3

SQL Server 2008 Tools

Several years ago, when I was beta testing SQL Server 2005, I was surprised to see familiar tools like the Enterprise Manager, a Microsoft Management Console (MMC)-based interface, and the SQL Query Analyzer done away with. In fact, with the exception of the SQL Server Profiler, pretty much everything had been replaced with a new set of applications that were . . . well, different.

It's been my experience that most database administrators (DBAs) typically fall into one of two distinct groups. The first group is made up of database administrators whose background is system and network administration. The second group is made up of application and database developers who have become responsible for the administration of a SQL Server infrastructure, be it a production system or a test-bed environment. DBAs that fell into the first category, myself included, often responded with trepidation about the new SQL Server management tools, and with good reason. Most of the new tools available were based on the Visual Studio interface. In fact, one of them was indeed Visual Studio (although rebranded to sound less intimidating). What was Microsoft trying to do — make us developers?

Yes. A database administrator must be about half system administrator and half developer in order to be completely successful. Several years ago, when Microsoft announced its Microsoft Certified Database Administrator (MCDBA) certification, it was no real surprise that the required exams were both from the administrative side of database administration and the programming side. Microsoft's intent was clear. To be a database administrator worth his or her salt, it would be absolutely imperative to understand database design and database application development. This was where Microsoft wanted DBAs to go, and they made sure we had the tools to get there. Ironically, the current generation of certifications, the Microsoft Certified Information Technology Professional (MCITP), includes two distinct specializations for Database Administrators and for Database Developers.

There is no doubt that Microsoft considers database administrators to be, at least marginally, developers. However, this does not mean that the tools are not intuitive and easy to use. In fact, after having spent more than three years working with them, I can't ever see myself going back to what some of us jokingly referred to as *Enterprise Mangler*. The tools that you will use today to manage SQL Server 2008 (and supported previous versions) are *more* intuitive and *easier* to use. DBAs will also be able to take advantage of functionality that developers have become used to, such as source control, solution files that manage multiple related files, and a fully functional Integrated Development Environment (IDE).

If you've never worked with SQL before or haven't managed SQL since SQL Server 2000, the new tools may seem daunting. In reality, they are more streamlined, more efficient, and yet more powerful than anything we've had before.

SQL Server Management Studio

SQL Server Management Studio completely replaces Enterprise Manager and Query Analyzer from SQL Server 2000 and earlier. It also replaces some of the functionality formerly found in other applications, such as SQL Analysis Manager. The bulk of work that I often do is performed through SQL Server Management Studio, or SSMS.

On first glance, the SQL Server Management Studio interface looks a lot like the Visual Studio IDE. It should, since it is, in actuality, a Visual Studio shell. The Visual Studio shell brings many very useful tools and features to the creation and organization of database objects, as well as the full feature set of the old tools.

When the SQL Server Management Studio is first launched, the default view is a great deal like the old Enterprise Manager with a slight Query Analyzer influence (see Figure 3-1).

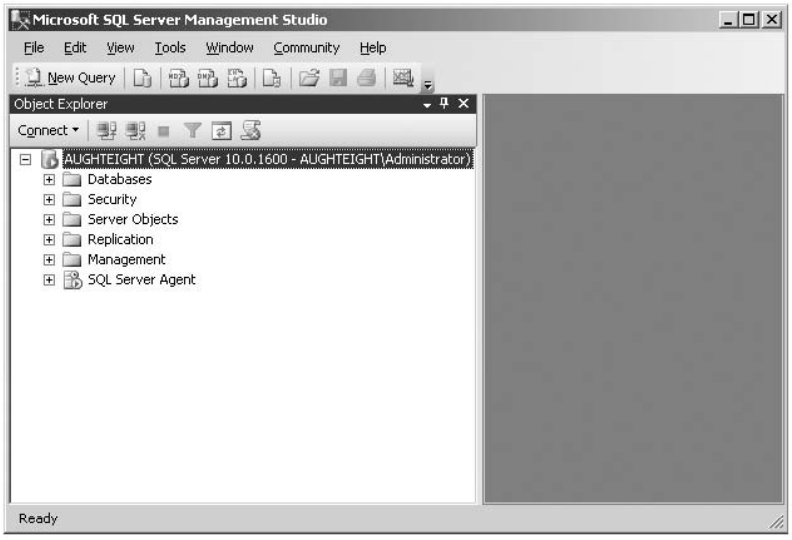


Figure 3-1: SQL Server Management Studio.

Because there are many different windows that can be viewed in the Management Studio, the management of screen real estate becomes critical. Most of the windows have the capability to either be pinned open or configured to fly out when the mouse pointer is placed over the menu bar, or auto-hide when the mouse cursor is placed elsewhere. If you are familiar with the Visual Studio Integrated Development Environment (IDE), this will all be very familiar; if not, it may take a little while to get used to.

If you are unfamiliar with the Visual Studio interface, here's a tip: Any window that supports the pinned or unpinned option will have a pin at the top right of the window. When the window is pinned, the pin

will appear vertically oriented. When the window is unpinned, it will be horizontal (see Figure 3-2), and the toolbar will auto-hide or fly out, depending on the mouse cursor location.

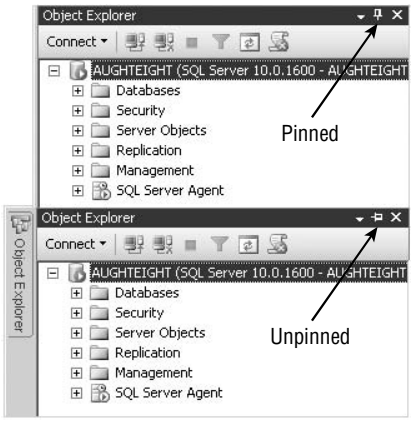


Figure 3-2: Object Explorer with a pinned and unpinned window.

As mentioned before, the Visual Studio interface has a bit of a learning curve, but once you get used to it, it's hard to imagine any interface that works as well. The biggest advantage of the interface is that it's heavily customizable. Everything from window placement to colors can be altered to suit your personal management style. I used to drive my old manager (and cowriter), Dan, crazy by setting my Query window to a black background with bright green text (yes, it was hideous). Being able to hide and unhide windows with little effort offers a huge benefit. This conserves a great deal of screen real estate without having to click several menus to expose the features you want. The expanding popularity of Netbook computers with smaller screen sizes and limited resolution makes this a more and more attractive feature for those of us who tend to administer from the road.

Tool Windows

SQL Server Management Studio offers many different tool windows that facilitate the development and modification of database objects, as well as the effective management of SQL Server. The various views are accessible from the View menu as well as the Standard Toolbar. Each window can be configured as Dockable, which is the default, but can also be configured as a Tabbed Document or a Floating window. You can change the state of the window by clicking on the down arrow next to the pushpin in the window's title bar, or if the window is floating, by right-clicking on the title bar (Figure 3-3).

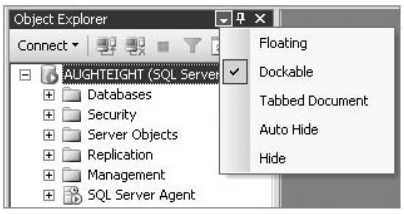


Figure 3-3: Window placement options.

A *dockable window* means that the window can be dragged and docked at almost any location in the environment. If you don't like the Object Explorer window on the left of the Studio, just drag it to the right, top, or bottom, and dock it there. When dragging a tool window, a guide diamond will appear in the center of the screen representing the dockable areas. Dragging the window over one of the area representations (see Figure 3-4) will cause a shadow to appear in that area, indicating that the window can be docked there by releasing the mouse button.

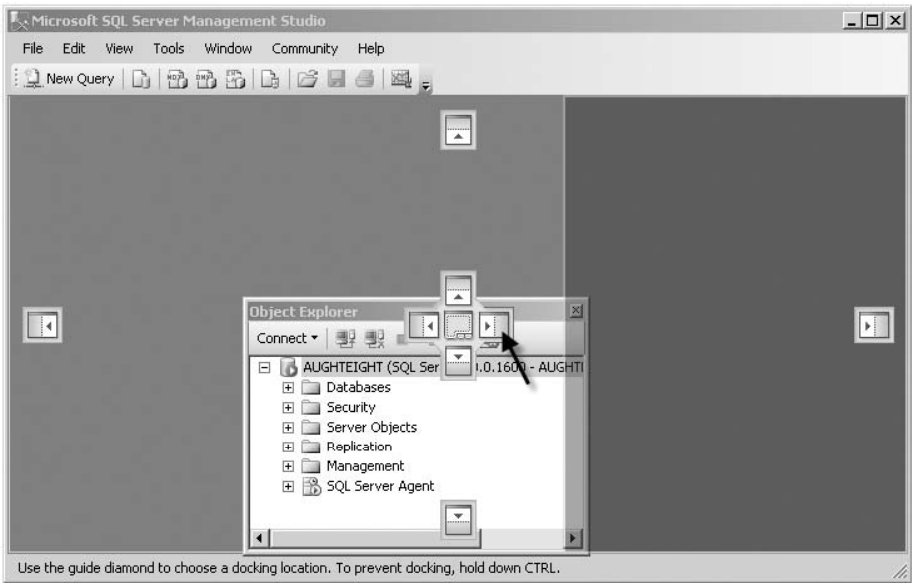


Figure 3-4: Dockable window.

Changing a windows property to Tabbed Document mode changes the window into a tab on the main window. The Floating window option specifies that the tool window is not anchored anywhere and can be moved around the main interface.

Object Explorer

The Object Explorer (see Figure 3-2) is more than just a way to explore the database objects on a server. The Object Explorer is also the tool that will be used to initiate most database management tasks. It is arranged in a standard tree view with different groups of objects nested in folders.

The Object Explorer's functionality is exposed through the context menu. Right-clicking on any object or folder within the Object Explorer exposes a list of context-sensitive options, from creating tables and users to configuring replication and Database Snapshots. The context menu also presents the ability to create scripts that manipulate. For example, right-clicking on a table exposes a context menu that allows the user to either view or modify the table structure through the graphical interface, or create scripts to perform actions against the table or its data (perhaps to be saved and executed later). This functionality exists for virtually every object that is visible in the Object Explorer.

Another great feature of SQL Server Management Studio that is exposed through the Object Explorer and other areas of the Studio interface is the ability to create scripts based on actions performed in the graphical designers. For example, right-clicking on the table folder and choosing to create a new folder launches a graphical interface where the table structure can be defined. Once the table design is complete,

you can either save the table (which creates it) or click the ''Generate Change Script'' button on the Table Designer toolbar (which will write the appropriate T-SQL to complete the task). Using the ''Generate Change Script'' option can be beneficial when creating objects in a test or development environment that will also need to be created in a production environment.

Likewise, when working with other objects in Management Studio, a Script button will appear at the top of the respective designer, which will cause the actions performed in the designer to be scripted to a new Editor window. This feature is particularly useful when several different objects of the same type are to be created. The first one can be designed in the designer, the script generated for it, and that script modified to create the remaining objects. It can also be a good learning tool, by allowing inexperience database administrators to learn the T-SQL equivalent of a task that is performed through the Graphical User Interface (GUI).

Creating a Script

In the following example, you use the Object Explorer to create a script for a new database called `DVDCollection`:

1. In Object Explorer, right-click Databases. In the context menu that appears, click ''New Database.''
2. The New Database dialog appears (see Figure 3-5).

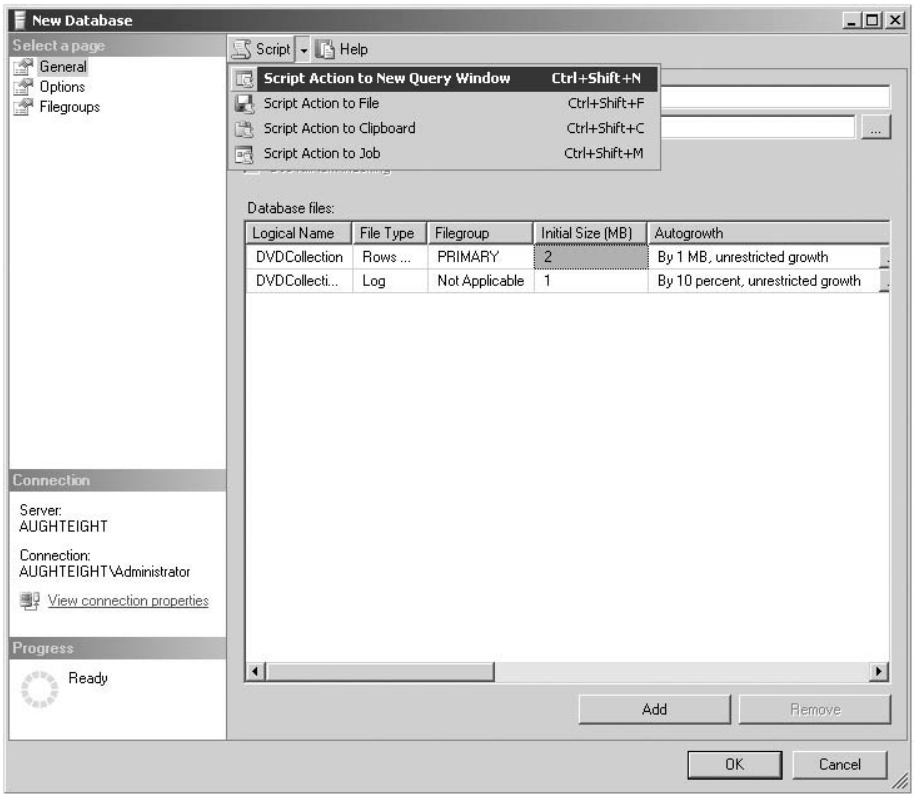


Figure 3-5: New Database dialog.

3. Enter **DVDCollection** for the name of the database.
4. Click on the Script button at the top of the New Database dialog.
5. The Script button causes the appropriate T-SQL code to be written to a new Query window.

   Clicking the down arrow to the right of the Script button (Figure 3-5) gives you the option of sending the script to a variety of locations.
6. In the New Database dialog box, click Cancel. (Clicking OK will cause the database to be created.)

The script remains, but the database is not created unless the script is executed.

Code Editor

The Code Editor in SQL Server Management Studio provides the ability to open, edit, or create new queries. The types of queries supported by the Editor are:

- **Database Engine Queries** — These are written in Transact-SQL (T-SQL) against a SQL Server OLTP database.
- **Analysis Services MDX Queries** — These use the *MultiDimensional eXpression* (MDX) language. MDX queries are used to retrieve information from multidimensional objects created in Analysis Services.
- **Analysis Services DMX Queries** — These are created by using extensions to the Structured Query Language (SQL) called *Data Mining eXtensions* (DMX). DMX queries are written to return information from data-mining models created in SQL Server Analysis Services databases.
- **Analysis Services XMLA Queries**
- **SQL Server Compact** — As the name implies, these can perform Transact-SQL queries using a SQL Server Compact Edition database file as a data source.

The Code Editor is essentially a word processor. It provides color coding of syntax, multiple query windows, and partial code execution when you highlight the desired code and click on the Execute button or press [F5]. The SQL Server 2008 Books Online documentation will often refer to the Code Editor as the *Query Editor* (its most common moniker), *Text Editor*, or simply the *Editor*, depending on what aspect of SQL Server you are reading about.

