



## Chapter 14

# Developing Desktop Applications with SQL Server Express Edition

—Rob Walters

### In this chapter:

What Is SQL Server Express Edition? .....	427
Configuration .....	435
Working with SQL Server Express Edition .....	437
Installing SQL Server Express Edition .....	451
Summary .....	485

As time passes, database engines that perform basic relational database actions are becoming more of a commodity. With the advent of MySQL, Postgres, and a number of other open source database vendors, more options exist today for developers than ever before. The impact of these disruptive technologies can be seen in the free database offerings not only from Microsoft but also from Oracle (Oracle Express) and IBM (DB2 Express). Each of these relational database vendors wants you, the developer, to use their product so that one day your application might grow enough that you actually need to buy a license for an “upper-level” edition. Although these marketing tactics might be of little interest to us as developers, in reality we benefit tremendously from this competitive environment. Could you ever have imagined 10 years ago that companies like Microsoft would essentially give away a powerful developer tool like Visual Studio Express? Bundled with a rock-solid database like SQL Server, it solves a lot of development needs for a reasonable price: free.

This chapter covers SQL Server Express Edition. It goes into depth about installation, configuration, and management, and it explores Express Edition’s features. It also compares and contrasts Express Edition with the Microsoft SQL Server Desktop Engine (MSDE). Finally, you will learn how Express Edition fits into the overall SQL Server product lineup.

## What Is SQL Server Express Edition?

Express Edition is one of four SQL Server editions. The others are Workgroup, Standard, and Enterprise. Figure 14-1 shows the progression of editions from Express to Enterprise.

Express	Workgroup	Standard	Enterprise
<i>Fastest way for developers to learn, build, &amp; deploy simple data driven applications</i>	<i>Easiest to use &amp; most affordable database solution for smaller departments &amp; growing businesses</i>	<i>Complete data management &amp; analysis platform for medium businesses and large departments</i>	<i>Fully integrated data management and analysis platform for business critical enterprise applications</i>
1 CPU 1 GB RAM 4 GB Database Size	2 CPU 3 GB RAM	4 CPU Unlimited RAM (64bit)	Unlimited Scale + Partitioning
Management Tool Reporting Replication Service Broker Full Text	Management Tools Import/Export Limited Replication Publishing Back-up Log-shipping	Database Mirroring OLAP Server Report Server New Integraton Services Data Mining Full Replication	Database mirroring, Complete online & parallel operations. Database snapshot Advanced Analysis Tools including full OLAP % Data Mining Customized & High Scale Reporting

**Figure 14-1** The four editions of SQL Server 2005

Some people might be wondering, “Where is Developer Edition?” The Developer and Evaluation editions are actually the same as Enterprise Edition, but with some licensing restrictions. Each SQL Server edition builds on the previous one’s functionality. So Standard Edition, for example, has the same functionality as Express Edition and Workgroup Edition (such as management tools, the ability to import and export, and so on).

Note that the Express database engine itself is the same as those of the other editions. It differs only in the restrictions imposed on its use: 1 CPU, 1 GB of RAM, and 4 GB database file size. There are some interesting things to note about these restrictions. If you have a multi-core CPU, Express Edition will leverage all the cores. (After all, to the SQL engine it’s still just one CPU.) If you exceed the database file size limit, you will get an error on the command that caused the size to exceed the limit and any transactions in that session will be rolled back. This is the same behavior you would see if you set a fixed size for a database, the database almost reached that size, and then you tried to insert one more row, causing it to exceed the limit. One of the restrictions that existed previously in MSDE was the workload governor. This “feature” intentionally decreased performance as the workload increased. You can forget about this in Express Edition—there is no such thing, so you can throw as much work at Express Edition as your box can handle (and as the CPU, RAM, and database size restrictions allow). Express Edition truly is the same database engine as all the other editions.

## Licensing

Some in the developer community might be thinking about deploying Express Edition with their own applications. This is possible, legal, and royalty-free as long as you do not change any parts of the shipped bits of Express Edition. You must also register on Microsoft's Web site if you want to redistribute Express Edition. The URL is <http://go.microsoft.com/fwlink/?LinkId=64062>.

Later in this chapter, we will cover how to silently install Express Edition—this is the most likely way you will deploy Express Edition with your application. In addition to the database engine, Express Edition contains a few extra components such as SQL Server Management Studio Express and SQLCMD. All these tools are also redistributable as long as you don't change them. You cannot redistribute the tools alone, however—you must redistribute Express Edition in addition to the tools. You can, however, redistribute just the SQL Server Express Edition engine.

## Feature Review

With the exception of the memory, database size, and CPU restrictions, SQL Server Express Edition contains all the basic relational database features seen in the other editions. For example, you can create common language runtime (CLR) stored procedures, store data as XML, and encrypt your data using symmetric keys. However, Microsoft did not give away the farm in this edition—if you want to use more “enterprise-like” features such as database mirroring or partitioning, you have to pay for the upper-level edition. The business intelligence tools (such as SQL Server Analysis Services) and extract, transform, and load (ETL) tools (such as SQL Server Integration Services, known as Data Transformation Services in SQL 2000) are also not available in Express Edition.

The lack of Integration Services in Express Edition causes some minor pain. For one thing, there is no Import And Export Wizard for moving a database nor any relatively easy alternative. The easiest workaround is to perform a detach, file copy, and attach. Alternatively, you can back up and then restore a database on the destination server. If you need just specific tables within a database, you might consider using the Bulk Copy Program (BCP), which ships with Express Edition as well as all the other editions of SQL Server.

Now let us focus on the database engine-specific features that are significant to SQL Server Express Edition users. Table 14-1 shows key database administration features for the SQL Server product as described by the folks in SQL Server marketing. This table includes a column that describes the availability of the feature in Express Edition.

**Table 14-1 SQL Server Database Engine Features**

Feature	Available in Express Edition?
Database Mirroring	No
Online Restore	No
Online Indexing Operations	Yes
Fast Recovery	Yes

**Table 14-1 SQL Server Database Engine Features**

Feature	Available in Express Edition?
Security Enhancements	Yes
New SQL Server Management Studio	Available as a scaled-down version called SQL Server Management Studio Express.
Dedicated Administrator Connection	Yes, but it is not enabled by default. To use it in Express Edition, you must start the service using the trace flag 7806. You do not need to specify this trace flag if you are using any other edition of SQL Server.
Snapshot Isolation	No
Data Partitioning	No
Replication Enhancements	Available but scaled down. Later in this chapter, see the “Replication in Express Edition” section for additional details.
SQL Server Agent	No. The workaround is to create Transact-SQL (T-SQL) scripts that are executed via SQLCMD through the Windows Task Scheduler.
SQL Mail and Database Mail	No
Mirrored Media Sets	No
Address Windowing Extensions (AWE)	No
Hot-add memory	No
Failover clustering	No
VIA Protocol support	No

Table 14-2 shows the key development features of SQL Server as well as their availability in Express Edition.

