows you to connect to multiple servers in the same pane. The server can be any of the existing servers for Database Engine, Analysis Services, Reporting Services, or Integration Services. This feature is user-friendly, because it allows you to manage all servers of the same or different types from one place.

NOTE

Object Explorer has several other features, explained later in this chapter.
Organizing and Navigating SQL Server Management Studio’s Panes

You can dock or hide each of the panes of SQL Server Management Studio. By right-clicking the title bar at the top of the corresponding pane, you can choose between the following presentation possibilities:

- **Floating**   The pane becomes a separate floating pane on top of the rest of SQL Server Management Studio panes. Such a pane can be moved anywhere around the screen.

- **Dockable**   Enables you to move and dock the pane in different positions. To move the pane to a different docking position, click and drag its title bar and drop it in the new position.

- **Tabbed Document** You can create a tabbed grouping using the Designer window. When this is done, the pane’s state changes from dockable to tabbed document.

- **Hide**   Closes the pane. (Alternatively, you can click the × in the upper-right corner of the pane.) To display a closed pane, select its name from the View menu.

- **Auto Hide**   Minimizes the pane and stores it on the left side of the screen. To reopen (maximize) such a pane, move your mouse over the tabs on the left side of the screen and click the push pin to pin the pane in the open position.

**NOTE**

The difference between the Hide and Auto Hide options is that the former option removes the pane from view in SQL Server Management Studio, while the latter collapses the pane to the side panel.

To restore the default configuration, choose Window | Reset Window Layout. The Object Explorer pane appears on the left, while the Object Explorer Details tab appears on the right side of SQL Server Management Studio. (The Object Explorer Details tab displays information about the currently selected node of Object Explorer.)

**NOTE**

You will find that often there are several ways of accomplishing the same task within SQL Server Management Studio. This chapter will indicate more than one way to do things, whereas only a single method will be given in subsequent chapters. Different people prefer different methods (some like to double-click, some like to click the +/- signs, some like to right-click, others like to use the pull-down menus, and others like to use the keyboard shortcuts as much as possible). Experiment with the different ways to navigate, and use the methods that feel most natural to you.
Within the Object Explorer and Registered Servers panes, a subobject appears only if you click the plus (+) sign of its direct predecessor in the tree hierarchy. To see the properties of an object, right-click the object and choose Properties. A minus (−) sign to the left of an object's name indicates that the object is currently expanded. To compress all subobjects of an object, click its minus sign. (Another possibility would be to double-click the folder, or press the LEFT ARROW key while the folder is selected.)

Using SQL Server Management Studio with the Database Engine

SQL Server Management Studio has two main purposes:

- Administration of the database servers
- Management of database objects

The following sections describe these functions of SQL Server Management Studio.

Administering Database Servers

The administrative tasks that you can perform by using SQL Server Management Studio are, among others, the following:

- Register servers
- Connect to a server
- Create new server groups
- Manage multiple servers
- Start and stop servers

The following subsections describe these administrative tasks.

Registering Servers

SQL Server Management Studio separates the activities of registering servers and exploring databases and their objects. (Both of these activities can be done using Object Explorer.) Every server (local or remote) must be registered before you can use its databases and objects. A server can be registered during the first execution of SQL Server Management Studio or later. To register a database server, right-click the folder of your database server in Object Explorer and choose Register. (If the Object Explorer pane doesn't appear on your screen, select View | Object Explorer.) The New Server
Registration dialog box appears, as shown in Figure 3-3. Choose the name of the server that you want to register and the authentication mode (Windows Authentication or SQL Server Authentication). Click Save.

**Connecting to a Server**

SQL Server Management Studio also separates the tasks of registering a server and connecting to a server. This means that registering a server does not automatically connect you to the server. To connect to a server from the Object Explorer window, right-click the server name and choose Connect.

**Creating a New Server Group**

To create a new server group in the Registered Servers pane, right-click Local Server Groups and choose New Server Group. In the New Server Group properties dialog box, enter a (unique) group name and optionally describe the new group.
Managing Multiple Servers
SQL Server Management Studio allows you to administer multiple database servers (called instances) on one computer by using Object Explorer. Each instance of the Database Engine has its own set of database objects (system and user databases) that are not shared between different instances.