The basic functionality that the Code Editor brings is the same for all the possible types of queries it supports. However, more complete functionality is provided for specific languages. For example, when creating MDX, DMX, or XMLA queries, the Code Editor provides basic IntelliSense functions such as those found in Visual Studio. SQL Server 2008 also introduces, for the first time, IntelliSense for Transact-SQL, which includes code completion (for object names) and error handling. For example, while typing the following script, as soon as you type the **H** in *HumanResources*, a dropdown list appears with the HumanResources schema selected. Pressing the period (.) key results in a list of objects that exist within the HumanResources schema, from which you can use the arrow keys to highlight and select the Employee table.

```
USE AdventureWorks2008

Select * from HumanResources.Employee
Where Gender = 'M';
GO
```

Additionally, if you mouse-over the column name, *Gender*, the Query Editor provides you with metadata about the gender column, as shown in Figure 3-6.

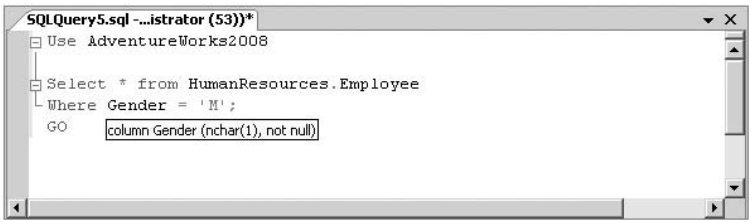


Figure 3-6: IntelliSense displaying column information.

Right-clicking on the Code Editor window, when that window is associated with a Database Engine query, results in a context menu that includes the ''Design Query in Editor'' option (see Figure 3-7). The Query Designer is also available from the SQL Editor toolbar described later. The Query Designer can be very helpful when writing queries against databases that are not familiar to the query writer.

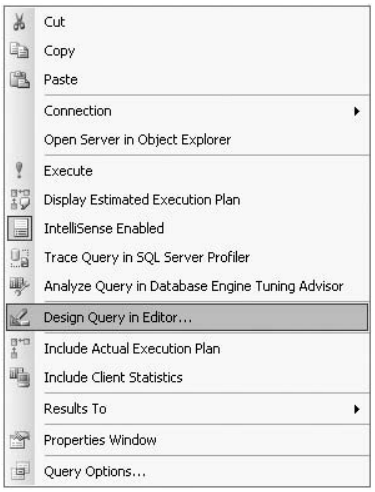


Figure 3-7: Query window context menu.

Solution Explorer

In the past, DBAs and database developers who had to keep track of saved queries that were used together as part of a batch process, or required source control and versioning, often had to manage multiple independent files manually. I don't know how many times I've browsed a common file system and found scattered .sql files stored here and there. SQL Server Management Studio takes full advantage of Visual Studio's solution system by providing the means of grouping various connection objects and scripts into a single solution called a *SQL Server Management Studio Solution*. Each solution can have one or more projects associated with it. For example, if you are developing several objects for a new application that includes both Database Engine and Analysis Engine objects, you can create a new solution that links them all together by creating a new SQL Server Management *Solution* with one or more associated *Projects* (see Figure 3-8).

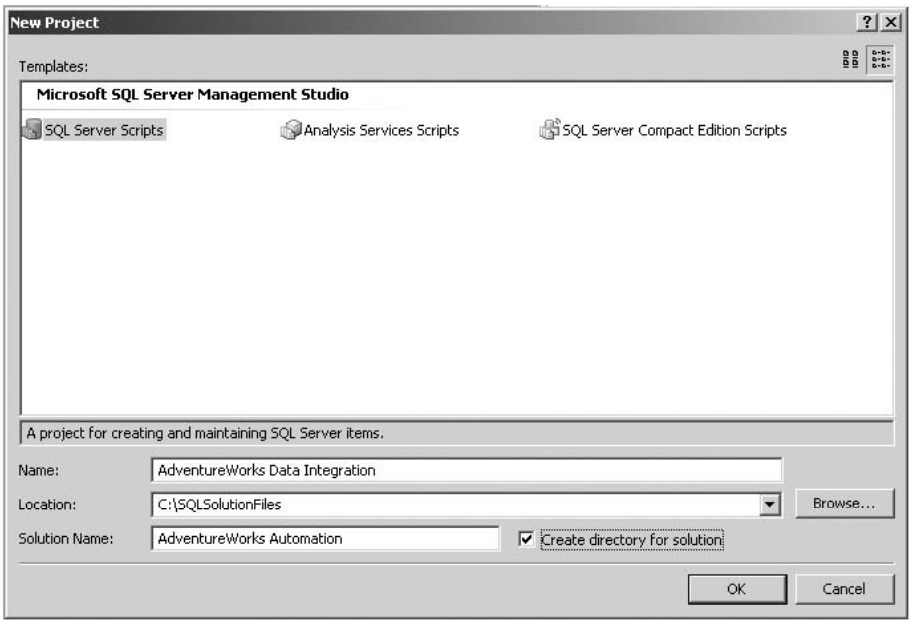


Figure 3-8: Associating projects and solutions.

If no solution is currently open, the Management Studio will create a new one. As you can see in Figure 3-8, there are three types of projects to choose from:

- **SQL Server Script** — These projects contain T-SQL Database Engine queries.
- **Analysis Services Script** — These projects contain MDX, DMX, and XMLA analysis queries.
- **SQL Server Compact Edition Script** — These projects contain SQL Server Compact queries, as you might expect.

The solution is managed through a SQL Server Management Studio Solution file with an .ssmssln extension. The example shown in Figure 3-8 created a new solution folder called *AdventureWorks Automation* that contains a project folder called *AdventureWorks Data Integration*. By default, the solution folder and the first project folder will have the same name, so it is generally a good idea to change the name of the

solution. The ''Create directory for solution'' option can also be cleared and a solution folder specified. In this way, only a project folder will be created in the specified directory. If a solution is already opened, creating a new project can add the project to the solution, or be configured to create a whole new solution and close the open one. Solutions can contain many projects. For example, a project called *AdventureWorks Data Preparation* can be added to organize the files for the sales piece of the solution (see Figure 3-9).

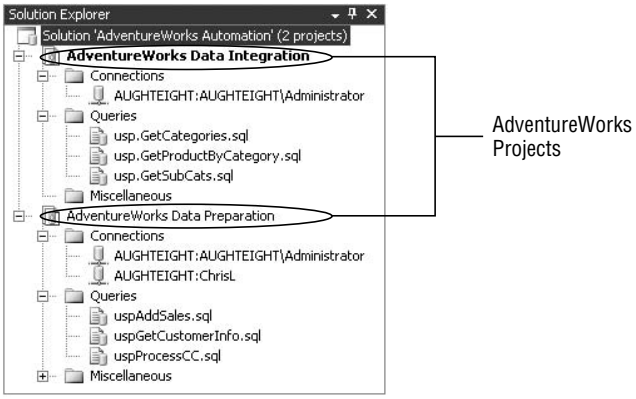


Figure 3-9: Multiple projects.

Projects contain three folders:

- **Connection Folders** — These folders store objects that contain connection parameters for the queries in the solution. For example, if you look at the AdventureWorks Data Preparation project shown in Figure 3-9, you will note that there are two connection objects, one for the AughtEight\Administrator account and another for a SQL account named *ChrisL*.
- **Queries Folders** — Each of the queries in the *Queries folders* of the project will use one of those configured connection objects. The query will run in the context of the associated connection object.
- **Miscellaneous Folder** — This folder can be used to store just about any other file that is pertinent to the project. This may be project documentation, XML files, or even the .NET assemblies used to create managed-code procedures.

The solution folder contains two files:

- **Solution File** — One file is the *solution file*, which, in this case, is called *AdventureWorks Automation.ssmssln*. This contains a list of all the projects in the solution and their locations.
- **SQL Solution Options File** — The second file is the *SQL Solution Options* file, *AdventureWorks Automation.sqlsuo*. The solution options file contains information about the options that customize the development environment. This file is hidden by default.

The solution folder will contain a project folder for every project added to the solution. The project folder contains all the project files, including the project definition file. The project definition file, or SQL Server Management Studio SQL Project file, is an XML file with the .ssmssqlproj

extension. In the previous AdventureWorks Data Integration project example, this file is called *AdventureWorks Data Integration.ssmssqlproj*. The project definition file contains the connection information, as well as metadata about the remaining files in the project.

Properties Window

The Properties window is linked to the Solution Explorer and simply displays the properties for the currently selected item in the Solution Explorer window. Editable properties will be bolded.

Registered Servers

Multiple servers can be registered and managed with the Management Studio. Right-clicking on any blank area in the Registered Servers window or on any server group name (see Figure 3-10) will expose a context menu that allows for the addition of new server registrations. It also allows for the creation of server groups. The Registered Servers window is not visible by default. To open it, use the View menu, and select Registered Servers or press *[Ctrl]+[Alt]+G*.

Figure 3-10: Registered Servers window.

If you have multiple servers in your organization, server groups can be very useful. For example, server registrations can be segregated so that all the test and development servers are in one group and the production servers are in another, or servers could be grouped based on function or department. Instances of the Database Engine, Analysis Services, Reporting Services, Integration Services, and SQL Server Compact can be registered in the Registered Servers window (Figure 3-10). Once registered, the Registered Servers window provides the ability to manage the associated services or launch other SQL Server tools associated with the respective instance.

A new feature of SQL Server 2008 includes the ability to use policy-based management, enforceable on multiple servers simultaneously through the use of Central Management servers. Central Management servers can be registered in the Registered Servers window and can also have Server Groups created to group together services with similar configuration requirements. Policy-based administration can be used to apply policies to the Central Management server, the Server Group, or the individual registered SQL Server. More information about Policy-Based administration is presented in Chapter 8.

Bookmark Window

When working with very large scripts in the Code Editor, it is very useful to be able to mark a location in the script. Bookmarks enable this functionality. The Bookmark window is made visible from the View menu and is enabled when working with any SQL Server script type. Any number of bookmarks can

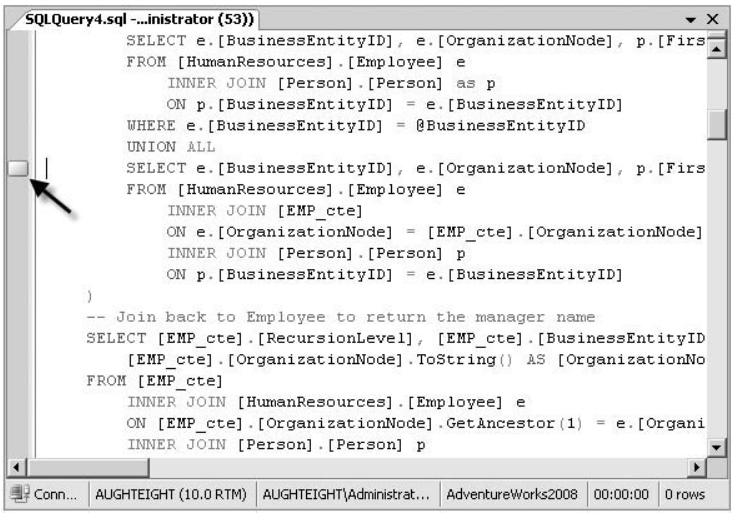


Figure 3-11: Bookmark window.

be created and then renamed with an intuitive name that identifies the bookmark (see Figure 3-11). If the script is part of a solution, the bookmarks are saved with the solution in the Solution Options file. Bookmarks can be added to a line by pressing *[Ctrl]+K* twice. Navigating bookmarks is easy. In addition to selecting the bookmarks in the Bookmark window, you can use the key combinations of *[Ctrl]+K*, *[Ctrl]+P* and *[Ctrl]+K*, *[Ctrl]+N* to move to the previous and next bookmarks, respectively. You can also organize your bookmarks into multiple folders for each project, which can make it easier to navigate through bookmarks by function.

Toolbox

The Toolbox window (see Figure 3-12) consists of maintenance plan tasks that can be dragged and dropped into maintenance plan subtasks using the Maintenance Plan Designer, which is described in more detail in Chapter 8.

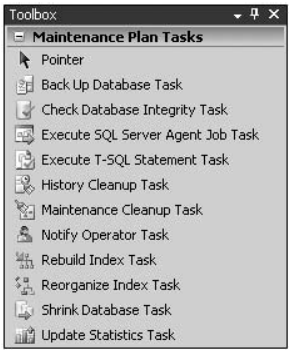


Figure 3-12: Toolbox window.

Error List

The Error List can be handy when trying to troubleshoot a query, even simple ones like the example in Figure 3-13, by providing descriptive information about the error, as well as line and position number in the query text. As you can see, the three lines of code have generated four errors. You can now resolve these errors before you execute your query.

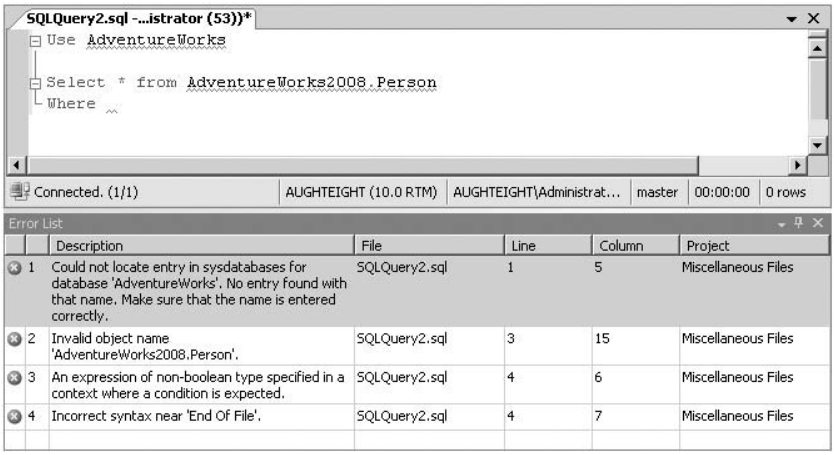


Figure 3-13: Error List window.

Object Explorer Details

The Object Explorer Details window replaces the Summary View from SQL Server 2005. It is a great deal like the List or Detail view in Windows Explorer; however, it also provides a very useful reporting feature. This feature allows the rendering of various server and database reports. The report feature is enabled when right-clicking on an object in the Object Explorer or in the Object Explorer Details window that has reports associated with it, and selecting the Reports option from the context menu. The following table contains a list of all the supported reports and where they can be found:

Continued

Web Browser

SQL Server Management Studio also includes the ability to launch a Web Browser window within the context of the management studio. The browser uses the Internet Explorer renderer, if desired, to minimize the number of open applications and to allow direct access to Internet content from within the Management Studio application. The Web Browser window is made visible from the View menu (or by pressing *[Ctrl]+[Alt]+R*). You can use the address bar at the top of the window to enter a URL, or you can use the Web Browser Search button to take you to the MSDN home page.