**Table 14-2 SQL Server Database Development Features**

Feature	Available in Express Edition?
.NET Framework Hosting	Yes. This feature is off by default in all SQL Server editions, so you must enable the CLR by selecting the Enable CLR check box in the SQL Server Surface Area Configuration tool.
XML Technologies	Yes
ADO.NET 2.0	Yes
T-SQL Enhancements	Yes
SQL Service Broker	Available but scaled down. Later in this chapter, see the “Service Broker in Express Edition” section for additional details.
Notification Services	No
Web Services	No. You cannot create HTTP endpoints in Express Edition, but it is still possible to create a Web service and use Express Edition—you just can’t make Express Edition an HTTP listener.
Reporting Services	Yes. Later in this chapter, see the “Reporting Services in Express Edition” section for additional details.
Full-Text Search Enhancements	Yes. Later in this chapter, see the “Full Text in Express Edition” section for additional details.

## Replication in Express Edition

Express Edition can serve as a Subscriber for all types of replication, providing a convenient way to distribute data to client applications that use SQL Server Express Edition. Note that this is a change in behavior from MSDE. If you are using MSDE for replication, MSDE can be either a Snapshot Publisher or a Merge Publisher. If you want to continue to use this functionality, you must upgrade to at least Workgroup Edition.

SQL Server Express Edition does not include the SQL Server Agent, which is typically used to run replication agents. If you use push subscription, replication agents run at the Distributor, which will be an instance of SQL Server 2005, so options are available for synchronizing. But if you use a pull subscription, in which agents run at the Subscriber, you must synchronize the subscription by using the Windows Synchronization Manager tool or do it programmatically using Replication Management Objects (RMO).

Windows Synchronization Manager is available with Microsoft Windows 2000 and later. If SQL Server is running on the same computer as Synchronization Manager, you can do the following:

- Synchronize a subscription.
- Reinitialize a subscription.
- Change the update mode of an updatable transactional subscription.

## Service Broker in Express Edition

Service Broker is a new technology in SQL Server 2005 that helps developers create distributed applications that provide support for queuing and reliable messaging. Developers can compose applications from independent, self-contained components called services. Applications can then use messages to interact with these services and access their functionality. Service Broker uses TCP/IP to exchange messages between SQL Server instances, and it includes features to help prevent unauthorized access from the network and to encrypt messages sent over the network.

The SQL Server Express Edition database engine supports Service Broker only as a client. Express Edition can participate in a Service Broker messaging application only when a paid edition of SQL Server (Workgroup, Standard, or Enterprise) is part of the message chain. Express Edition can also send Service Broker messages to itself.

## SQL Server 2005 Express Edition with Advanced Services

At the time SQL Server 2005 Express Edition shipped, Microsoft promised that it would soon release a version of Reporting Services as well as Full-Text Search capabilities for Express Edition. The time has come—with the release of Service Pack 1 (SP1) also comes the release of SQL Server Express Edition with Advanced Services, SQL Server Express Edition Toolkit, and SQL Server Management Studio Express.

## Express Edition Download Page

SQL Server 2005 Express Edition with Advanced Services is available for download from the main SQL Server Express Edition page. The URL is <http://msdn.microsoft.com/vstudio/express/sql/download/>.

On the download page for Express Edition, you have a number of options, depending on exactly what features you need. The following are the available downloads:

- **SQL Server Express Edition with SP1 (SQLEXPRESS.exe)** This download gives you just the Express Edition database engine and connectivity tools. It does not include any management tools or any of the advanced services such as Full-Text Search or Reporting Services.
- **SQL Server Express Edition with Advanced Services (SQLEXPRESS\_ADV.exe)** This download gives you the Express Edition database engine plus additional components that include SQL Server Management Studio Express, support for full-text catalogs, and support for viewing reports via Report Server. SQL Server Express Edition with Advanced Services also includes SP1. It's a bit tricky if you already have Express Edition installed from RTM: Installing this component is just like installing another instance of Express Edition. During the setup of Express Edition with Advanced Services, if you choose not to upgrade your current Express Edition installation, you will end up with two installations of Express Edition on your box.
- **SQL Server Express Edition Toolkit (SQLEXPRESS\_TOOLKIT.exe)** SQL Server 2005 Express Edition Toolkit provides tools and resources to manage SQL Server Express Edition and SQL Server Express Edition with Advanced Services. It also allows you to create reports using SQL Server 2005 Business Intelligence Development Studio. This download does not include the database engine; it includes only the management tools.
- **SQL Server Management Studio Express (SQLServer2005\_SSMSEE.msi)** SQL Server Management Studio Express provides a graphical management tool for managing SQL Server 2005 Express Edition and SQL Server 2005 Express Edition with Advanced Services instances. Management Studio Express can also manage relational engine instances created by any edition of SQL Server 2005. It cannot manage Analysis Services, Integration Services, SQL Server 2005 Mobile Edition, Notification Services, Reporting Services, or SQL Server Agent. This download is just for the Management Studio Express tool; it does not contain the database engine or the reporting services designer.

In summary, if you want to install Express Edition with all the bells and whistles, just download and install SQL Server Express Edition with Advanced Services (SQLEXPRESS\_ADV.exe) followed by SQL Server Express Edition Toolkit (SQLEXPRESS\_TOOLKIT.exe). This will get you all the components that Express Edition has to offer.

## Reporting Services in Express Edition

Reporting Services in SQL Server Express Edition is a server-based solution that enables the creation and management of reports derived from data stored in Express Edition. The full version of Reporting Services found in other editions provides many more features—such as the ability to schedule deployment of reports and create subscriptions—but even without these advanced features, Express Edition users have a powerful enough database engine and toolset to conquer many scenarios.

To use Reporting Services within Express Edition, you must have Microsoft Internet Information Services (IIS) installed before you install Express Edition.