To manage a server and its configuration, right-click the server name in Object Explorer and choose Properties. The Server Properties dialog box that opens contains several different pages, such as General, Security, and Permissions.

The General page (see Figure 3-4) shows general properties of the server. The Security page contains the information concerning the authentication mode of the server.

![Figure 3-4 The Server Properties dialog box: the General page](image)
server and the login auditing mode. The Permissions page shows all logins and roles that can access the server. The lower part of the page shows all permissions that can be granted to the logins and roles.

You can replace the existing server name with a new name. Right-click the server in the Object Explorer window and choose Register. Now you can rename the server and modify the existing server description in the Registered Server frame.

**NOTE**

*Do not rename servers, because changing names can affect other servers that reference them.*

### Starting and Stopping Servers

A Database Engine server starts automatically by default each time the Windows operating system starts. To start the server using SQL Server Management Studio, right-click the selected server in the Object Explorer pane and click Start in the context menu. The menu also contains Stop and Pause functions that you can use to stop or pause the activated server, respectively.

### Managing Databases Using Object Explorer

The following are the management tasks that you can perform by using SQL Server Management Studio:

- Create databases without using Transact-SQL
- Modify databases without using Transact-SQL
- Manage tables without using Transact-SQL
- Generate and execute SQL statements (will be described later, in the section “Query Editor”)

### Creating Databases Without Using Transact-SQL

You can create a new database by using Object Explorer or the Transact-SQL language. (Database creation using Transact-SQL is discussed in Chapter 5.) As the name suggests, you also use Object Explorer to explore the objects within a server. From the Object Explorer pane, you can inspect all the objects within a server and manage your server and databases. The existing tree contains, among other folders, the Databases folder. This folder has several subfolders, including one for the system databases and one for each new database that is created by a user. (System and user databases are discussed in detail in Chapter 15.)
Chapter 3: SQL Server Management Studio

To create a database using Object Explorer, right-click Databases and select New Database. In the New Database dialog box (see Figure 3-5), type the name of the new database in the Database Name field and then click OK. Each database has several different properties, such as file type, initial size, and so on. Database properties can be selected from the left pane of the New Database dialog box. There are several different pages (property groups):

- General
- Files

![Figure 3-5 The New Database dialog box](image)
52 Microsoft SQL Server 2012: A Beginner’s Guide

- Filegroups
- Options
- Change Tracking
- Permissions
- Extended Properties
- Mirroring
- Transaction Log Shipping

**NOTE**

For an existing database, the system displays all property groups in the preceding list. For a new database, as shown in Figure 3-5, there are only three groups: General, Options, and Filegroups.

The General page of the Database Properties dialog box (see Figure 3-6) displays, among other things, the database name, the owner of the database and its collation. The properties of the data files that belong to a particular database are listed in the Files page and comprise the name and initial size of the file, where the database will be stored, and the type of the file (PRIMARY, for instance). A database can be stored in multiple files.

**NOTE**

SQL Server has dynamic disk space management. This means that databases can be set up to automatically expand and shrink as needed. If you want to change the Autogrowth property in the Files page, click the ellipses (…) in the Autogrowth column and make your changes in the Change Autogrowth dialog box. The Enable Autogrowth check box should be checked to allow the database to autogrow. Each time there is insufficient space within the file when data is added to the database, the server will request the additional space from the operating system. The amount (in megabytes) of the additional space is set by the number in the File Growth frame of the same dialog box. You can also decide whether the file can grow without any restrictions (the default value) or not. If you restrict the file growth, you have to specify the maximum file size.

The Filegroups page of the Database Properties dialog box displays the name(s) of the filegroup(s) to which the database file belongs, the art of the filegroup (default or nondefault), and the allowed operation on the filegroup (read/write or read-only).

The Options page of the Database Properties dialog box enables you to display and modify all database-level options. There are several groups of options: Automatic, Containment, Cursor, Miscellaneous, Recovery, Service Broker, and State. For instance, the following four options exist for State:
Database Read-Only  Allows read-only access to the database. This prohibits users from modifying any data. (The default value is False.)

Database State  Describes the state of the database. (The default value is Normal.)

Restrict Access  Restricts the use of the database to one user at a time. (The default value is MULTI_USER.)

Encryption Enabled  Controls the database encryption state. (The default value is False.)
The Extended Properties page displays additional properties of the current database. Existing properties can be deleted and new properties can be added from this dialog box.