Although using a browser within Management Studio might seem unnecessary, it does offer some benefits. For example, it allows tabbed browsing of content or newsgroups that may be pertinent to the current solution. You can search or ask questions without having to switch back and forth between Management Studio and Internet Explorer. Keep in mind that the Web Browser window is just an instance of Internet Explorer embedded in Management Studio. The behavior of the Web Browser window is the same as Internet Explorer, and the security configuration of Internet Explorer is in full effect in the Web Browser window. However, because a separate executable is not launched, it may actually be more efficient from a resource perspective to launch the Web Browser within the context of Management Studio. For example, on my test system, when I opened up a new instance of IE and browsed to `www.msdn.com`, the process consumes about 13 MB of memory. Launching the Web Browser window in SSMS and clicking on the Web Search button in the toolbar increased the memory utilization for the SSMS process by only 3 MB.

Template Explorer

The Template Explorer (see Figure 3-14) contains hundreds of SQL Server, Analysis Server, and SQL Compact scripts. Each script is grouped into folders based on their function. The template scripts can be opened by being dragged onto an open Query window. If no Query window is open, the templates can be opened by double-clicking with a mouse, using the Edit menu, or right-clicking on a context menu, all of which cause a new Query window to open.

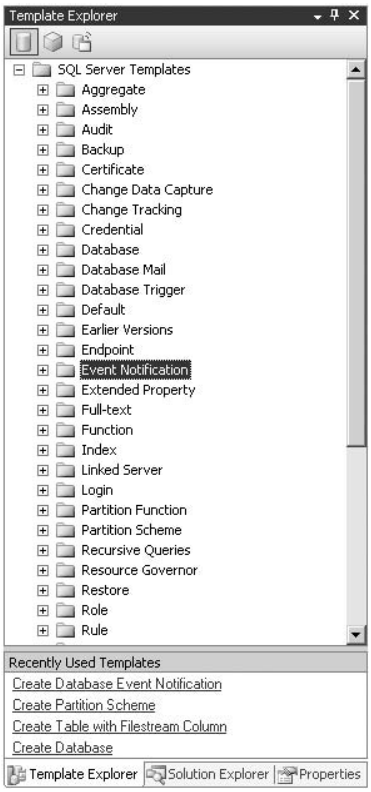


Figure 3-14: Template Explorer.

When using a template, you can modify the text directly in the Query Editor, or you can use the ''Specify Values for Template Parameters'' option to replace the placeholders in the template (see Figure 3-15). This dialog can be launched from the SQL Editor toolbar or through the Query menu.

Toolbars

SQL Server Management Studio includes 14 preconfigured toolbars that contain features from various menus. Each toolbar can be displayed or hidden by using the View ➢ Toolbars menu (see Figure 3-16). The existing toolbars can be customized to display only the buttons that are most often used, or you can create a new toolbar that has only the commands you typically use.

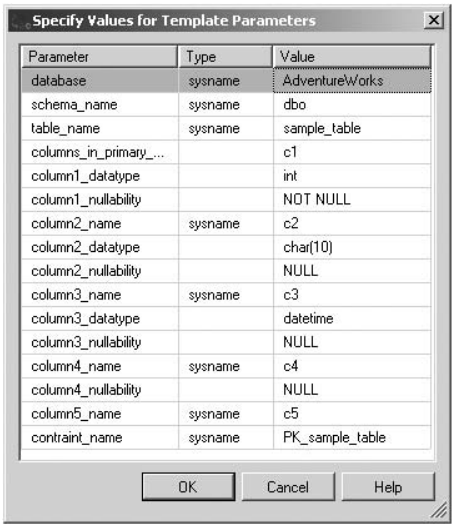


Figure 3-15: Parameter replacement.

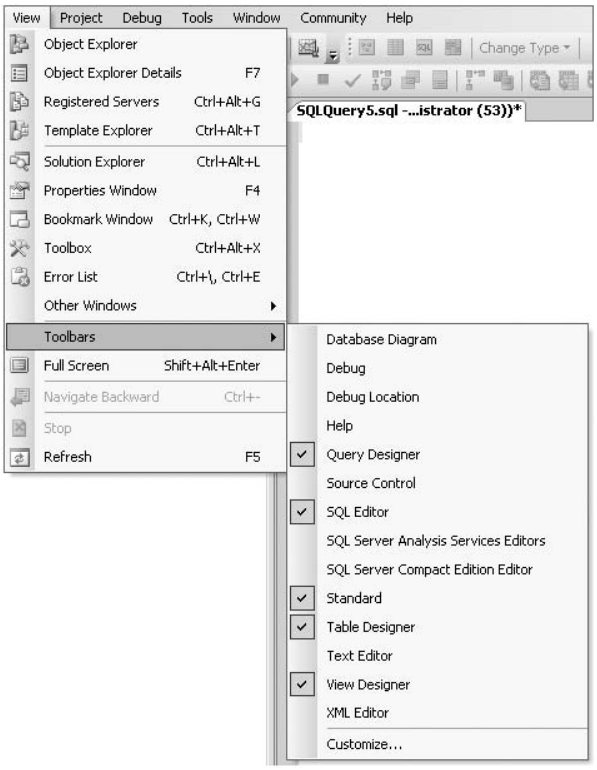


Figure 3-16: Toolbars menu.

Creating a Custom Toolbar

Create a new custom toolbar by completing the following steps:

1. Select the Customize command on the View ➢ Toolbars menu. This will launch the Customize window.
2. On the Customize window, click on the New button (see Figure 3-17), give your toolbar a new name (for this example, I just created one called *My Toolbar*), and click OK. Your new toolbar will show up in the Toolbars list, as well as a floating toolbar on your screen.

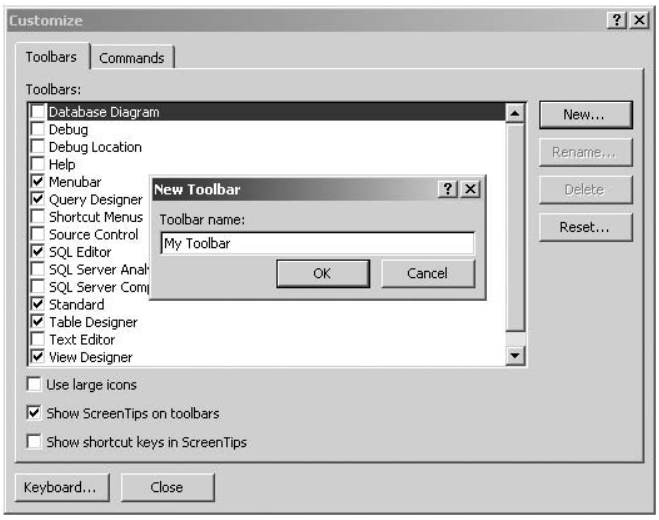


Figure 3-17: Custom toolbar window.

3. With your new toolbar highlighted, select the Commands tab on the Customize window. Two panes are visible on the Commands tab: Categories and Commands. Each category contains commands specific to that category. For example, the File category contains commands such as Open File, Save Project, and so on.
4. Select the Edit Category, and drag several commands to the new custom toolbar created in Step 2 (see Figure 3-18). You can also right-click on a button on the toolbar to change some of the options of that toolbar — for example, changing the name or button image used for that command. Once you have all the commands that you want on the new toolbar, you can drag it and dock it in a desired location.

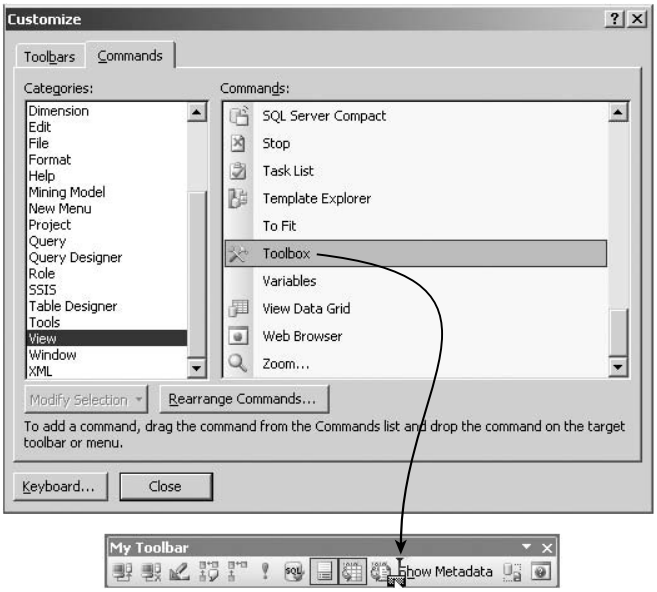


Figure 3-18: Custom edit toolbar.

Creating new toolbars or customizing existing ones can help you manage your screen real estate by allowing you to create more useful and efficient toolbars.

Database Diagram Toolbar

The Database Diagram toolbar (see Figure 3-19) exposes a great deal of functionality for use on database diagrams.

Figure 3-19: Database Diagram toolbar.

The toolbar is not used just for diagramming the database, but also for modifying or creating database objects from within the diagram interface. The Database Diagram toolbar features are described in the following table:

[redacted]

Debug Toolbar

The Debug toolbar, as shown in Figure 3-20, includes several tools useful when debugging projects in SQL Server Management Studio that let you step through long queries to help identify potential problem areas.

Figure 3-20: Debug toolbar.

The Debug toolbar's commands are described in the following table:

[redacted]

Debug Location Toolbar

The Debug Location toolbar (Figure 3-21) displays thread and stack frame information about the current command being executed in the Debug window.

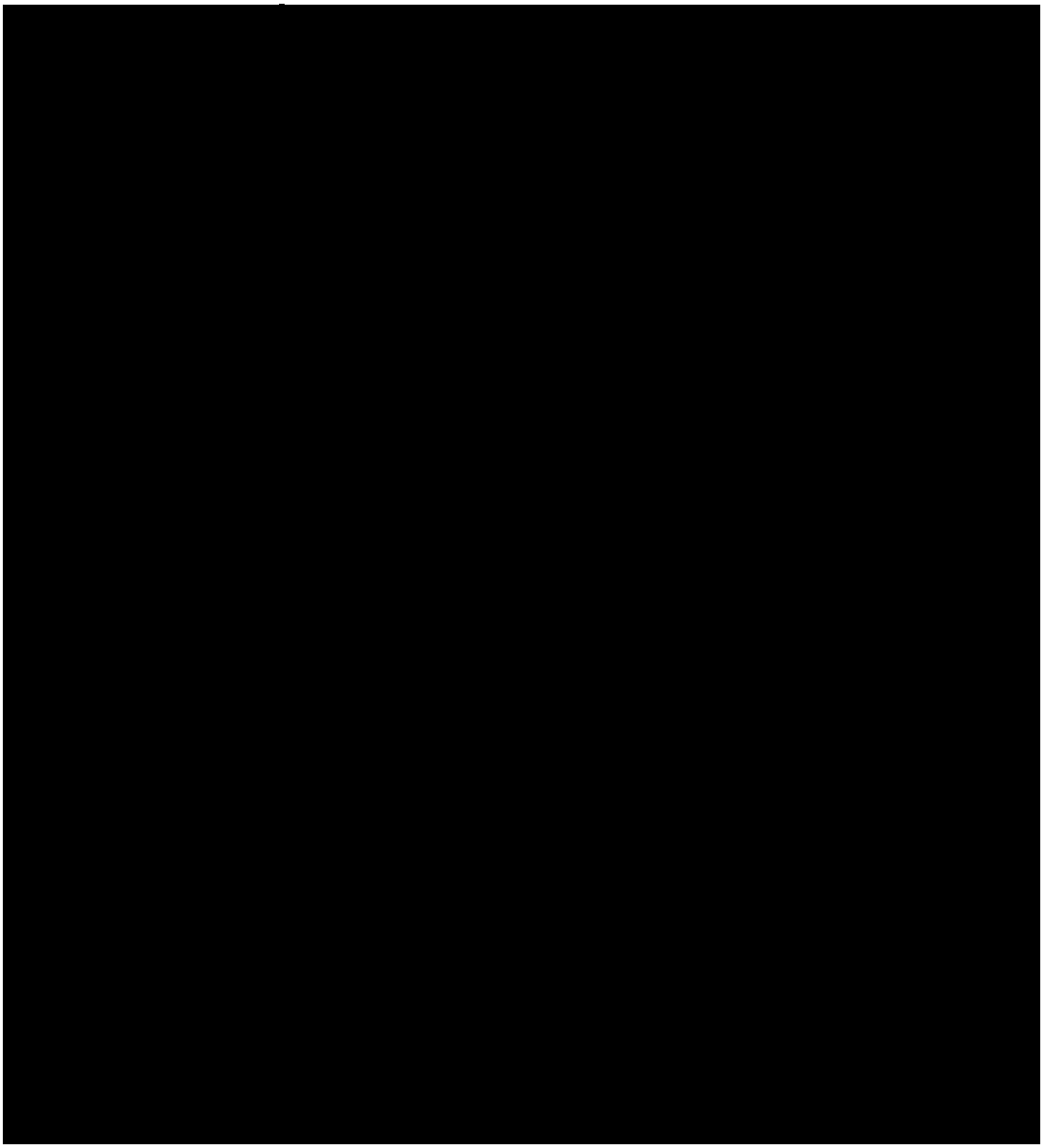
Figure 3-21: Debug Location toolbar.

Help Toolbar

The Help toolbar (see Figure 3-22) provides a very easy and convenient mechanism for consulting online help articles while using the Management Studio.

Figure 3-22: Help toolbar.

The Help toolbar's commands are described in the following table:



Query Designer Toolbar

The Query Designer toolbar (see Figure 3-23) is enabled when a table is opened for editing with Object Explorer.

Figure 3-23: Query Designer toolbar.

To open a table for editing, follow these steps:

1. Right-click on the table you want to open in Object Explorer.
2. Click ''Edit Top 200 Rows.''

If the Query Designer was not visible, it will be when the table is opened. If it was visible, it will now be enabled. Although opening a table in a test and development environment might be acceptable, opening a table in this manner in a production environment is never recommended. Opening a table with the Object Explorer dumps the data from the table in to an updatable scrollable cursor. What this means is that while the table data is exposed in the results window, any change to the displayed data is also made to the underlying data in the table. There is no confirmation message or warning. The data is just modified. This can be very dangerous. Displaying the entire contents of the table can also consume a great deal of server resources if the table is large. The default behavior of SQL Server Management Studio only exposes the first 200 rows of a table for editing, but this can be changed through the Tools ➢ Options menu. As a general rule, if the entire contents of a table need to be exposed for quick viewing, the best way is to write a query with no filters, such as the following:

```
USE AdventureWorks2008
GO
SELECT * FROM Person.Person
```

This exposes the same information as opening the table, but does not populate an updatable cursor, so the results are Read Only. If the data in that table needs to be updated, an update command is more appropriate than modifying the data in an open table results window.

The Query Designer toolbar features are described in the following table:

[redacted]

Source Control Toolbar

The Source Control toolbar (see Figure 3-24) is enabled when working with scripts and a Source Control plug-in has been configured such as Visual Source Safe 2005. The addition of source-control functionality to SQL Server projects is a great step forward in recognizing the need for a structured solution environment in the development of database solutions.

Figure 3-24: Source Control toolbar.

The following example uses Visual Source Safe 2005 as the source-control tool, but there are other source-control applications available that will interact with SQL Server Management Studio. A full description of Visual Source Safe 2005 configuration and use is beyond the scope of this book, so it will be limited to just the interaction with SQL Server Management Studio.