**Note** If you installed the Microsoft .NET Framework before installing IIS, you might receive an error in the Express Edition setup program asking you to reinstall the .NET Framework. To avoid rolling back and reinstalling the .NET Framework, run the following on the command line and then click the Retry button.

```
Aspreg_iis.exe -i
```

This .exe file is located in the .NET Framework folder under <installation drive>:\WINDOWS\Microsoft.NET\Framework\v2.0.50727.

Reporting Services can be considered a two-part installation. The Reporting Server service and Web-based administration page can be installed as an option with SQL Server Express Edition with Advanced Services (SQLEXPRESS\_ADV.exe). If you want to actually create reports and do not want to modify XML using Notepad, you might want to also install BI Development Studio from the SQL Server Express Edition Toolkit (SQLEXPRESS\_TOOLKIT.exe). BI Development Studio is a Visual Studio application that allows you to create reports using a full-featured wizard or by manually building reports using a designer surface (Figure 14-2).

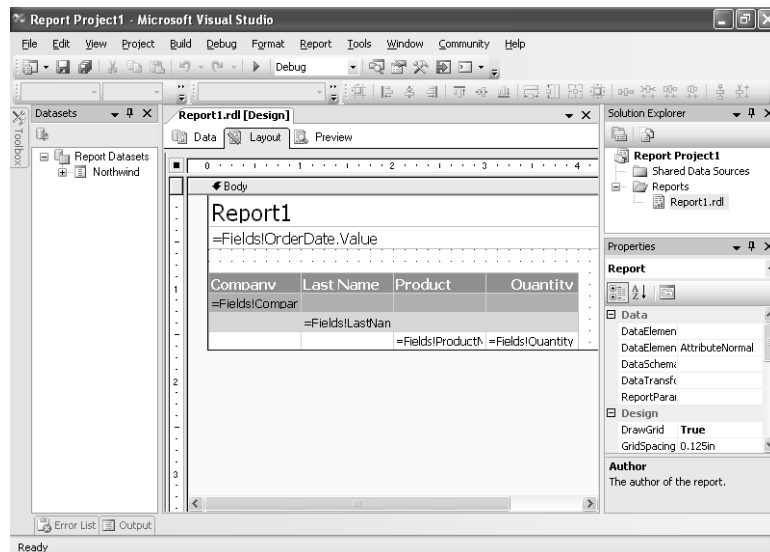


Figure 14-2 Report designer in SQL Server BI Development Studio

During the setup of SQL Server Express Edition with Advanced Services, if you opt to include Report Server, you are presented with some additional forms asking whether you want to configure the Report Server now or later. If you install the report server without configuring it, you can use the Reporting Services Configuration Manager to perform the same configuration work. You must configure the Report Server before you can deploy any reports (Figure 14-3).



Figure 14-3 Configure Report Server tool



**Note** If you instructed setup to automatically configure Report Server, you should know that it creates the Report Server Virtual Directory as ReportServer\$SQLEXPRESS and it creates the Report Manager Virtual Directory as Reports\$SQLEXPRESS. This is important because when you are deploying your report using the Report Designer, you must tell the designer to use ReportServer\$SQLEXPRESS, not ReportServer, which is the default. This property is called the *TargetServerURL*, and you can change it by going to the Properties dialog box reached via the Project menu.

An in-depth discussion of Reporting Services is beyond the scope of this book, but a plethora of resources are available online, including free Report Packs to help you get started writing reports. Report Packs can be found at <http://www.microsoft.com/downloads/details.aspx?FamilyId=D81722CE-408C-4FB6-A429-2A7ECD62F674&displaylang=en>.

MSDN provides tutorials on Reporting Services as well as Webcasts and many other types of information. Go to this URL: <http://msdn.microsoft.com/sql/bi/reporting/>.

## Full Text in Express Edition

The full-text feature within SQL Server Express Edition is no different than in any other edition of SQL Server. The main issue with full text in Express Edition is that the SQL Server

Management Studio Express tool does not have any fancy user interface to manage the full-text catalogs or anything related to full text. Thus, to use full text within Express Edition, you must brush up on your T-SQL skills. Luckily, Template Explorer within Management Studio Express helps you by providing templates for creating a full-text catalog, creating a full-text index, populating an index, and stopping population of an index.

To start playing with full text in Express Edition, we'll walk through creating a full-text index on the `ProductName` of the `Products` table in the `Northwind` database. First we must create our full-text catalog as shown here:

```
USE Northwind
GO
CREATE FULLTEXT CATALOG MyNorthwindCatalog AS DEFAULT
```

In SQL Server 2005, all user-defined databases are enabled for full text by default, so there is no need to explicitly call `sp_fulltext_database 'enable'`, as you do in previous versions of SQL Server. Next we will create the actual index:

```
CREATE FULLTEXT INDEX ON dbo.Products(ProductName) KEY INDEX PK_Products
```

Now we can use this index and query the `Products` table. Because I love gumbo, I can now quickly return all the gumbo products using the `CONTAINS` keyword, as shown here:

```
SELECT ProductName from Products where CONTAINS(ProductName, 'Gumbo')
```

The result is as follows:

```
ProductName
-----
Chef Anton's Gumbo Mix
```

In SQL Server 2005, only columns of type `char`, `varchar`, `nchar`, `nvarchar`, `text`, `ntext`, `image`, `xml`, and `varbinary` can be indexed for full-text search. The previous example shows how to quickly get an index up and running. However, as with most multi-generational features in SQL Server, the capabilities of full text go far beyond what was discussed here. If you are interested in learning more about full-text searching within SQL Server 2005, go to [http://msdn2.microsoft.com/en-us/library/ms142547\(SQL.90\).aspx](http://msdn2.microsoft.com/en-us/library/ms142547(SQL.90).aspx).