If you choose the Permissions page, the system opens the corresponding dialog box and displays all users and roles along with their permissions. (For the discussion of permissions, see Chapter 12.)

The rest of the pages (Change Tracking, Mirroring, and Transaction Log Shipping) describe the features which are related to data availability and are therefore explained in detail in Chapter 16.

Modifying Databases Without Using Transact-SQL

Object Explorer can also be used to modify an existing database. Using this component, you can modify files and filegroups that belong to the database. To add new data files, right-click the database name and choose Properties. In the Database Properties dialog box, select Files, click Add, and type the name of the new file. You can also add a (secondary) filegroup for the database by selecting Filegroups and clicking Add.

**NOTE**

Only the system administrator or the database owner can modify the database properties just mentioned.

To delete a database using Object Explorer, right-click the database name and choose Delete.

Managing Tables Without Using Transact-SQL

After you create a database, your next task is to create all tables belonging to it. As with database creation, you can create tables by using either Object Explorer or Transact-SQL. Again, only Object Explorer is discussed here. (The creation of a table and all other database objects using the Transact-SQL language is discussed in detail in Chapter 5.)

To create a table using Object Explorer, expand the Databases folder, expand the database, right-click the Tables subfolder, and then click New Table.

To demonstrate the creation of a table using Object Explorer, the `department` table of the `sample` database will be used as an example. Enter the names of all columns with their properties. Enter the column names, their data types, and the NULL property of each column in the two-dimensional matrix, as shown in the top-right pane of Figure 3-7.
Chapter 3: SQL Server Management Studio

All data types supported by the system can be displayed (and one of them selected) by clicking the arrow sign in the Data Type column (the arrow appears after the cell has been selected). Subsequently, you can type entries in the **Length**, **Precision**, and **Scale** rows for the chosen data type on the Column Properties tab (see the bottom-right pane of Figure 3-7). Some data types, such as CHAR, require a value for the **Length** row, and some, such as DECIMAL, require a value in the **Precision** and **Scale** rows. On the other hand, data types such as INTEGER do not need any of these entries to be specified. (The valid entries for a specified data type are highlighted in the list of all possible column properties.)

The check box in the Allow Nulls column must be checked if you want a table column to permit NULL values to be inserted into that column. Similarly, if there is a default value, it should be entered in the **Default Value or Binding** row of the Column Properties tab. (A default value is a value that will be inserted in a table column when there is no explicit value entered for it.)
The column **dept_no** is the primary key of the **department** table. (For the discussion of primary keys of the **sample** database, see Chapter 1.) To specify a column as the primary key of a table, you must right-click the column and choose Set Primary Key. Finally, click the × in the right pane with the information concerning the new table. After that, the system will display the Choose Name dialog box, where you can type the table name.

To view the properties of an existing table, double-click the folder of the database to which the table belongs, double-click Tables, and then right-click the name of the table and choose Properties. Figure 3-8 shows the Table Properties dialog box for the **employee** table.

**Figure 3-8**  Table Properties dialog box for the employee table
Chapter 3: SQL Server Management Studio

To rename a table, right-click the name of the table in the Tables folder and choose Rename. To remove a table, right-click the name of the table in the Tables folder in the database to which the table belongs and select Delete.

**NOTE**

You should now create the other three tables of the sample database.

After you have created all four tables of the sample database (employee, department, project, and works_on), you can use another feature of SQL Server Management Studio to display the corresponding entity-relationship (ER) diagram of the sample database. (The process of converting the existing tables of a database into the corresponding ER diagram is called reverse engineering.)

To see the ER diagram of the sample database, right-click the Database Diagrams subfolder of the sample database and select New Database Diagram.

**NOTE**

If a dialog box opens asking you whether the support objects should be created, click Yes.

The first (and only) step is to select tables that will be added to the diagram. After adding all four tables of the sample database, the wizard completes the work and creates the diagram (see Figure 3-9).

The diagram shown in Figure 3-9 is not the final diagram of the sample database because, although it shows all four tables with their columns (and the corresponding primary keys), it does not show any relationship between the tables. A relationship between two tables is based on the primary key of one table and the (possible) corresponding column(s) of the other table. (For a detailed discussion of these relationships and referential integrity, see Chapter 5.)