To configure Management Studio to use source control:

1. Click File, and click Launch Microsoft Visual Source Safe. The Add Visual SourceSafe Database Wizard will launch.
2. Click Next on the Welcome screen. The Database Selection screen will appear, asking for the location of an existing Source Safe database. If you or your organization has already configured a source-control database, select the ''Connect to an existing database'' option. If this is a new installation, check the ''Create a new database'' option (see Figure 3-25).
3. The next step is either to choose an existing source control share location or to create one. After choosing to either use an existing share or to create a new one, the summary screen for the Wizard will appear.
4. Clicking Finish on the Wizard will launch the Visual SourceSafe Explorer. The Visual SourceSafe Explorer can be used to create and manage project folders for both SQL Server Management Studio and Visual Studio solutions.

In a previous example, I created a Management Studio solution called *AdventureWorks Automation*. Now that Visual SourceSafe is configured for use with Management Studio, I can add the solution to the

source-control database to control the modification of the included files and to provide structured version control.

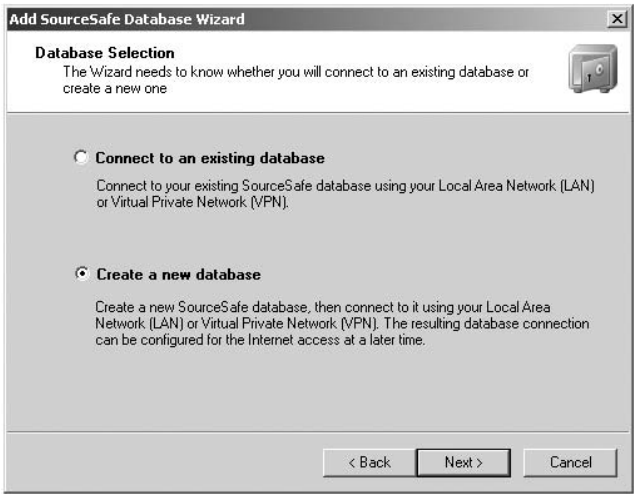


Figure 3-25: Source-control database selection.

Much of the functionality of the Source Control toolbar is only enabled if the current project has already been added to the source-control database.

To add a solution to source control, right-click on the solution in Solution Explorer and select ''Add Solution to Source Control.'' After logging in to source control, choose a location for the solution (see Figure 3-26) and click OK.

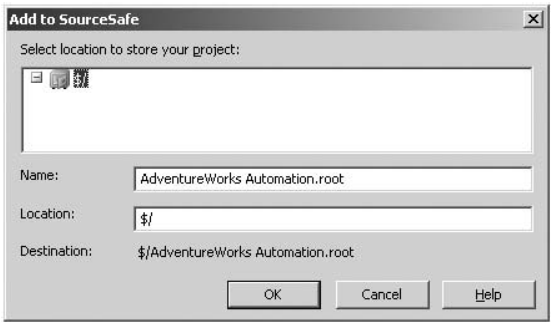


Figure 3-26: Add solution to source control.

Now that the solution has been added to source control, the Source Control toolbar is fully enabled for managing the solution.

The features available on the Source Control toolbar are described in the following table:

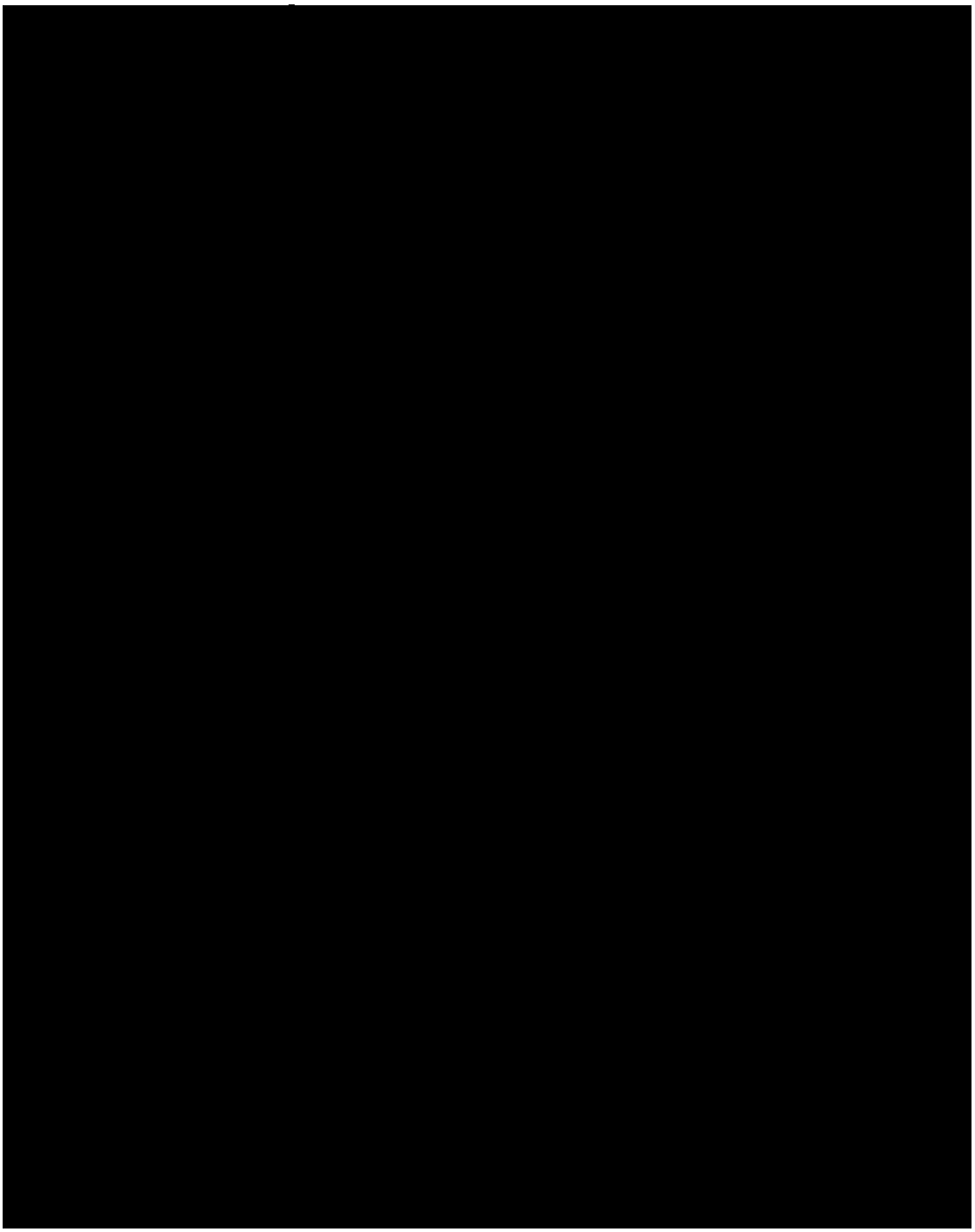
[redacted]

SQL Editor Toolbar

The SQL Editor toolbar (see Figure 3-27) becomes visible (or is enabled if already visible) when a new SQL Query window is opened. It provides the most common features used by SQL programmers and DBAs.

Figure 3-27: SQL Editor toolbar.

The supported features available on the SQL Editor toolbar are described in the following table: }



SQL Server Analysis Services Editors Toolbar

The Analysis Services Editors toolbar (see Figure 3-28) also becomes visible (or is active if already visible) when a new Analysis Services query is opened or created. The tools on this toolbar are a subset of the SQL Editor tools, but contain only those tools applicable to Analysis Services queries (DMX, MDX, XMLA).

Figure 3-28: Analysis Services Editors toolbar.

SQL Server Compact Edition Editor Toolbar

The SQL Server Compact Edition Editor toolbar (see Figure 3-29) becomes visible (or enabled) when a new SQL Compact Edition Query window is opened. The tools on the SQL Server Compact Edition toolbar are a subset of the SQL Editor tools that are applicable for SQL Compact queries.

Figure 3-29: Compact Edition Editor toolbar.

Standard Toolbar

The Standard toolbar (see Figure 3-30) provides buttons to execute the most common actions such as opening and saving files. It also provides buttons that will launch new queries and expose additional tool windows.

Figure 3-30: Standard toolbar.

The commands available on the Standard toolbar are described in the following table:

[redacted]

Continued

[redacted]

Table Designer Toolbar

The Table Designer toolbar (see Figure 3-31) becomes visible (or enabled) when a new table is created using Table Designer or an existing table is modified using Table Designer. Table Designer is launched by right-clicking on the Table node in Object Explorer and choosing New Table from the context menu, or by right-clicking on an existing table in the Table node of Object Explorer and choosing Design.

Figure 3-31: Table Designer toolbar.

The following table describes the toolbar:

[redacted]

Text Editor Toolbar

The Text Editor toolbar (see Figure 3-32) offers additional shortcuts to those provided in the other language-specific editors.

Figure 3-32: Text Editor toolbar.

The features are described in the following table:

[redacted]

Continued

[redacted]

View Designer Toolbar

The View Designer toolbar (see Figure 3-33) is almost exactly like the Query Designer toolbar, with the exception of being limited to writing `SELECT` queries. In addition, queries written with the View Designer are saved as views and not just query scripts. For information about the function of the buttons on the View Designer toolbar, consult the table in the earlier section, ''Query Designer Toolbar.''

Figure 3-33: View Designer toolbar.

XML Editor Toolbar

The XML Editor toolbar (see Figure 3-34) contains several shortcuts that are specific to managing XML files. This is often used with XMLA scripts.

Figure 3-34: Text Editor toolbar.

The features of the Text Editor toolbar are described in the following table:

[redacted]

SQL Server Management Studio Configuration

Management Studio's look and feel can be customized through the Tools ➢ Options menu (see Figure 3-35), which is accessed by selecting Tools on the main menu and clicking Options.

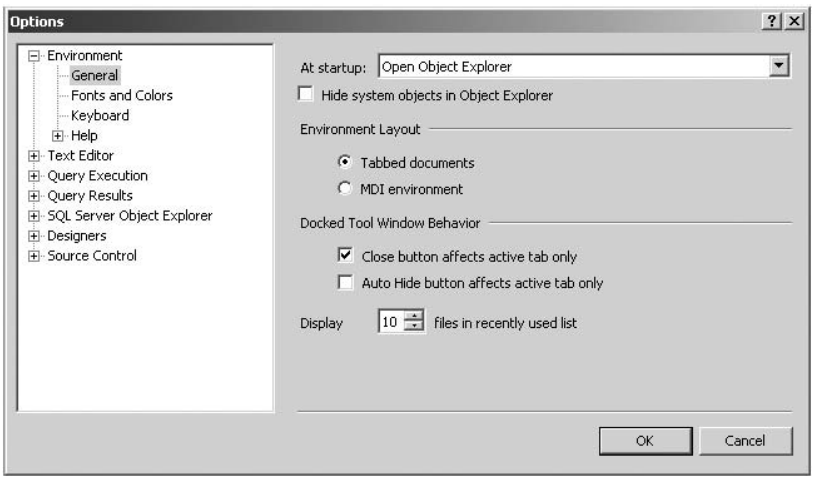


Figure 3-35: Options menu.

The Options dialog enables the customization of the Management Studio IDE. The configuration options are divided into the following seven areas.

Environment

The Environment configuration section is broken down into four subareas:

- **General** — Start-up options and environment layout (such as tabbed windows versus MDI windows) and how the windows behave. Recent file history is also configured on this screen.
- **Fonts and Colors** — The fonts and colors used in the Text Editor are completely customizable in this area. The colors and fonts used for items such as reserved words, stored procedures, comments, and background colors are just a small portion of what can be changed.
- **Keyboard** — For those database administrators who are used to Query Analyzer's keyboard shortcuts, this configuration area enables the setting of the keyboard shortcuts to the same ones used in Query Analyzer. The keyboard configuration area also allows for the addition of custom keyboard shortcuts.
- **Help** — The Help area enables the integration of Help into a Management Studio window or launching Help externally. It also allows for customizing local and online help resources.

Text Editor

The Text Editor section enables the customization of the various Text Editors and is divided into the following six subareas:

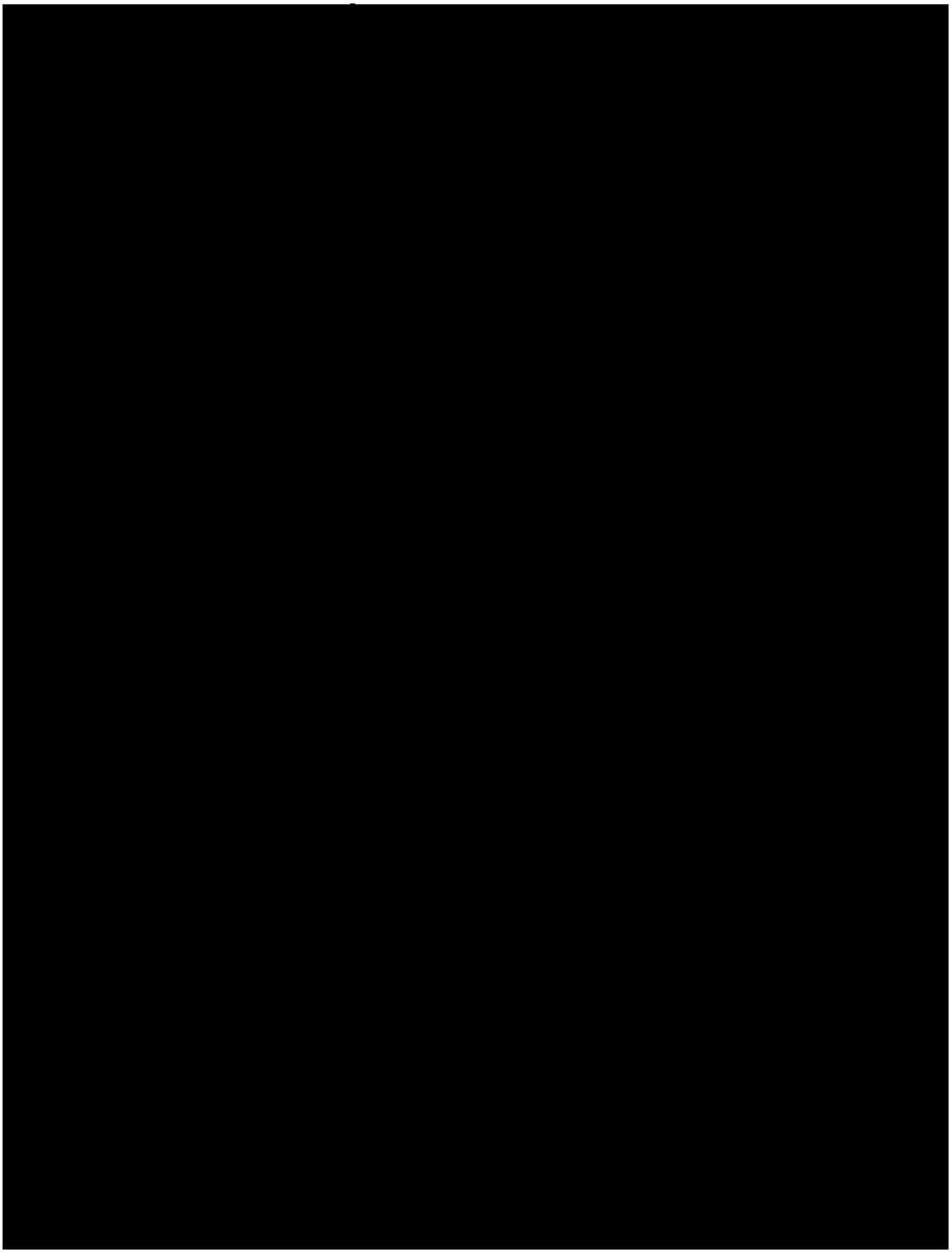
- **File Extension** — File extensions for all the possible script and configuration files can be configured in the File Extension area. Known file extensions such as .sql, .mdx, .dmx, and .xml are not listed, but are automatically associated with their respective editors. They can be reassigned with a ''with encoding'' option so that Management Studio will prompt for specific language encoding every time an associated file type is opened. Custom file extensions can also be added.
- **All Languages** — The All Languages area is divided into two parts, General and Tabs, and provides configuration settings for IntelliSense features, word-wrap, line numbers, and indentation for all script languages.
- **Plain Text** — Configuration settings for plain-text documents not associated with a particular scripting language.
- **Transact-SQL** — Configuration settings specific to T-SQL. There is also a separate tab here for enabling and configuring IntelliSense for Transact-SQL queries.
- **XML** — Configuration settings for XML documents. These settings consist of the same settings from the All Languages area, as well as XML-specific settings such as automatic formatting and schema download settings.
- **Editor Tab and Status Bar** — Configuration settings for the status bar, which is displayed at the bottom of the Query Editor window. You can choose the colors of the status bar, as well as what information is included in the display.