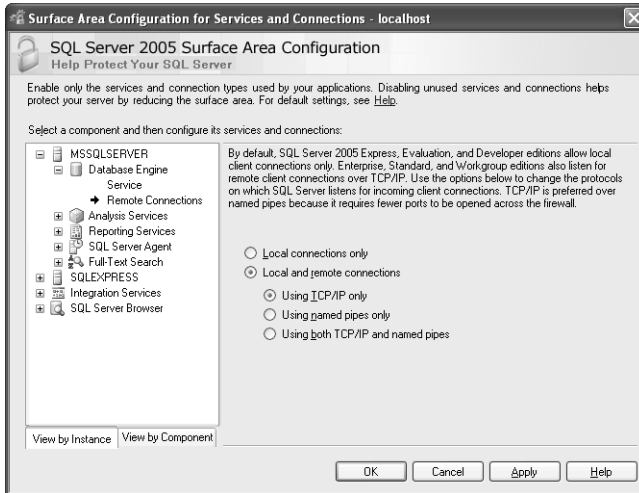
## Configuration

Every new database version seems to have even more features, more knobs to turn, and other things that users must learn about. SQL Server 2005 reflects a lot of effort put into improving the security of the product. One aspect of this effort was an “off by default” initiative, which basically required all nonessential features of SQL Server to be turned off. For example, on a default installation of SQL Server, the Browser service and Full-Text service are not running. Turning these features off reduces the surface area available for a malicious user.

The SQL Server Surface Area Configuration tool allows you to turn on and off these various features. This tool is installed as part of a database engine installation, so you have it regardless of which edition or components you install.

When the Surface Area Configuration tool is launched, you get two options: Surface Area Configuration for Services and Connections, and Surface Area Configuration for Features.

The Services and Connections dialog box, shown in Figure 14-4, shows a tree view list of SQL Server components installed on the server.



**Figure 14-4** The Surface Area Configuration tool's remote connections panel

Under Database Engine you'll find two panels: Services allows the user to start, stop, and change the automatic startup settings of the service, and Remote Connections is perhaps one of the more important switches to remember. By default, in the Express, Evaluation, and Developer editions of SQL Server, remote connections are disabled. Thus, if you installed Express Edition and try to connect to it from another client machine, the connection will fail. To enable remote connections, you must choose which kind of remote connections to allow.



**Note** If you allow remote connections and are still having trouble connecting, check your firewall settings. You might have to add an exception for the `sqlsvr.exe` process or open a specific port for SQL Server.

The other important feature of this tool is the ability to configure which features should be on or off. Clicking on Surface Area Configuration for Features launches the dialog box shown in Figure 14-5.

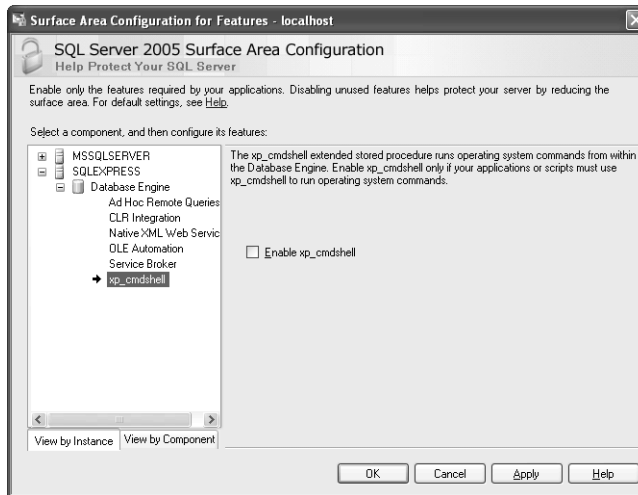


Figure 14-5 The Surface Area Configuration tool's features panel

Express Edition has a reduced set of features to enable or disable compared to other versions of SQL Server. This is because Express Edition doesn't have features such as SQL Mail, Database Mail, and SQL Server Agent. In Figure 14-5, you can see that the Surface Area Configuration tool allows you to enable and disable the use of the *xp\_cmdshell* extended stored procedure. Other features that you can turn on and off include the CLR, ad hoc remote queries, and Service Broker.

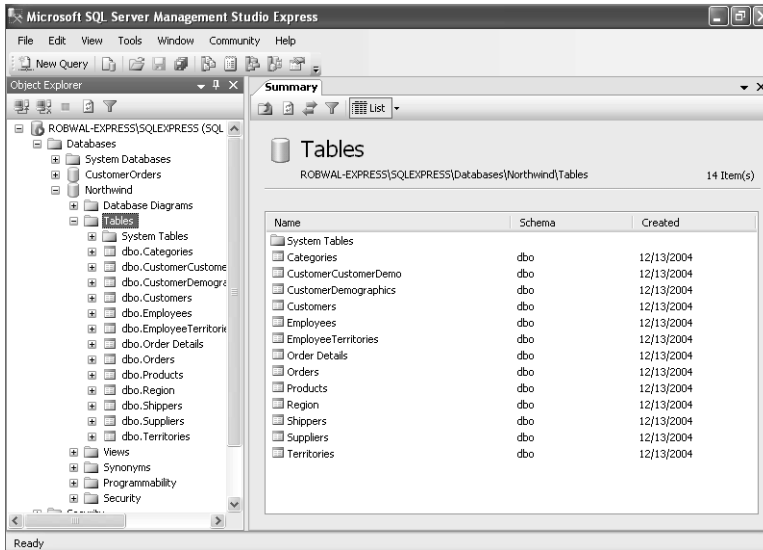
You can programmatically turn on and off each of these options via the *sp\_configure* stored procedure. To view these options, set the Show Advanced property on *sp\_configure* first. There is also a new system view called *sys.configurations* that effectively shows the same information as *sp\_configure* with the Show Advanced option set.

## Working with SQL Server Express Edition

SQL Server Management Studio Express is a graphical user interface tool for managing SQL Server. It is a stripped-down version of its parent SQL Server Management Studio. The full version, which ships with all other editions of SQL Server, includes support for managing Analysis Services and Integration Services and offers much more functionality. This isn't to say that Management Studio Express is not useful; on the contrary, it supports all the basic administration functions, such as backup and restore, and it provides a rich text editor for creating and executing queries.

You can launch Management Studio Express from the Start menu. You first see a Connection dialog box. The default *SQLEXPRESS* instance name is automatically provided for you, so to connect to your Express Edition instance on your local machine, simply click the Connect button.

The GUI has three main parts: the toolbar, the Object Explorer tree view, and the document window, as shown in Figure 14-6. Visual Studio users will be most familiar with how the GUI works—in fact, SQL Server Management Studio is technically a Visual Studio application.