There are exactly three relationships between the existing tables of the sample database: first, the tables department and employee have a 1:N relationship, because for each value in the primary key column of the department table (dept_no), there is one or more corresponding values in the column dept_no of the employee table. Analogously, there is a relationship between the tables employee and works_on, because only those values that exist in the primary key of the employee table (emp_no) appear also in the column emp_no of the works_on table. The third relationship is between the tables project and works_on, because only values that
exist in the primary key of the `project` table (\texttt{pr_no}) appear also in the \texttt{pr_no} column of the `works_on` table.

To create each of the three relationships described, you have to redesign the diagram with the column that corresponds to the primary key column of the other table. (Such a column is called a \textit{foreign key}.) To see how to do this, use the `employee` table and define its column \texttt{dept_no} as the foreign key of the `department` table:

1. Click the created diagram, right-click the graphical form of the `employee` table in the detail pane, and select Relationships. In the Foreign Key Relationships dialog box, click Add.
2. In the dialog box, expand the Tables and Columns Specification column and click the … button.
3. Select the table with the corresponding primary key (the department table).
4. Choose the dept_no column of this table as the primary key and the column with the same name in the employee table as the foreign key and click OK. Click Close.

Figure 3-10 shows the modified diagram after all three relationships in the sample database have been created.
Authoring Activities Using SQL Server Management Studio

SQL Server Management Studio gives you a complete authoring environment for all types of queries. You can create, save, load, and edit queries. SQL Server Management Studio allows you to work on queries without being connected to a particular server. This tool also gives you the option of developing your queries with different projects.

The authoring capabilities are associated with Query Editor as well as Solution Explorer, both of which are described in this section. Besides these two components of SQL Server Management Studio we will describe how you can debug SQL code using the existing debugger.

Query Editor

To launch the Query Editor pane, click the New Query button in the toolbar of SQL Server Management Studio. If you expand it to show all the possible queries, it shows more than just a Database Engine query. By default, you get a new Database Engine query, but other queries are possible, such as MDX queries, XMLA queries, and other queries.

Once you open Query Editor, the status bar at the bottom of the pane tells you whether your query is in a connected or disconnected state. If you are not connected automatically to the server, the Connect to SQL Server dialog box appears, where you can type the name of the database server to which you want to connect and select the authentication mode.

NOTE

Disconnected editing has more flexibility than connected editing. You can edit queries without having to choose a server, and you can disconnect a given Query Editor window from one server and connect it to another without having to open another window. (You can use disconnected editing by clicking the Cancel button in the Connect to SQL Server dialog box.)

Query Editor can be used by end users for the following tasks:

- Generating and executing Transact-SQL statements
- Storing the generated Transact-SQL statements in a file
- Generating and analyzing execution plans for generated queries
- Graphically illustrating the execution plan for a selected query
Query Editor contains an internal text editor and a selection of buttons in its toolbar. The main window is divided into a query pane (upper) and a results pane (lower). Users enter the Transact-SQL statements (queries) that they want to execute into the query pane, and after the system has processed the queries, the output is displayed in the results pane.

The example shown in Figure 3-11 demonstrates a query entered into Query Editor and the output returned. The first statement in the query pane, USE, specifies the **sample** database as the current database. The second statement, SELECT, retrieves all the rows of the **works_on** table. Clicking the Query button in the Query Editor toolbar and then selecting Execute or pressing F5 returns the results of these statements in the results pane of Query Editor.

Figure 3-11  Query Editor with a query and its results
You can open several different windows—that is, several different connections to one or more Database Engine instances. You create new connections by clicking the New Query button in the toolbar.

The following additional information concerning the execution of the statement(s) is displayed in the status bar at the bottom of the Query Editor window:

- The status of the current operation (for example, “Query executed successfully”)
- Database server name
- Current username and server process ID
- Current database name
- Elapsed time for the execution of the last query
- The number of retrieved rows

One of the main features of SQL Server Management Studio is that it’s easy to use, and that also applies to the Query Editor component. Query Editor supports a lot of features that make coding of Transact-SQL statements easier. First, Query Editor uses syntax highlighting to improve the readability of Transact-SQL statements. It displays all reserved words in blue, all variables in black, strings in red, and comments in green. (For a discussion of reserved words, see the next chapter.)