Query Execution

The Query Execution section provides configuration options for how queries are executed, as well as connection properties and time-out settings. The Query Execution section is divided into two primary areas:

- **SQL Server** — The SQL Server area has configuration options that control the maximum row count and the maximum amount of text or Unicode text that is returned to the Management Studio results window. This area also has options to specify a batch delimiter other than `GO` and to specify Query Execution time-out settings. There are also Advanced and ANSI areas that provide for the configuration of specific connection level options described in the following table:

Continued



- **Analysis Services** — Configuration setting to control the execution time-out setting for Analysis Server queries.

Query Results

The Query Results section provides configuration options for how query results are formatted. As with the Query Execution options, this is also divided into two sections for the SQL Server Database Engine and the Analysis Services Engine.

- **SQL Server** — The SQL Server section has configuration options to specify the default location for query results: to a grid, as text, or to a file, as well as the default location for results sent to a file. You can also enable the Windows default beep sound to play whenever a query batch completes. The ''Results to Grid'' settings are described in the following table:

Continued

The ''Results to Text'' settings are described in the following table:

Multi-server Result settings include enabling or disabling adding the login name or server name to the results, as well as merging the results from multiple servers into a single output.

- **Analysis Services** — Configuration settings for Analysis Services query results include showing grids in separate tabs and playing the default Windows beep when the query completes. Both settings are disabled by default.

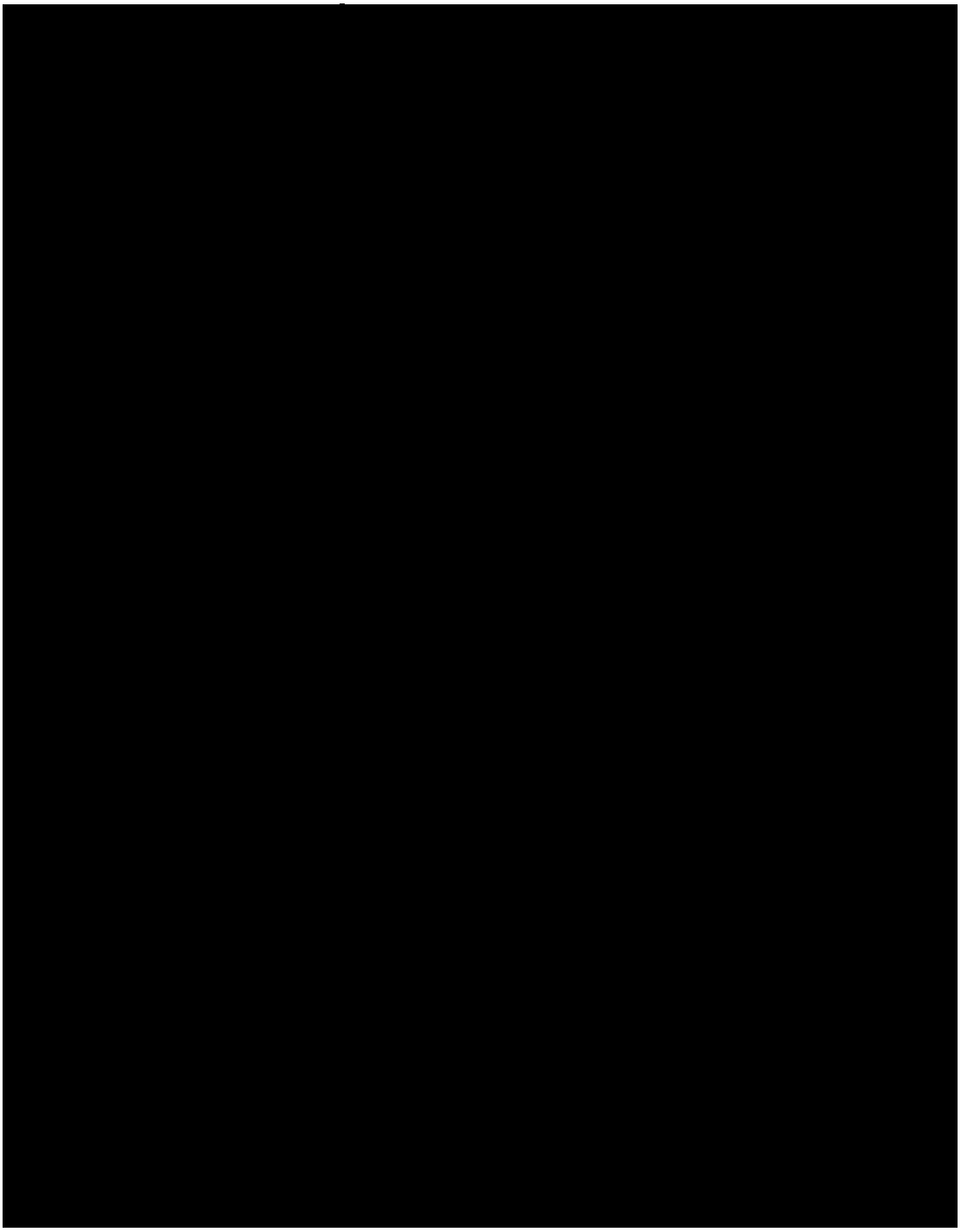
SQL Server Object Explorer

The SQL Server Object Explorer options list contains two tabs, one for managing command settings and another for handling scripting behavior. The command settings allow you to specify the number of rows returned used by the menu items for the number of rows returned for the `Select Top <n> Audit records`, `Edit Top <n> Rows`, and `Select Top <n> Rows` commands.

The configurable scripting options are identified in the following table:

[redacted]

Continued



Designers

The Designers section provides configuration options for the graphical designers used in Management Studio. The Designers section is divided into three subareas:

- **Table and Database Designers** — The Table and Database Designers area allows for the configuration of specific designer behavior. The following table describes the Table options:

The following table describes the Diagram options:

Continued

- **Maintenance Plans** — The Maintenance Plan Designer options determine the way new shapes are added to the maintenance plan design area, including the precedence constraint and positioning of the new shape relative to an existing shape.
- **Analysis Designers** — The Analysis Designers options page provides options to set the connection and query time-out values for the Analysis Designers and the colors for the Data Mining viewers.

Source Control

The Source Control configuration section allows for the integration of a source-control plug-in such as Visual Source Safe 2005. The Source Control section is broken down into three different areas:

- **Plug-In Selection** — Here, the specific plug-in can be chosen (such as Visual Source Safe 2005, or Visual Studio Team System).
- **Environment** — The Environment section allows for the configuration of the Source Control Environment settings supported by the configured source-control plug-in. For Visual Source Safe 2005, there are three preconfigured settings: Visual Source Safe, Independent Developer, and Custom. These settings determine the automatic Check-In and Check-Out behavior of source-control projects.
- **Plug-in Settings** — The Plug-in Settings section provides the ability to customize the source-control actions (such as what to do with unchanged files that have been checked out and how to manage file comparisons and timestamps).

The features available in the Source Control section are dependent on the source control application used. Consult the documentation of the applicable program for more information.

Log File Viewer

The Log File Viewer (see Figure 3-36) is launched from within SQL Server Management Studio. To open it, follow these steps:

1. Expand the Management node in Object Explorer.
2. Expand SQL Server Logs.
3. Right-click on a log, and select ''View SQL Server Log.''

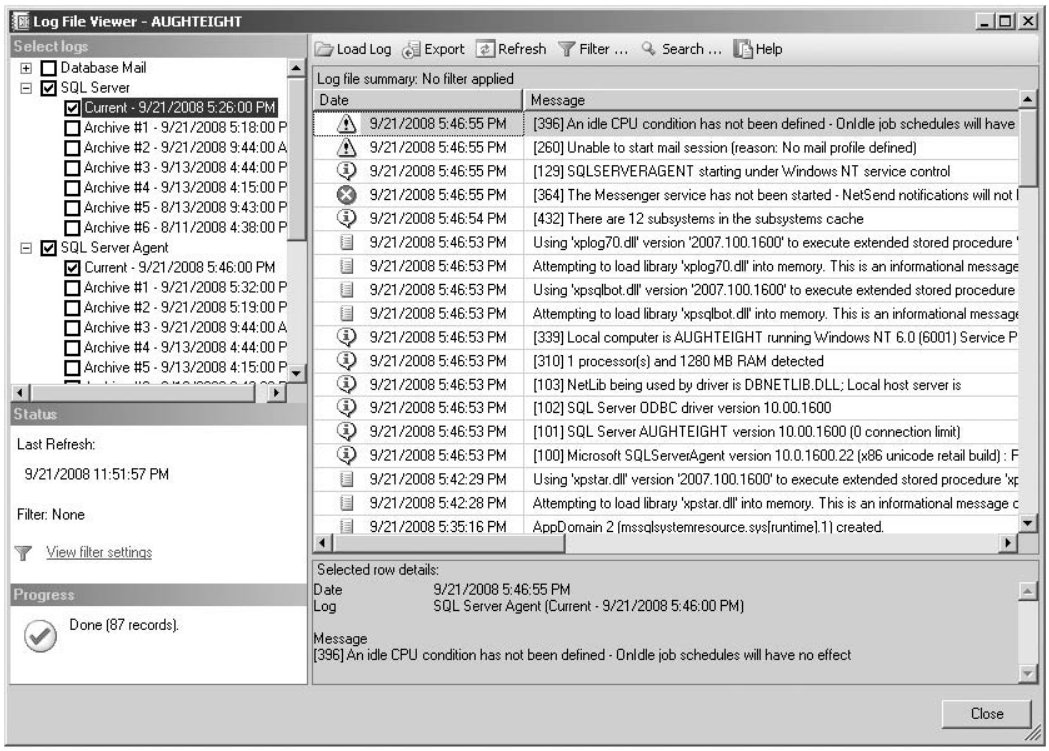


Figure 3-36: Log File Viewer.

One of the benefits of the Log File Viewer is that it allows consolidation of practically all the logs the DBAs are most interested in. SQL Server logs, SQL Agent logs, and Operating System logs can be opened in the same window for easy correlation of system and SQL Server events.

When viewing multiple logs in the Log Viewer, filters can become useful in ensuring that only the information of interest is shown. For example, the filter settings allow the specification of a start date and an end date. Filter settings can also be set to display only those events from a certain subsystem. Applying the appropriate filters helps mitigate the problem of ''Information Overload'' when trying to sift through thousands of log entries.

SQL Server Business Intelligence Development Studio

When SQL Server 2000 Reporting Services was released, Visual Studio was the only way for users to be able to create and manage reports. However, many non-developers were scared off by an interface that was unfamiliar to them. When SQL Server 2005 was released, Microsoft knew they had to respond to user concerns and provide them with a new interface for not only managing reports, but one that could be used for Analysis Services and Integration Services tasks, as well.

Thus, the SQL Server Business Intelligence Development Studio (BIDS) was born. Users could now feel more confident that they had a tool made especially for their Business Intelligence (BI) needs.

In all actuality, BIDS is, in fact, Visual Studio, and SQL Server 2008 includes Visual Studio 2008. Granted, it's not the full Visual Studio 2008, which includes the templates and compilers for Visual Basic, C#, and ASP.NET, but many DBAs were surprised to find Visual Studio installed on their workstation after installing the SQL Server tools. Regardless of whether you launch the Business Intelligence Development Studio shortcut from the SQL Server 2008 folder or the Visual Studio 2008 shortcut from the Visual Studio folder in your Start menu, they launch the exact same application. If the full Visual Studio suite has not been installed, the only available project templates will be Business Intelligence projects. However, if the full suite is installed, all the installed features and templates will be available.

A complete discussion of the Visual Studio IDE is beyond the scope of this book, but a very brief description is definitely in order.

Microsoft has divided Business Intelligence into three distinct pieces: ETL (Extract-Transform-Load), Analysis, and Reporting. These three parts of the Business Intelligence package are implemented through SQL Server Integration Services, SQL Server Analysis Services, and SQL Server Reporting Services. Correspondingly, BIDS provides Business Intelligence project templates that focus on these three areas. The templates are available when creating a new project from BIDS (see Figure 3-37) by selecting File ➢ New Project from the main BIDS menu.

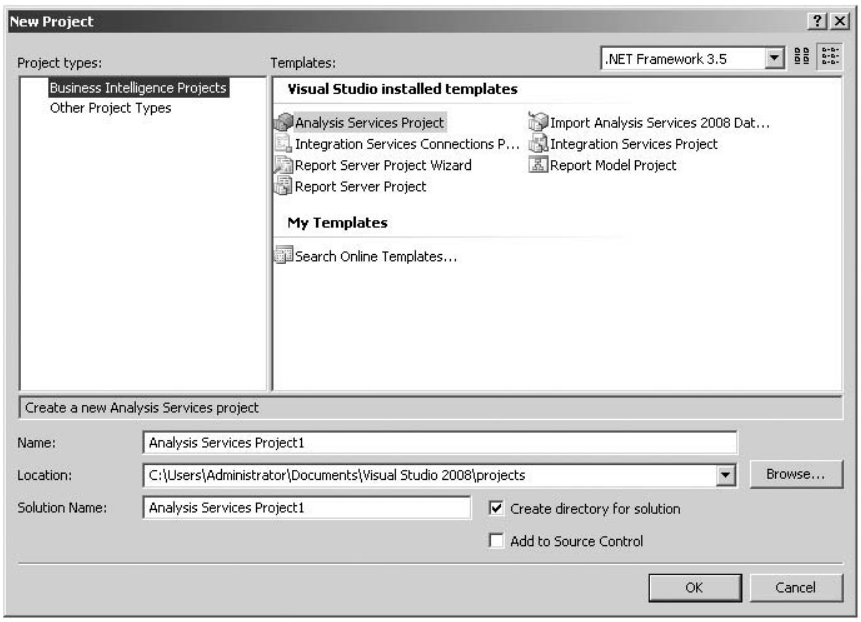


Figure 3-37: Business Intelligence Studio.