**Figure 14-6** The SQL Server Management Studio Express tool

Figure 14-6 shows Management Studio Express connected to a server. The Object Explorer tree enumerates the Tables node of the Northwind database. One issue to note is that multi-select is not supported in Object Explorer in either Management Studio or Management Studio Express. To perform an action such as deleting multiple objects, you must use the Summary panel. By default, this panel shows information that depends on which object is selected in Object Explorer. You can close this panel without closing Management Studio Express. If this panel is not visible, you can always enable it by choosing Summary from the View menu or by pressing F7.

On the subject of multi-select, you can hold down the Ctrl key and select multiple objects in the grid and then right-click on the grid to see a shortcut menu of available options. This is the same behavior you would expect from any other Windows application. Another thing you will notice with shortcut menus and multi-select is that a lot more options are available when you select only one object. Because Object Explorer only supports selecting one item at a time, you get the same shortcut menu for both Object Explorer and the Summary grid if you have only one object selected.

You can use Management Studio Express to manage other editions of SQL Server, but the functionality is limited to what is available for Express Edition. (For example, when you connect to Workgroup Edition or Enterprise Edition, there is no way to manage Agent jobs.)

At this point, we are ready to discuss some of the more useful features within Management Studio Express. For this example, we will create a new database called CustomerOrders. Connect to your Express Edition server and select New Database from the shortcut menu of the Database Node in Object Explorer. This brings up the New Database dialog box (Figure 14-7).

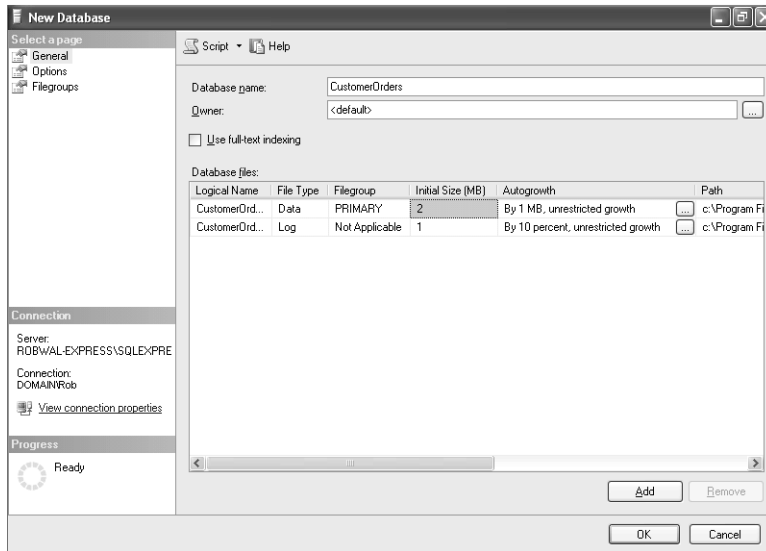
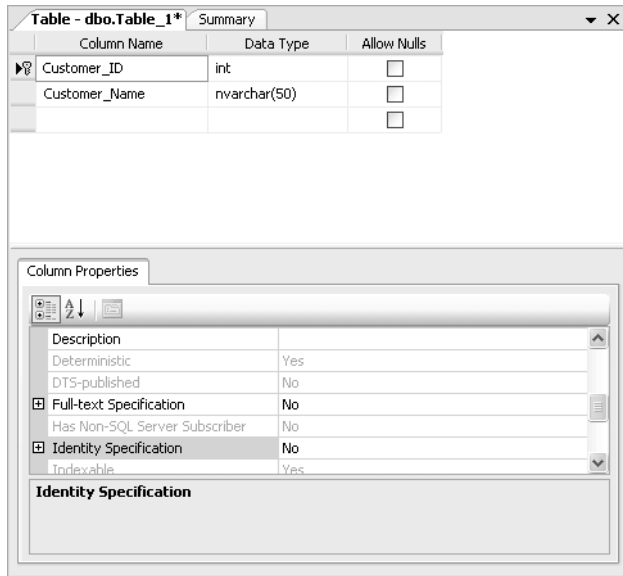


Figure 14-7 The New Database dialog box in Management Studio Express

Using this dialog box, we can specify the most common parameters, such as data and log file location, autogrowth parameters, and other database attributes. Almost all dialog boxes in Management Studio Express (including those in the full version of Management Studio) allow you to script the action instead of actually executing the action against the server. This is useful in a variety of ways—for example, if we want to know exactly what will be executed against our server without actually executing any commands. All you do is click the Script button in the dialog box and choose where to dump the script.

Once we have specified the name of the database—in our case, CustomerOrders—we can click OK, and we will see the new CustomerOrders node in Object Explorer. Now it's time to create a table for our new database. We could open a new query window and type the *CREATE TABLE* syntax with all the interesting columns we want, but there is a much simpler way to use Management Studio Express.

Management Studio Express provides a table designer for creating and modifying tables, as well as a view designer. To launch the table designer, right-click on the Tables node under our new database and choose New Table. This launches the table designer in another document window, as shown in Figure 14-8. This modeless approach allows users to go back and forth between document windows without having to close down each one first. Every dialog box used in Management Studio Express is modeless, so you do not have to launch another instance of Management Studio Express to do other work while another task (such as restoring a database) is taking a long time.



**Figure 14-8** The table designer in Management Studio Express

The table designer has two main sections, the column definition grid and the column properties panel at the bottom of the window. The grid allows you to simply type the name of the column, select a data type from the combo box, and specify whether the column will allow null values. You can set the primary key, create an index, and create check constraints—among other things—by right-clicking a row in the grid. The power of the table designer and its ease of use make it a valuable asset in Management Studio Express.

The View designer in Management Studio Express provides a feature-rich environment for creating and modifying views. This designer has four panes: Diagram, Criteria, SQL, and Results. You can enable all panes at once (which results in a busy user interface, but if you have a big enough monitor this might be OK). For a clearer demonstration, Figure 14-9 shows the designer with the Diagram and Criteria panes, and Figure 14-10 shows the SQL and Results panes. These figures show data from the Northwind database because it offers a more complex relationship than the single table we have built so far in this chapter.

The Diagram pane shows an entity relationship among the various tables within the view.



**Note** To diagram the database, you can use a separate Diagram node in Object Explorer, which produces a user interface that is similar to the designers previously described.

The boxes indicate a table; you can resize and move them by clicking and dragging with the mouse. This view is not read-only; it's interactive—you can modify the contents of the tables and add or remove tables. For example, if you selected the Description check box in the Categories table, this would add Description to the criteria pane and you can add this to your view.