There is also the context-sensitive help function called Dynamic Help that enables you to get help on a particular statement. If you do not know the syntax of a statement, just highlight that statement in the editor and select Help | Dynamic Help. You can also highlight options of different Transact-SQL statements to get the corresponding text from Books Online.

SQL Server 2012 supports the SQL Intellisense tool. Intellisense is a form of automated autocompletion. In other words, this add-in allows you to access descriptions of frequently used elements of Transact-SQL statements without using the keyboard.

Object Explorer can also help you edit queries. For instance, if you want to see the corresponding CREATE TABLE statement for the employee table, drill down to this database object, right-click the table name, select Script Table As, and choose CREATE to New Query Editor Window. Figure 3-12 shows the Query Editor...
window with the CREATE TABLE statement. (This capability extends also to other objects, such as stored procedures and functions.)

Object Explorer is very useful if you want to display the graphical execution plan for a particular query. (The execution plan is the plan selected by the optimizer to execute a given query.) If you select Query | Display Estimated Execution Plan, the system will display the graphical plan instead of the result set for the given query. This topic is discussed in detail in Chapter 19.

**Solution Explorer**

Query editing in SQL Server Management Studio is solution-based. If you start a blank query using the New Query button, it will still be based on a blank solution. You can see this by choosing View | Solution Explorer right after you open your blank query.
A solution can have zero, one, or more projects associated with it. A blank solution does not contain any project. If you want to associate a project with the solution, close your blank solution, Solution Explorer, and the Query Editor window, and start a new project by choosing File | New | Project. In the New Project window, choose SQL Server Scripts. A project is a method of organizing files in a selected location. You can choose a name for the project and select its location on disk. When you create a new project, by default you start a new solution. You can add a project to an existing solution using Solution Explorer.

Once the new project and solution are created, Solution Explorer shows nodes in each project for Connections, Queries, and Miscellaneous. To open a new Query Editor window, right-click the Queries node and choose New Query.

**SQL Server Debugging**

Since SQL Server 2008, you can debug SQL code using the existing debugger. To start debugging, choose Debug | Start Debugging in the main menu of SQL Server Management Studio. A batch from Chapter 8 (see Example 8.1) will be used here to demonstrate how the debugger works. (A batch is a sequence of SQL statements and procedural extensions that comprises a logical unit and is sent to the Database Engine for execution of all statements included in the batch.) Figure 3-13 shows a batch that counts the number of employees working for the p1 project. If the number is 4 or more, the corresponding message is displayed. Otherwise, first and last names of the employees will be printed.

You can set the breakpoints shown in Figure 3-13 just by clicking in front of the line where the execution process should stop. At the beginning, the editor shows a yellow arrow to the left of the first line of code. You can move the arrow by choosing Debug | Continue. In that case, all statements up to the first breakpoint are executed, and the yellow arrow moves to that breakpoint.

In debugger mode, SQL Server Management Studio opens two panes, which are placed at the bottom of the editor. All the information concerning the debugging process is displayed in these two panes. Both panes have tabs that you can select to control which set of information is displayed in the pane. The left pane contains Autos, Locals and up to four Watch tabs. The right pane contains Call Stack, Threads, Breakpoints, Command Window, Immediate Window, and Output tabs. For instance, you can use the Locals tab to view values of variables, the Call Stack tab to review the call stack, and the Watch tabs to type (or drag) a part of the code of an SQL expression and to evaluate it. (In Figure 3-13, for instance, the Watch1 tab is activated in the left pane, and the Breakpoints tab is activated in the right pane.)

To end the debugging process, select the blue square icon in the debugging toolbar or choose Debug | Stop Debugging.
SQL Server 2012 enhances the functionality of the SQL Server Management Studio Debugger with several new features. You can now do the following:

- **Specify a breakpoint condition** A breakpoint condition is an SQL expression whose evaluation determines whether the breakpoint is invoked. To specify a breakpoint condition, right-click the breakpoint glyph and click Condition on the pop-up menu. In the Breakpoint Condition dialog box, enter a Boolean expression and choose either Is True, if you want to break when the expression evaluates to true, or Has Changed, if you want to break, when the value has changed.