Once a template is selected, the template loads with the appropriate tools for the project. The available templates are briefly described in the following table:

SQL Server Profiler

The SQL Server Profiler is an absolutely essential tool for both DBAs and developers alike. Profiler provides the ability to monitor and record virtually every facet of SQL Server activity. It is actually a graphical interface for SQL Trace, which is a collection of stored procedures and functions that are used to monitor and record server activity. SQL Server Profiler can be launched from the Tools menu of SQL Server Management Studio, or from the All Programs ➢ Microsoft SQL Server 2008 ➢ Performance Tools menu.

SQL Server Trace

The Profiler can be used to create and view SQL Server Traces. When creating a new trace, the Profiler will prompt you for the server on which you will be running the trace. Remember that the Profiler is just

a graphical interface for SQL Trace, and what is occurring in the background is the execution of stored procedures and functions on the server you connect to. If the server is very busy and is operating at the edge of its capabilities, the additional load of running SQL Trace on it may well put it over the edge. Profiler and SQL Trace procedures are discussed in greater detail in Chapter 10.

Trace Properties

When creating a new trace, the Trace Properties dialog is shown (see Figure 3-38). The Trace Properties dialog includes two tabs: the General tab and the Events Selection tab. A third tab, Events Extraction Settings, will be enabled if any XML `SHOWPLAN` event is selected in the Events Selection tab.

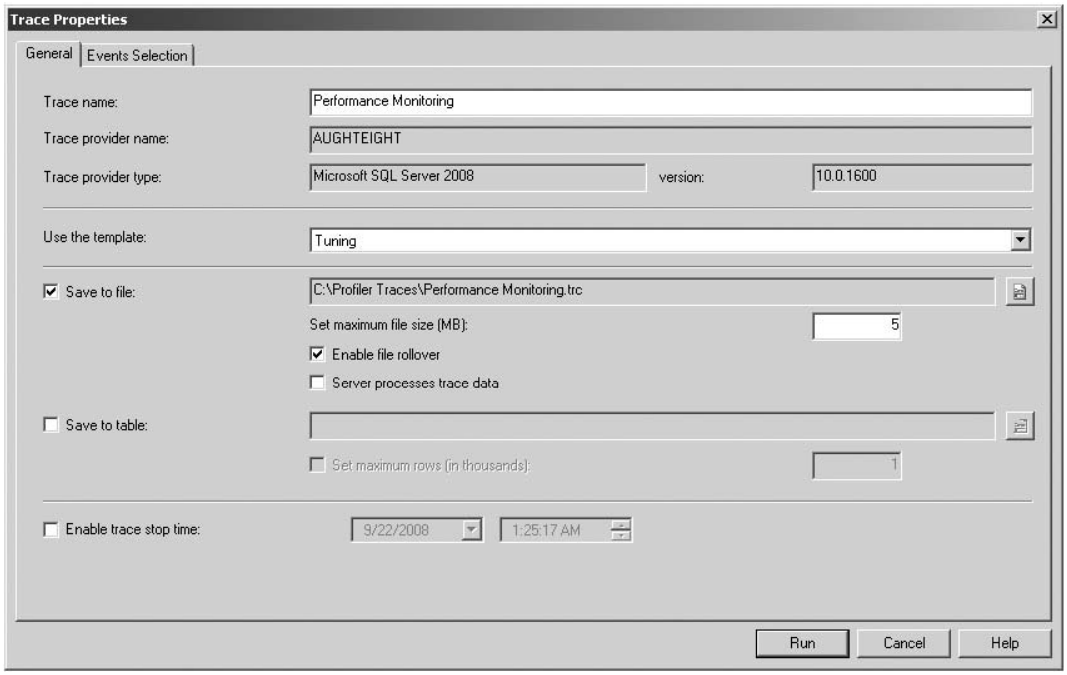


Figure 3-38: Trace Properties dialog.

General Tab

The General tab provides the ability to set the basic structure of the trace (such as the trace name, trace template, saving options, and trace stop time). It also displays the provider name and type, because SQL Server Profiler is not limited to the Data Engine. It can also be used to trace SQL Server Analysis Services.

- **Use the Template** — This dropdown list contains several pre-built trace templates. Each template is a pre-defined set of events and filters that provide for the monitoring of SQL Server for particular purposes. These templates can be a good place to start when creating traces to monitor SQL Server. It is also possible to create your own templates, and it is strongly recommended that you do so. The provided templates are fine, but you will undoubtedly want to collect different information from that which the templates provide. To avoid having to create the same

custom trace over and over again, create and save a template to capture the information you are interested in.

- **Save to File** — Selecting this checkbox will display a dialog prompting for a file location to save the trace data to. The filename defaults to the name assigned to the trace with the .trc extension. However, the name can be changed if desired. The default maximum file size for a trace file is 5 MB, but it can be set to virtually any size. When the ''Save to file'' option is selected, two additional options are enabled: the ''Enable file rollover'' option and the ''Server processes trace data'' option.
  - **Enable File Rollover** — This option causes a new file to be created every time the maximum file size is reached. Each file created is named the same as the original file with a sequential number added to the end of the name. Each sequential file is linked to the preceding file, so that each file can be opened in sequence, or they can all be opened in a single trace window.
  - **Server Processes Trace Data** — This option causes the server that the traces are running on to also process the trace information. By default, the Profiler application processes the trace information. During high-stress operations, if the Profiler processes the data, it may drop some events and even become unresponsive. If the server processes the trace data, no events will be dropped. However, having the server process the trace data and run the trace puts an additional load on the server, which can have a negative impact on server performance.
- **Save to Table** — Trace data can also be saved to a table instead of a file by selecting the ''Save to table'' option. This is very useful if the trace data is going to be analyzed by an external application that requires access to the data stored in a relational format. The down side is that large traces will generate huge amounts of data that will be inserted into the storage table. This can also cause server performance issues, but you can mitigate this by saving trace information to a different server from your production system. If saving trace data to a table, the maximum amount of rows to be stored can also be assigned.
- **Enable Trace Stop Time** — Traces can be started and configured to automatically stop at a pre-defined time by enabling the ''Enable trace stop time'' option and assigning a stop time.

Events Selection Tab

The Events Selection tab provides the ability to choose what SQL Server events are to be traced (see Figure 3-39). Events are grouped in 21 SQL Server event groups with a total of 170 distinct SQL Server events, plus 10 user-definable events. There are also 11 Analysis Services Groups with 38 distinct events. SQL Server Books Online has an excellent reference that describes each group and event. Search for the titles of ''SQL Server Event Class Reference'' for SQL Server events and ''Analysis Services Event Classes'' for Analysis Services Events.

- **Column Filters** — Also in the Events Selection tab is the option to filter the events that are traced (see Figure 3-40). The ability to filter the data is incredibly useful. For example, if you are troubleshooting a particular application, you can filter on just the events generated by the application of interest and avoid having to sift through all the events generated by SQL Server and other applications.

- **Organize Columns** — The Organize Columns button enables you to place the trace columns you are most interested in so that they are easily seen when viewing the trace. Because a great deal of

data can be returned, it may very well be that the column you are most interested in is off the screen to the left. The Organize Columns button helps prevent this.

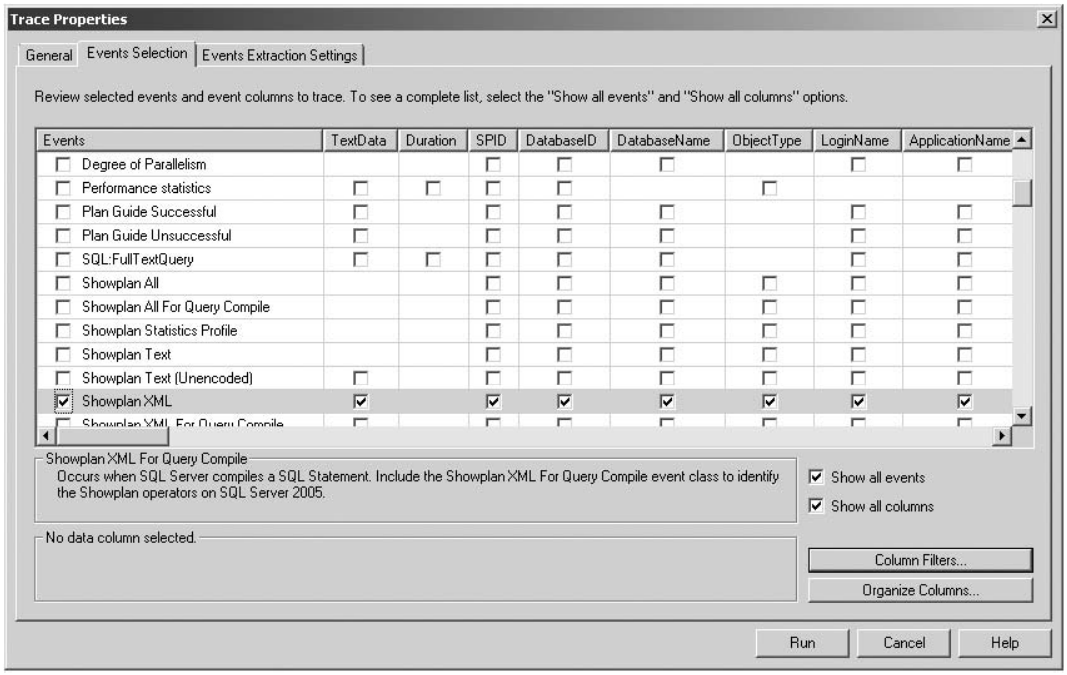


Figure 3-39: Events to be traced.

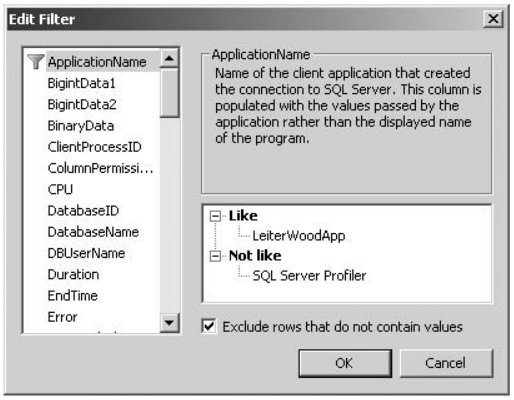


Figure 3-40: Filtering traced events.

Events Extraction Settings Tab

The Events Extraction Settings tab (see Figure 3-41) is enabled when one of the `SHOWPLAN` XML events is chosen from the Performance event group. This tab is divided into two group boxes. The

first provides the ability to save SHOWPLAN information. All SHOWPLAN information can be saved to a single file or multiple XML files that can be opened in SQL Server Management Studio. When opened, they are displayed as graphical execution plans (which are described in detail in Chapter 10). The second group is used for saving graphical deadlock information. Because deadlocks are automatically detected and killed by SQL Server, they are often hard to troubleshoot. SQL Server Profiler provides the ability to trace deadlocks and graphically represent the sequence of events that led to the deadlock.

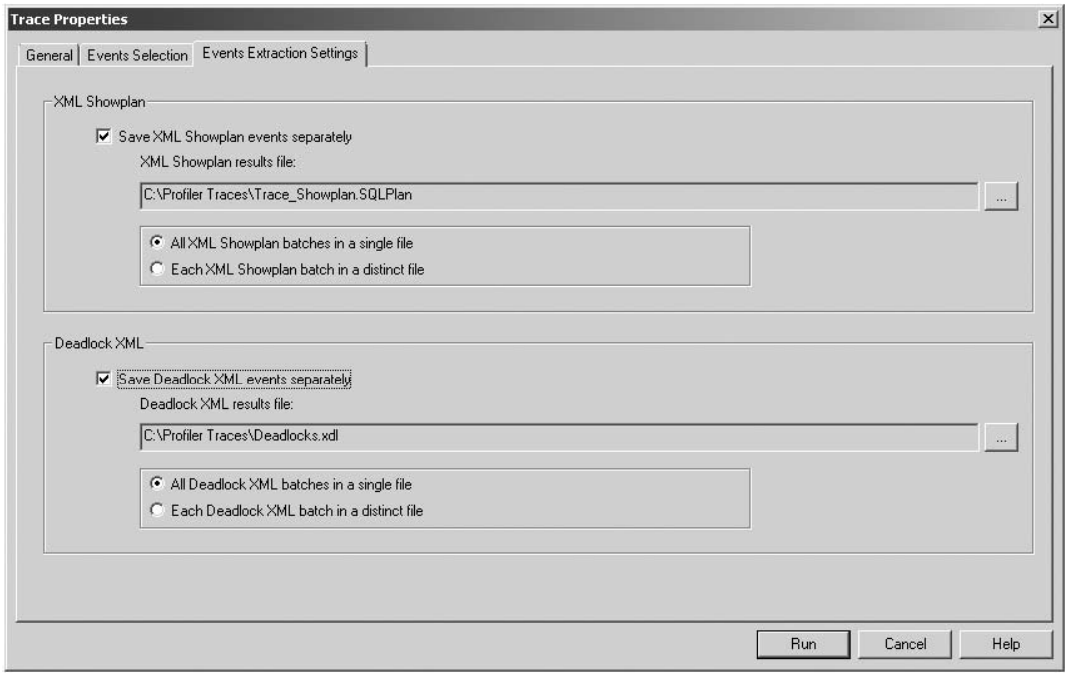


Figure 3-41: Events Extraction Settings.

Chapter 10 describes how to use the SQL Server Profiler to gather pertinent SQL Server data and how to use the profile traces to troubleshoot and optimize SQL Server performance.

Database Engine Tuning Advisor

The Database Engine Tuning Advisor (DTA) can analyze SQL Server scripts or SQL Server Profiler traces to evaluate the effective use of indexes. It can also be used to get recommendations for building new indexes or indexed views, or for creating physical table partitions.

Chapter 11 describes how to use the DTA to help optimize SQL Server databases, so this section is limited to describing the tool and its features. When the DTA is started, it prompts for a server to connect to and then automatically creates a new session. The session is displayed in two tabs: a General tab and a Tuning Options tab.

General Tab

The General tab (see Figure 3-42) is used to define the session name, the workload for analysis, and the database(s) to tune.