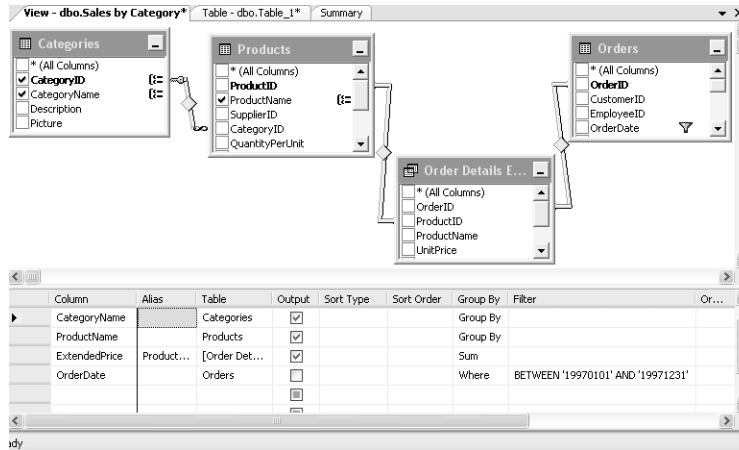


Figure 14-9 The View designer with the Diagram and Criteria panes

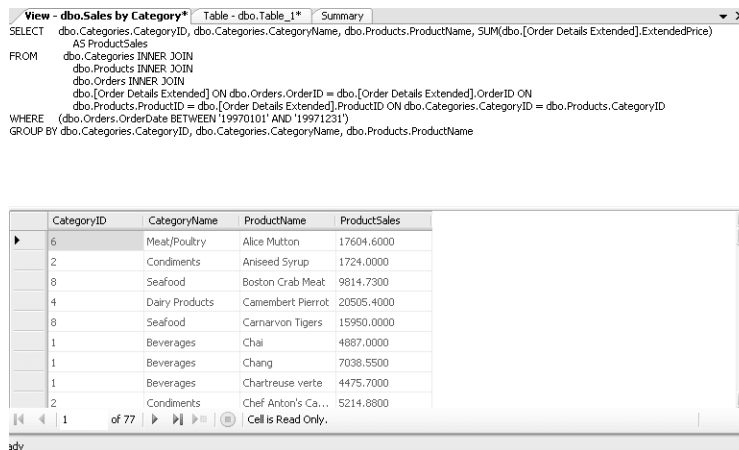


Figure 14-10 The View designer with the SQL and Results panes

If you want to tweak the T-SQL in the view, you can do so through the SQL pane. Any changes made in the SQL pane are carried over to all the other panes, and vice-versa. So if you checked the Description check box in the Categories table, this information is automatically added to the Criteria pane and the T-SQL within the SQL pane.

To conclude our discussion of Management Studio Express, we'll look at the ability to create and manage ad hoc queries. To create a new query, you simply click the New Query button. This creates a new document window and allows you to free-type T-SQL or load or save a script file. The query editor (as it is called) does not have IntelliSense (which offers autocompletion based on the first few characters of an object name that are typed), but it does have some other helpful features, such as line numbering and displaying the execution plan. Figure 14-11 shows the execution plan of the Invoices view in the Northwind database.

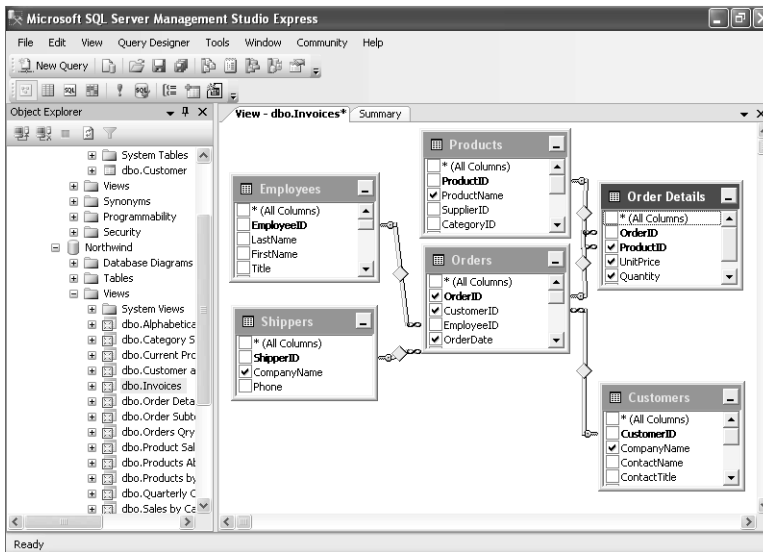


Figure 14-11 Invoices view

Notice the three tabs in the bottom section of the document window: Results, Messages, and Execution Plan. The Results tab normally shows up by default, displaying the results of whatever we submitted to SQL Server. In this example, to get the Execution Plan tab to show up, we had to first choose Include Actual Execution Plan from the Query menu.

Management Studio Express goes far beyond any management tool provided with MSDE—which is, basically, nothing. Most important, its tools are free and can be freely distributed with any custom application as long as the SQL Server Express Edition engine is shipped in the same application.

## SQLCMD Command-Line Tool

Traditionally, command-line tools have always been available for executing script or ad hoc queries against SQL Server. These have included `iSQL` and `oSQL`, which each use a different technology, such as Open Database Connectivity (ODBC), to connect to the server instance and have different features. In SQL Server 2005, a new command-line tool replaces both `oSQL` and `iSQL`. `SQLCMD` is not just a replacement—it extends the functionality of the current tools in a variety of ways. Some of the new features include the following:

- Scripting variables
- Ability to include multiple in-line scripts
- Ability to dynamically change connections
- Support for Dedicated Administrator Connection (DAC)
- Support for the new SQL Server 2005 data types: `xml` and `nvarchar(max)`

Suppose you need to create a maintenance script that performs a DBCC *CHECKDB*, performs a database backup, and then copies the backup file to another location. Here is an example T-SQL script:

```
DBCC CHECKDB ('Northwind') WITH NO_INFOMSGS
GO
BACKUP DATABASE [Northwind] TO DISK='C:\Backups\Northwind\Northwind.bak'
GO
Exec xp_cmdshell 'copy C:\Backups\Northwind\*.bak D:\TapeBackupDropFolder\Northwind'
GO
```

Life with this single script file goes well, and eventually you are tasked with doing this for another database. Fine—you copy and paste the code and change the Northwind entries to pubs entries, and now you have the script shown in Listing 14-1.