- **Specify a breakpoint hit count** A hit count is a counter that specifies the number of times a breakpoint is reached. If the specified hit count is reached, and any specified breakpoint condition is satisfied, the debugger performs the action specified for the breakpoint. The action could be any of the following:
  - Break always (the default action)
  - Break when the hit count equals a specified value
Break when the hit count equals a multiple of a specified value

Break when the hit count is greater than or equal to a specified value

To specify a hit count, right-click the breakpoint glyph on the Breakpoint window and click Hit Count on the pop-up menu (see Figure 3-13). In the Breakpoint Hit Count dialog box, select one of the actions from the preceding list. If you need to set the hit count to a value, enter an integer in the text box that appears. Click OK to make the modifications.

Specify a breakpoint filter  A breakpoint filter limits the breakpoint to operating only on specified computers, processes, or threads. To specify a breakpoint filter, choose Breakpoint | Filter. You can then specify the resource that you want to limit in the Breakpoint Filters dialog box. Click OK to make the modifications.

Specify a breakpoint action  A breakpoint When Hit action specifies a custom task that is performed when the breakpoint is invoked. The default action for a breakpoint is to break execution when both the hit count and breakpoint condition have been satisfied. The alternative could be to print a specified message.

To specify a breakpoint action, right-click the breakpoint glyph and then click When Hit on the pop-up menu. In the When Breakpoint Is Hit dialog box, select the action you want. Click OK to make the modifications.

Use the QuickWatch window  You can use the QuickWatch window to view the value of a Transact-SQL expression, and then save that expression to a Watch window. (To select the Quick Watch window, choose Debug | Quick Watch.) To select an expression in QuickWatch, either select or enter the name of the expression in the Expression field of the Quick Watch window.

Use the Quick Info pop-up  When you move the cursor over an SQL identifier, the Quick Info pop-up displays the name of the expression and its current value.

Summary

This chapter covered the most important SQL Server tool: SQL Server Management Studio. SQL Server Management Studio is very useful for end users and administrators alike. It allows many administrative functions to be performed. These are touched on here but are covered in more detail later in the book. This chapter discussed most of the important functions of SQL Server Management Studio concerning end users, such as database and table creation.
Chapter 3: SQL Server Management Studio

SQL Server Management Studio contains, among others, the following components:

- **Registered Servers**  Allows you to register SQL Server instances and connect to them.
- **Object Explorer**  Contains a tree view of all the database objects in a server.
- **Query Editor**  Allows end users to generate, execute, and store Transact-SQL statements. Additionally, it provides the ability to analyze queries by displaying the execution plan.
- **Solution Explorer**  Allows you to create solutions. A solution can have zero or more projects associated with it.
- **Debugger**  Allows you to debug code.

The next chapter introduces the Transact-SQL language and describes its main components. After introducing the basic concepts and existing data types, the chapter also describes system functions that Transact-SQL supports.

**Exercises**

**E.3.1**
Using SQL Server Management Studio, create a database called *test*. Store the database in a file named *testdate_a* in the directory C:\tmp and allocate 10MB of space to it. Configure the file in which the database is located to grow in increments of 2MB, not to exceed a total of 20MB.

**E.3.2**
Using SQL Server Management Studio, change the transaction log for the *test* database. Give the file an initial size of 3MB, with growth of 20 percent. Allow the file for the transaction log to autogrow.

**E.3.3**
Using SQL Server Management Studio, allow only the database owner and system administrator to use the *test* database. Is it possible that both users could use the database at the same time?
E.3.4
Using SQL Server Management Studio, create all four tables of the sample database (see Chapter 1) with all their columns.

E.3.5
Using SQL Server Management Studio, view which tables the AdventureWorks database contains. After that, choose the Person.Address table and view its properties.

E.3.6
Using Query Editor, type the following Transact-SQL statement:

```
CREATE DATABASE test
```

Explain the error message shown in the result pane.

E.3.7
Store the Transact-SQL statement in E.3.6 in the file C:\tmp\createdb.sql.

E.3.8
Using Query Editor, how can you make the test database the current database?

E.3.9
Using Query Editor, make the AdventureWorks database the current database and execute the following Transact-SQL statement:

```
SELECT * FROM Sales.Customer
```

How can you stop the execution of the statement?

E.3.10
Using Query Editor, change the output of the SELECT statement (E.3.9) so that the results appear as the text (and not as the grid).