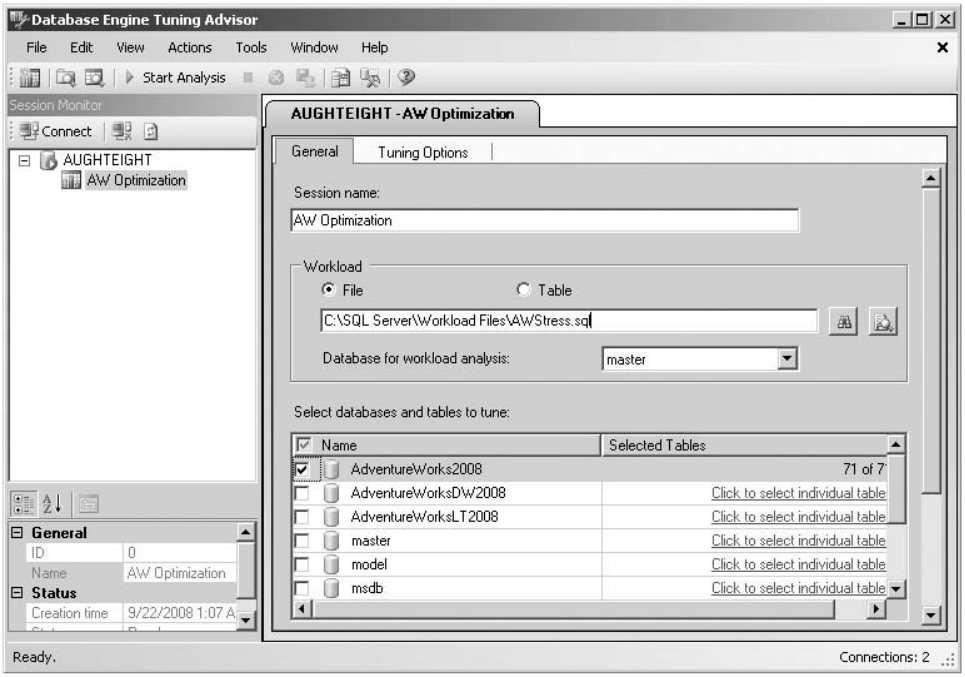


Figure 3-42: DTA General tab.

Following are some options found under this tab:

- **Session name** — By default, the *session name* is the name of the logged-on user combined with the current date and time, but it can (and should) be changed to a more descriptive name.
- **Workload** — The Workload section provides the ability to retrieve trace information from either a file or a table. The table designated must have been previously created by a SQL Server Profiler trace, and the table must be located on the same server the DTA is running on. The file can be a SQL script, a Profiler trace (.trc) file, or a Profiler trace saved as XML.
- **Database for workload analysis** — This option sets the initial connection information for the DTA.
- **Select databases and tables to tune** — In this section, you can designate the database or databases to be tuned. Keep in mind that the more objects chosen to monitor, the bigger the performance impact on the server being monitored. The DTA doesn't actually re-run all the activity from the trace, but it does retrieve a great deal of metadata about the objects contained in the workload, along with any available statistics. This activity alone generates a lot of server activity. Both SQL Server Profiler and DTA activity should be as specific as possible for performance reasons because the more specific the monitoring is, the better the results will be.

Another reason for being specific about choosing the right tables to tune is that if the DTA sees no activity for a table that was selected for monitoring, it will recommend dropping any indexes on that table not associated with a constraint.

Tuning Options Tab

The Tuning Options tab (see Figure 3-43) contains the controls used to configure how the DTA analyzes the workload and what kind of recommendations it will return. At the bottom of the tab is a description box that both describes the individual options and provides feedback for incompatible settings.

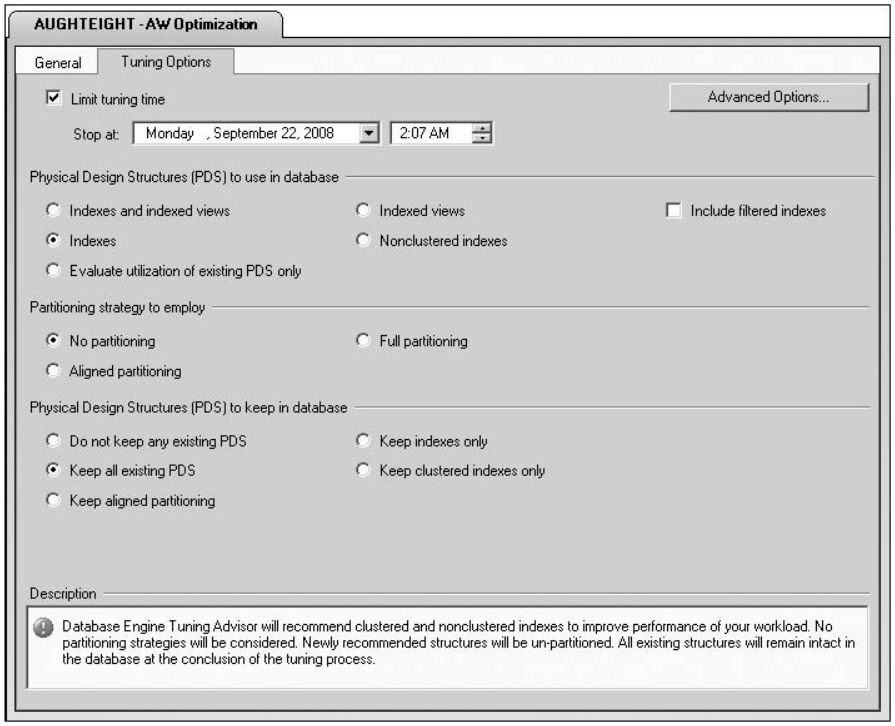


Figure 3-43: Tuning Options tab.

- **Limit tuning time** — Large workloads can take a very long time to fully analyze and can be very expensive in CPU and Database Engine resources. Limiting the amount of time the DTA spends analyzing the workload will cause it to return any recommendations generated with the amount of workload it was able to analyze in the time allocated. For the best results, the DTA should be allowed to run until it has completed; however, that may not always be possible on production systems. Once analysis has started, it can be stopped by clicking the Stop Analysis button on the DTA toolbar.
- **Physical Design Structures (PDS) to use in database** — This option group allows the configuration of the type of PDS recommendations the DTA will return. Options include returning recommendations for the creation of all indexes and indexed views, indexes only, non-clustered

indexes only, and indexed views only. There is also an option for the DTA to only evaluate the effectiveness of current PDS structures, but not recommend the creation of additional structures. Filtered indexes can also be included.

- **Partitioning strategy to employ** — This option group is used to configure the type of physical table partitioning to employ: no partitioning, full partitioning, and aligned partitioning. Physical partitioning is described in Chapter 4.
- **Physical Design Structures (PDS) to keep in database** — When the DTA analyzes workloads, if it determines the PDS structure is not beneficial, it may recommend dropping the structure from the database. This option group is used to configure what PDS structures the DTA will *not* recommend dropping. The DTA can be configured to recommend dropping any non-beneficial PDS structure, to keep indexes only, to not recommend dropping any PDS, to keep clustered indexes only, and to keep any aligned partitioning structure.
- **Advanced Options** — The Advanced Options dialog is used to configure the maximum amount of disk space to use for recommendations, the maximum number of table columns to include per individual index, and online indexing recommendations.

SQL Server Configuration Manager

The SQL Server Configuration Manager is a Microsoft Management Console (MMC) snap-in and is used to manage all the services and protocols employed by an instance of SQL Server. It combines all the functionality that had been in three separate applications — SQL Server 2000's Service Manager, Client Network Utility, and Server Network Utility. The Configuration Manager is divided into three nodes:

- **SQL Server Services** — The Services node offers the similar functionality as the Services applet in the Administrative toolset. However, because it only shows SQL Server services, it is much easier to both control and monitor the status of SQL Server services.
- **SQL Server Network Configuration** — The Network Configuration node displays and enables the configuration of all the available server protocols. The protocols available for use with SQL Server 2008 are Shared Memory, Named Pipes, TCP/IP, and Virtual Interface Adapter (VIA). Protocols that are not in use should be disabled (or left disabled) to minimize the attack surface of the SQL Server.
- **SQL Native Client 10.0 Configuration** — The SQL Native Client Configuration node displays and enables the configuration of the client protocols used to connect to an instance of SQL Server 2008. The configurations only affect the computer that the Configuration Manager is running on. In addition to protocol configuration, the Native Client Configuration node enables the configuration of server aliases.

Reporting Services Configuration Manager

SQL Server 2008 includes an updated Reporting Services Configuration Manager that is more streamlined and easier to use than configuration tools from prior versions. Depending on which options you chose during the installation of SQL Server Reporting Services (''Native Mode,'' ''SharePoint Integrated mode,'' or ''I Will Configure Later'' mode), SQL Server may already be configured and ready to deliver reports.

More information about Reporting Services can be found in Chapter 18. For a thorough discussion of SQL Server 2008 Reporting Services, check out the book Professional Microsoft SQL Server 2008 Reporting Services *by Paul Turley, Thiago Silva, Bryan C. Smith, and Ken Withee (Wiley, 2008).*

Each has its own configuration areas, including the following:

- **Report Server Status** — Selecting the Server name will display the Service Status area, which allows you to monitor the status and stop and start the Reporting Services service. Although this area is called *Server Status*, it is really only the status of the Reporting Services service.
- **Service Account** — This area is used to configure the account under which the Reporting Service runs. Best practices recommend using an Active Directory domain account with the minimum permissions required to run SQL Server Reporting Services. If a domain account is not available, the Network Service account may be used. Local System and Local Service accounts will not work very well, unless SQL Server and Reporting Services are installed on the same computer.
- **Web Service URL** — The Report Server Virtual Directory configuration area enables the viewing or changing of the virtual directory on the SQL Server that hosts the Reporting Services Web Service. Unlike prior versions of Reporting Services, the Web Service does not use IIS. The default virtual directory name is *ReportServer*.
- **Database** — The Database area is used to create or configure SQL Server 2008 Report Server databases. The Report Server databases provide storage of report definitions, report connections, and intermediately rendered reports. The database can be configured in either Native or SharePoint Integrated mode. If you wish to switch database modes, you will have to create a new database with the correct target mode. You can also configure the credentials that are used by the Report Server to connect to the Report Server database. By default, the Service Account for the Reporting Services engine is used.
- **Report Manager URL** — This area is where the virtual directory for the administrative interface, Report Manager, is viewed or configured. This is the Virtual Directory that users will access when creating or managing reports.
- **E-mail Settings** — The SMTP Server settings are very straightforward and simple. However, using the Reporting Services Configuration tool, you can only specify the SMTP server to use and the sender's address. Additional configuration to the e-mail settings must be done manually by editing the Report Server configuration file.
- **Execution Account** — The Execution Account is used when a report needs resources that are not locally available (such as a graphic stored on a remote server). It can also be used to connect to resources that do not require credentials. Configuration of an Execution Account is optional, but may be necessary when accessing shared resources.
- **Encryption Keys** — During the installation of Reporting Services, the installation program automatically generates a symmetric key that is used to encrypt security credentials stored in the Report Server database. To preserve access to this encrypted information, it is critical to back up and restore the key during certain Report Server maintenance procedures. For example, if the database is moved to a different server or the service accounts are changed, the key will have to be restored to preserve access to the encrypted information. The Encryption Keys configuration area provides an easy-to-use graphical interface to back up and restore the keys. It also provides the ability to replace the existing encryption key with a newer one, as well as delete all encrypted

content, in which case, all the stored security credentials would have to be re-entered. In the past, this functionality was provided only through the `RSKEYMGMT` command-line utility, which is still available.

- **Scale-out Deployment** — SQL Server Reporting Services 2008 provides the ability to scale-out Web Service and report access by allowing multiple Reporting Services instances to share a common Report Server database. Scaling-out provides fault tolerance (for front-end services), as well as being able to handle more concurrent connections and specific report execution loads. SSRS is not ''Cluster Aware,'' but can leverage Network Load Balancing (NLB) for Web Services and clustering of the database through a Fault Tolerant Cluster.

Command-Line Tools

SQL Server 2008 comes with plenty of great graphical tools to accomplish almost everything you could ever need to do, but there also comes a time when a simple command-line tool is the best tool for the job. While there are a few command-line tools out there, this section will look at the more prominent ones, which have historically been SQLCMD (and previously OSQL) and BCP, as well as introduce you to Microsoft's newest, and arguably most powerful, command-line utility, PowerShell.

SQLCMD

The SQLCMD utility replaces OSQL as the utility used to execute Transact-SQL statements, Stored Procedures, and SQL script files from the command prompt. OSQL is still available for backward compatibility, but SQLCMD is a more full-featured tool. SQLCMD uses OLE DB to connect to SQL Server and execute Transact-SQL batches.

The SQLCMD utility includes the ability to use variables, connect to servers dynamically, query server information, and pass error information back to the calling environment. Access to the Dedicated Administrator Connection (DAC) is also provided by the SQLCMD utility. The DAC is a special diagnostic connection that can be used by the DBA to connect to a SQL Server server when all other connection types fail to diagnose and correct server problems.

SQLCMD supports several arguments that change the way it behaves and connects to an instance of SQL Server. An abbreviated list is included in the following table. For a complete list of the argument options, consult SQL Server Books Online under the topic ''SQLCMD Utility.'' Unlike other command-line utilities, SQLCMD command-line arguments are case-sensitive.

The SQLCMD utility is typically used to execute saved Transact-SQL scripts in batch processes. This functionality is further enhanced by the ability of SQLCMD to accept scripting parameters. The following code is an example of a SQLCMD script that accepts a parameter called `DBName` to back up a designated database to a file named *DatabasenameDB-Month-Day-Year.BAK* to the C:\SQLBackups folder:

```
DECLARE @BackupDest AS varchar(255)
SET @BackupDest = 'C:\SQLBackups\'
+ '$(DBName)'
+ 'DB-'
+ DATENAME(m,GETDATE())
+ '-'
+ DATENAME(dd,GETDATE())
+ '-'
+ DATENAME(yy,GETDATE())
+ '.BAK'
BACKUP DATABASE $(DBName)
TO DISK = @BackupDest
```

If the preceding script is saved to a file called *BackupDBs.SQL* in the C:\SQLBackups folder, it could be executed to back up the Master database on a server called *AughtEight* using Windows authentication with the following command line:

```
SQLCMD -E -S AughtEight -i C:\SQLBackups\BackupDBs.SQL -v DBName="Master"
```

SQL Server Management Studio makes the creation of SQLCMD scripts even easier with its SQLCMD mode. The BackupDBs.SQL script can be written and tested with Management Studio by selecting SQL-CMD mode in the Query menu. However, to fully test it in the Query Editor, the following command must be inserted in the beginning of the script:

```
:SETVAR DBName "Master"
```

The `SETVAR` command can also be used in the execution of SQLCMD from the command line, but it usually makes more sense to use the `-v` variable argument.