#### Listing 14-1 T-SQL script

```
DBCC CHECKDB ('Northwind') WITH NO_INFOMSGS
GO
BACKUP DATABASE [Northwind] TO DISK='C:\Backups\Northwind\Northwind.bak'
GO
Exec xp_cmdshell 'copy C:\Backups\Northwind\*.bak D:\TapeBackupDropFolder\Northwind'
GO
DBCC CHECKDB ('Pubs') WITH NO_INFOMSGS
GO
BACKUP DATABASE [Pubs] TO DISK='C:\Backups\Pubs\Pubs.bak'
GO
Exec xp_cmdshell 'copy C:\Backups\Pubs\*.bak D:\TapeBackupDropFolder\Pubs'
GO
```

Time passes, and you are asked to do this for another three databases, and the location of the drop folder has changed. You can imagine that every time you want to add databases to this script, another copy and paste will be required, which introduces the possibility of bugs within the script. Wouldn't it be great if we could use variables within the script instead? SQL-CMD in SQL Server 2005 supports the use of variables for this type of scenario. To create a variable, we use *:SETVAR*, as shown in the modified script in Listing 14-2.

#### Listing 14-2 T-SQL script (modified)

```
:SETVAR DatabaseName Northwind
DBCC CHECKDB ($(DatabaseName)) WITH NO_INFOMSGS
GO
BACKUP DATABASE $(DatabaseName) TO DISK='C:\Backups\$(DatabaseName)\$(DatabaseName).bak'
GO
Exec xp_cmdshell 'copy C:\Backups\$(DatabaseName)\*.bak D:\TapeBackupDropFolder\
$(DatabaseName)'
GO

:SETVAR DatabaseName Pubs
```

```

DBCC CHECKDB ($(DatabaseName)) WITH NO_INFOMSGS
GO
BACKUP DATABASE $(DatabaseName) TO DISK='C:\Backups\$(DatabaseName)\$(DatabaseName).bak'
GO
Exec xp_cmdshell 'copy C:\Backups\$(DatabaseName)\*.bak
D:\TapeBackupDropFolder\$(DatabaseName)'
GO

```

Before we optimize this example, we should mention a few key points. You can create and change variables throughout the script. You can also place the value of the variable by typing the name of the variable preceded by `$(` and followed by `)`. The replacement happens at run time for the script, and as you can see in this example, it dynamically places the backup in the folder whose name is defined in the variable. In addition, a list of environment variables are available to use within your script. They are listed in Table 14-3.

**Table 14-3 SQLCMD Environment Variables**

Variable	Related Switch	Read/Write
<i>SQLCMDUSER</i>	<i>U</i>	R
<i>SQLCMDPASSWORD</i>	<i>P</i>	N/A
<i>SQLCMDSERVER</i>	<i>S</i>	R
<i>SQLCMDWORKSTATION</i>	<i>H</i>	R
<i>SQLCMDDBNAM</i>	<i>d</i>	R
<i>SQLCMDLOGINTIMEOUT</i>	<i>l</i>	R/W
<i>SQLCMDSTATTIMEOUT</i>	<i>t</i>	R/W
<i>SQLCMDHEADERS</i>	<i>h</i>	R/W
<i>SQLCMDCOLSEP</i>	<i>s</i>	R/W
<i>SQLCMDCOLWIDTH</i>	<i>w</i>	R/W
<i>SQLCMDPACKETSIZE</i>	<i>a</i>	R
<i>SQLCMDERRORLEVEL</i>	<i>m</i>	R/W
<i>SQLCMDMAXVARTYPEWIDTH</i>	<i>y</i>	R/W
<i>SQLCMDMAXFIXEDTYPEWIDTH</i>	<i>Y</i>	R/W

Using these special variables is the same as using a custom variable. For example, if we want to print out the name of the workstation executing the script, we can do the following:

```
Print "$(SQLCMDWORKSTATION)"
```

Returning to our example, if we simply left the script as is, our managers and colleagues might question our programming skills. To make this script more efficient, let's just write the process once and define the variables on the command line. So our script, which we will call `BackupMyDatabase.sql`, becomes that shown in Listing 14-3.

**Listing 14-3** BackupMyDatabase.sql

```
DBCC CHECKDB ($(DatabaseName)) WITH NO_INFOMSGS
GO
BACKUP DATABASE $(DatabaseName) TO DISK='C:\Backups\$(DatabaseName)\$(DatabaseName).bak'
GO
Exec xp_cmdshell 'copy C:\Backups\$(DatabaseName)\*.bak D:\TapeBackupDropFolder\
$(DatabaseName)'
GO
```

To back up each database, we can use SQLCMD as follows:

```
SQLCMD -E -S.\SQLEXPRESS -i "C:\MaintenanceScripts\BackupMyDatabase.sql" -
v DatabaseName="Northwind"
```

Notice that the parameters are case sensitive. A *-v* tells SQLCMD that the subsequent value is a variable name, not a severity level (as in the case of a capital *V*). In this example, we tell SQLCMD to connect to the *SQLEXPRESS* instance, execute the *.sql* script file *BackupMyDatabase.sql*, and define the *DatabaseName* variable to be *Northwind*. Now all we have to do is repeat this for *pubs*, and we are all set. However, we can leverage yet another feature to make this even more efficient. We can include a script within an existing script and still keep our variable values. Expanding on this example, let's create a new script called *LaunchBackup.sql*, as defined in Listing 14-4.

**Listing 14-4** LaunchBackup.sql

```
:SETVAR DatabaseName Northwind
:R "BackupMyDatabase.sql"
:SETVAR DatabaseName Pubs
:R "BackupMyDatabase.sql"
```

The *:R* tells SQLCMD to load the *.sql* script in-line at run time, before execution. Now it's quite easy to add another database to our process of backing up the database.



**Note** To run this script, all we have to do is tell SQLCMD to launch the script file. We don't have to pass variables because they are defined in the script file itself.

```
SQLCMD -E -S.\SQLEXPRESS -i "LaunchBackup.sql"
```

What if we want to perform this operation on multiple servers or multiple instances of SQL Server? SQLCMD has yet another feature that allows the script to make connections from within the script using the *:CONNECT* command. Expanding on our example, say we want to connect to our test server and perform this operation. We can do this with the code in Listing 14-5.