Multiple variables can be set with the `SETVAR` command, as well as passed in to a SQLCMD script with the `-v` argument. The following example shows how to use multiple `SETVAR` commands:

```
USE AdventureWorks2008
GO
:SETVAR ColumnName "LastName"
:SETVAR TableName "Person.Person"

SELECT $(ColumnName)
FROM $(TableName)
```

If the preceding example is saved to a file called *GetContacts.SQL* with the `SETVAR` commands omitted, it would look like the following example:

```
USE AdventureWorks2008
GO

SELECT $(ColumnName)
FROM $(TableName)
```

This script could be executed with the SQLCMD utility using the following command line:

```
SQLCMD -E -S AughtEight -i C:\GetContacts.SQL -v ColumnName="LastName"
 TableName = "Person.Person"
```

Dedicated Administrator Connection (DAC)

SQLCMD is particularly useful for creating batch scripting jobs for administrative purposes. However, as an emergency utility to diagnose and hopefully correct server problems, it has no peer. With the `-A` argument, the SQLCMD utilizes an exclusive connection to SQL Server. If no other connection is possible, the SQLCMD `-A` command is the last and best hope for diagnosing server problems and preventing data loss. By default, only local DACs are allowed because the DAC components only listen on the loopback connection. However, remote DACs can be enabled using the `sp_configure` stored procedure by changing the `remote admin connections` option to `true`, as the following code illustrates:

```
sp_configure 'remote admin connections', 1
RECONFIGURE
```

Bulk Copy Program (BCP)

The BCP utility is mainly used to import flat-file data into a SQL Server table, export a table out to a flat file, or export the results of a Transact-SQL query to a flat file. In addition, it can be used to create format files that are used in the import and export operations.

The syntax of the BCP utility is as follows:

```
usage: bcp {dbtable | query} {in | out | queryout | format} datafile
[-m maxerrors] [-f formatfile] [-e errfile] [-F firstrow] [-L lastrow]
[-b batchsize] [-n native type] [-c character type] [-w wide character type]
[-N keep non-text native] [-V file format version] [-q quoted identifier]
[-C code page specifier] [-t field terminator] [-r row terminator] [-i inputfile]
[-o outfile] [-a packetsize] [-S server name] [-U username] [-P password]
[-T trusted connection] [-v version] [-R regional enable] [-k keep null values]
[-E keep identity values] [-h "load hints"] [-x generate xml format file]
```

BCP format files can be created in two separate formats: XML and non-XML. These files can then be referenced in the import and export of data. The BCP is well-documented in `Books Online`, but the following examples show the most common usage of BCP.

Non-XML Format File Example

This example shows how to begin an interactive BCP session to create a non-XML format file based on an existing table. The BCP utility will prompt for a column data type, a prefix length, and a field delimiter.

It is usually best to accept the defaults provided for the data type and the prefix length because these values are determined by the table being referenced in the BCP command. The delimiter value can be any character, but defaults to ''None.''

The following command uses BCP to create a format file based on the `CreditCard` table in the `AdventureWorks2008` database and `Sales` schema of the local default instance of SQL Server:

```
BCP AdventureWorks2008.Sales.CreditCard format nul -T -f C:\BCP\CreditCard.fmt
```

It is often better to provide the `-S` switch and specify the server name. The `format` argument tells BCP that the desired output is a format file. The absence of an `-x` switch specifies that the output file is not XML. The `nul` argument sends a `NULL` as the username, because the `-T` switch was used indicating that BCP should use a Windows trusted connection. If `-T` is not used, the `-U` username switch is required followed by the `-P` password switch. If `nul` is not used, BCP will fail with the error that a username was not provided.

The result of the preceding command, accepting the defaults for the field data type and prefix length, but entering a comma as the field delimiter, is as follows:

```
10.0
6
1   SQLINT      0     4    ","   1  CreditCardID  ""
2   SQLNCHAR    2     100  ","   2  CardType      SQL_Latin1_General_CP1_CI_AS
3   SQLNCHAR    2     50   ","   3  CardNumber    SQL_Latin1_General_CP1_CI_AS
4   SQLTINYINT  0     1    ","   4  ExpMonth      ""
5   SQLSMALLINT 0     2    ","   5  ExpYear       ""
6   SQLDATETIME 0     8    ","   6  ModifiedDate  ""
```

The `10.0` at the top of the results designates the version of BCP. ''10.0'' is SQL Server 2008, and ''9.0'' would be SQL Server 2005. The number `6` under the `10.0` specifies how many columns are in the file. Following the column number is the SQL Server data type of the column, followed by the number of bytes needed by the prefix length. The prefix length of a column depends on the maximum number of bytes, whether the column supports `NULL`s, and the storage type.

If the BCP command is supplied a data format argument (`-c` or `-n`), it will output a format file with all columns mapped to the supplied format without any interaction.

XML Format File Example

This example shows how to use the BCP command to generate an XML format file:

```
BCP AdventureWorks2008.Sales.CreditCard format nul -x -T -f C:\BCP\CreditCard.xml
```

As you can see, the syntax is identical, except that the `-x` switch is used to specify an XML output. The result is as follows:

```
<?xml version="1.0"?>
<BCPFORMAT xmlns=http://schemas.microsoft.com/sqlserver/2004/bulkload/format
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RECORD>
  <FIELD ID="1" xsi:type="NativeFixed" LENGTH="4"/>
  <FIELD ID="2" xsi:type="NCharPrefix" PREFIX_LENGTH="2" MAX_LENGTH="100"
```

```
        COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="3" xsi:type="NCharPrefix" PREFIX_LENGTH="2" MAX_LENGTH="50"
        COLLATION="SQL_Latin1_General_CP1_CI_AS"/>
  <FIELD ID="4" xsi:type="NativeFixed" LENGTH="1"/>
  <FIELD ID="5" xsi:type="NativeFixed" LENGTH="2"/>
  <FIELD ID="6" xsi:type="NativeFixed" LENGTH="8"/>
 </RECORD>
 <ROW>
  <COLUMN SOURCE="1" NAME="CreditCardID" xsi:type="SQLINT"/>
  <COLUMN SOURCE="2" NAME="CardType" xsi:type="SQLNVARCHAR"/>
  <COLUMN SOURCE="3" NAME="CardNumber" xsi:type="SQLNVARCHAR"/>
  <COLUMN SOURCE="4" NAME="ExpMonth" xsi:type="SQLTINYINT"/>
  <COLUMN SOURCE="5" NAME="ExpYear" xsi:type="SQLSMALLINT"/>
  <COLUMN SOURCE="6" NAME="ModifiedDate" xsi:type="SQLDATETIME"/>
 </ROW>
</BCPFORMAT>
```

Export a Table to a Flat-File Example

Once the format file is created, it can be used to control data export and import operations. To export data to a delimited flat file using the XML format file created in the preceding example, execute the following code:

```
BCP AdventureWorks2008.Sales.CreditCard OUT C:\BCP\CreditCard.dat -T
       -f C:\BCP\CreditCard.XML
```

Import Flat-File Example with a Format File

To test a BCP import, first create a copy of the `CreditCard` table with the following script:

```
USE AdventureWorks2008
GO
SELECT * INTO Sales.CreditCard2
FROM Sales.CreditCard
TRUNCATE TABLE Sales.CreditCard2
```

Once the destination table exists, the flat file and XML format file can be used to import the data to the new `CreditCard2` table with the following code:

```
BCP AdventureWorks2008.Sales.CreditCard2 IN C:\BCP\CreditCard.dat -T
        -f C:\BCP\CreditCard.xml
```

PowerShell

PowerShell is a new command-line shell designed for Systems Administrators. PowerShell is both an interactive console and a scripting interface that can be used to automate many common administrative tasks. A complete explanation of PowerShell is beyond the scope of this book, but this section will provide a summary of how PowerShell can be used for SQL Server Administration.

PowerShell is designed around the Microsoft .NET Common Language Runtime (CLR) and the .NET Framework. It exposes .NET objects through a series of *cmdlets*, which are single-feature commands that interact with objects. Cmdlets are formatted using a verb–noun structure, separated by a hyphen, such as

`Get-Process`, which returns a list of the current running processes. Another benefit of PowerShell is that cmdlets can be piped into one another; for example, you might invoke one command to retrieve information and then use a pipe character (|) on the same line to invoke another command that performs an action or controls formatting and output of the results of the first command. You can pipe several commands as a single function. PowerShell can be installed on Windows XP SP2, Windows Vista, Windows Server 2003, and comes installed in Windows Server 2008.

SQL Server 2008 supports PowerShell for SQL Administration, and, in fact, if it's not already installed on the system, the SQL Server installer will install it for you. SQL Server administrators can use PowerShell to administer any SQL Server running SQL Server 2008, SQL Server 2005, and even SQL Server 2000 SP4, although functionality will be limited.

SQL Server 2008 also includes a limited-functionality shell known as *sqlps*. The sqlps utility is designed to expose access to SQL Server objects and cmdlets automatically, but it is configured to run with a *Restricted* execution policy, which prevents PowerShell scripts from running. This can be changed, if necessary.

It may be preferred to access SQL Server objects from a standard PowerShell environment, in which case, you can create a PowerShell console that includes the snap-ins for SQL Server. The following example shows you how to create a new PowerShell console named *MySQLPS.psc1* with the SQL Server snap-ins, to a new folder called *PSConsoles*. In this case, PowerShell is being invoked from the Run command in the Start Menu.

```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> md C:\PSConsoles


    Directory: Microsoft.PowerShell.Core\FileSystem::C:\


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----         9/23/2008   9:33 AM            PSConsoles


PS C:\Users\Administrator> cd c:\PSConsoles
PS C:\PSConsoles> add-pssnapin SqlServerProviderSnapin100
PS C:\PSConsoles> add-pssnapin SqlServerCmdletSnapin100
PS C:\PSConsoles> Export-Console -Path MySQLPS.psc1
```

Now that you've created the new console, you can double-click on the file in Windows Explorer, or you can manually invoke the console using the following command from the Run command:

```
powershell.exe -psconsolefile C:\PSConsoles\MySQLPS.psc1
```

So what can you do with PowerShell? Quite a bit, actually. The Object Explorer includes the ability to launch PowerShell using the right-click context menu. The PowerShell console invoked is location-aware, meaning that when you right-click the `HumanResources.Employee` table in the `AdventureWorks2008` database and choose Start PowerShell from the menu, it will start `sqlps` using that object as the current path.

```
PS SQLSERVER:\SQL\AUGHTEIGHT\DEFAULT\Databases\AdventureWorks20
08\Tables\HumanResources.Employee>
```

Output Data using PowerShell

In this exercise, you will use PowerShell to invoke a SQLCMD function.

1. In Object Explorer, navigate to Server ➢ Databases ➢ AdventureWorks2008.
2. Right-click on the AdventureWorks 2008 database, and select Start PowerShell (Figure 3-44).

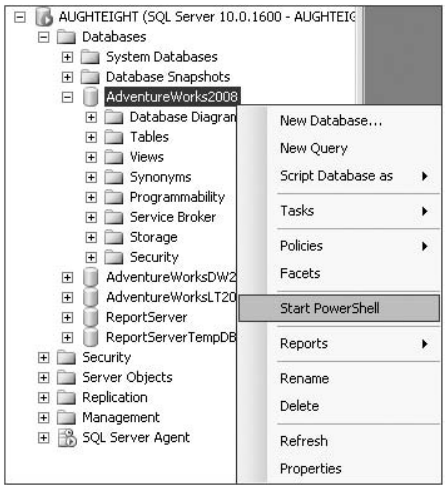
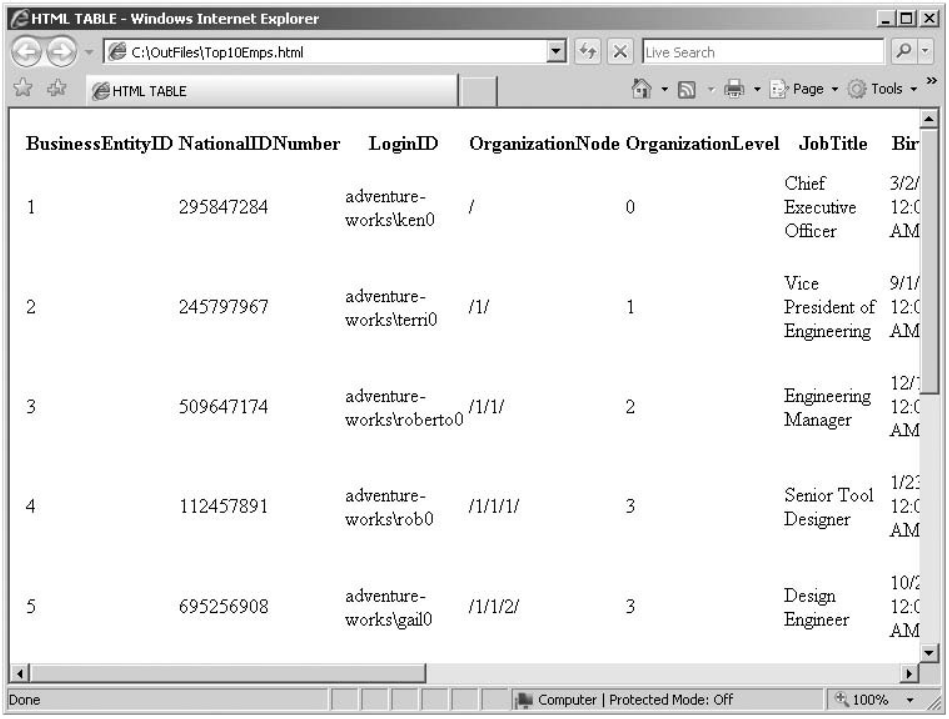


Figure 3-44: Start PowerShell.

3. Type the following commands into the PowerShell shell:

```
md C:\OutFiles
invoke-SQLCMD "Select TOP 10 * from HumanResources.Employee" |
    ConvertTo-html | Out-file C:\OutFiles\Top10Emps.html
```

4. Navigate to C:\OutFiles, and double-click on the Top10Emps.html file. In IE, you should see the HTML-formatted output shown in Figure 3-45.



BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	Bir
1	295847284	adventure-works\ken0	/	0	Chief Executive Officer	3/2/ 12:0 AM
2	245797967	adventure-works\terri0	/1/	1	Vice President of Engineering	9/1/ 12:0 AM
3	509647174	adventure-works\roberto0	/1/1/	2	Engineering Manager	12/ 12:0 AM
4	112457891	adventure-works\rob0	/1/1/1/	3	Senior Tool Designer	1/2 12:0 AM
5	695256908	adventure-works\gail0	/1/1/2/	3	Design Engineer	10/ 12:0 AM

Figure 3-45 HTML formatted output.

At this point, you've barely scratched the surface of what PowerShell can do, but because the entire set of SQL Management Objects (SMO) is exposed to PowerShell, administration of your SQL Server and its databases can be made much easier by automating and scripting many common processes.

Summary

This Chapter described the primary tools that are used by DBAs. Some of the less-frequently used tools were covered briefly, as they are not often used by the DBA, but instead by an application or Business Intelligence developer. By this point, you should have a good understanding of the tools available to you in your role as a DBA and how to customize those tools to meet your needs.

The most prominent of these tools is, of course, SQL Server Management Studio. SQL Server Management Studio for SQL Server 2008 includes many new and compelling features, such as IntelliSense for T-SQL, PowerShell integration, and expanded toolsets. However, DBAs should also be familiar with Business Intelligence Development Studio (BIDS) and SQL Server Configuration Manager.

Throughout this book, you will be using the tools described in this Chapter to build and manage SQL Server 2008 databases. Having a solid understanding of the tools available will make it easier to perform these tasks.

Chapter 4 describes how SQL Server stores its data physically and logically. It describes the physical architecture of data and log files, as well as how SQL Server manages these files. Since Disk I/O is often the slowest part of any SQL environment, understanding how SQL Server stores and accesses data will be the key to maintaining a solid infrastructure.