**Listing 14-5** LaunchBackup.sql connecting to multiple servers

```
--Connect to my local express machine
:CONNECT .\SQLEXPRESS
:SETVAR DatabaseName Northwind
:R "BackupMyDatabase.sql"
--Now connect to my test server and backup Northwind
:CONNECT TESTSERVER1\SQLEXPRESS
:SETVAR DatabaseName Northwind
:R "BackupMyDatabase.sql"
```

Combining the powerful functionality of connecting to different servers and variables gives us the ability to do a plethora of tasks within our database scripts. One additional key feature of SQLCMD is worth mentioning here. In the SQL Server 2005 engine there is a special thread that basically waits just for a connection from an SQLCMD client with a `-A` parameter. This special connection is called the Dedicated Administrator Connection (DAC), and its purpose is to provide administrators with a way to connect to SQL Server if, for whatever reason, the server becomes unresponsive. The DAC is available only when you connect to SQL Server via SQLCMD with the special `-A` parameter. You cannot make this connection from any other client software (such as SQL Server Management Studio). In addition, only one connection of this type is allowed per instance of SQL Server because this feature is designed to be used only in the event that administrators cannot gain access through a normal network connection.

There is one more important point about the DAC: In Express Edition, the DAC is disabled. To use the DAC, you must start the service running a trace flag of 7806. If you have trouble connecting to the DAC once you restart the service with this trace flag, make sure the SQL Server Browser service is running.



**Note** To set a trace flag in SQL Server, use `-T#` as a startup parameter, where `#` is the trace flag number. If you view the SQL Server service properties from the Service control panel, you will see a Start Parameters text box. To experiment with the DAC, you can place `-T7806` (no space between the T and 7) there.

SQLCMD can connect to the Express Edition-only feature called User Instances. However, there is another tool called SSEUTIL (available for download on MSDN) that looks and feels like SQLCMD but is used primarily to manage Express Edition user instances. Before we go into the details of SSEUTIL and how it compares with SQLCMD, we should first provide an overview of User Instances.

## User Instances

SQL Server 2005 Express Edition supports a new feature called User Instances that is available only when you use the .NET Framework data provider for SQL Server (SqlClient). This feature is available only with Express Edition. A user instance is basically a separate

instance of the Express Edition database engine that is generated by a parent instance. For example, as a Windows administrator, I can install Express Edition on my client machine. I can then take an application that will connect to Express Edition and create a separate instance of Express Edition for each client user of the application. When connected to this user instance, the client is the sysadmin for that instance but does not have any special privileges on the parent Express Edition instance. This is important because software executing on a user instance with limited permissions cannot make system-wide changes. This is because the instance of Express Edition is running under the non-administrator Windows account of the user, not as a service. Each user instance is isolated from its parent instance and from any other user instances running on the same computer. Databases running on a user instance are opened in single-user mode only, and multiple users cannot connect to databases running on a user instance. Replication, distributed queries, and remote connections are also disabled for user instances.

When you install Express Edition, one of the setup wizard pages asks if you want to enable User Instances. This is one way to enable them; the other is to toggle the setting using `sp_configure`, as follows:

```
-- Enable user instances.
sp_configure 'user instances enabled', '1'
-- Disable user instances.
sp_configure 'user instances enabled', '0'
```

There are a couple more restrictions on user instances. For one thing, the only way to connect to them is through Named Pipes. In addition, SQL Server logins are not supported on User Instances; connections to the User Instance must be made using Windows Authentication.

Now that we have discussed User Instances and some of the restrictions on them, take a look at the example connection string shown here:

```
Data Source=.\SQLEXPRESS;Integrated Security=true;
User Instance=true;AttachDBFilename=|DataDirectory|\InstanceDB.mdf;
Initial Catalog=InstanceDB;
```

Note the following:

- The *Data Source* keyword refers to the parent instance of Express Edition that is generating the user instance. The default instance is `.\sqlexpress`.
- *Integrated Security* is set to `true`. To connect to a user instance, you must use Windows Authentication; SQL Server logins are not supported.
- *User Instance* is set to `true`, which invokes a user instance. (The default is `false`.)
- *AttachDBFilename* indicates the location of the InstanceDB database.

- The *DataDirectory* substitution string enclosed in the pipe symbols refers to the data directory of the application opening the connection and provides a relative path indicating the location of the .mdf and .ldf database and log files. If you want to locate these files elsewhere, you must provide the full path to the files.

When the *SqlConnection* is opened, it is redirected from the default Express Edition instance to a run-time-initiated instance running under the caller's account.

Unlike with versions of SQL Server that run as a service, SQL Server Express Edition user instances do not need to be manually started and stopped. Each time a user logs in and connects to a user instance, the user instance is started if it is not already running. User instance databases have the *AutoClose* option set so that the database is automatically shut down after a period of inactivity. The *Sqlservr.exe* process that is started is kept running for a limited timeout period after the last connection to the instance is closed, so it does not need to be restarted if another connection is opened before the timeout has expired. The user instance automatically shuts down if no new connection opens before that timeout period has expired. A system administrator on the parent instance can set the duration of the timeout period for a user instance by using *sp\_configure* to change the user instance timeout option. The default is 60 minutes.

The first time a user instance is generated for each user, the master and msdb system databases are copied from the Template Data folder to a path under the user's local application data repository directory for exclusive use by the user instance. This path is typically C:\Documents and Settings\\Local Settings\Application Data\Microsoft\Microsoft SQL Server Data\SQLEXPRESS. When a user instance starts up, the tempdb, log, and trace files are also written to this directory. A name is generated for the instance, which is guaranteed to be unique for each user.

By default, all members of the Windows Builtin\Users group are granted permissions to connect on the local instance as well as read and execute permissions on the SQL Server binaries. Once the credentials of the calling user hosting the user instance have been verified, that user becomes the sysadmin on that instance. Only shared memory is enabled for user instances, which means that only operations on the local machine are possible.

Note that users must be granted both read and write permissions on the .mdf and .ldf files specified in the connection string. These files represent the database and log files, respectively, and are a matched set. The main database file will not open if it is coupled with the wrong log file.

To prevent data corruption, a database in the user instance is opened with exclusive access. If two different user instances share the same database on the same computer, the user on the first instance must close the database before it can be opened in a second instance. User instances are given a globally unique ID (GUID) as a name. This makes them a bit more challenging to manage. The SSEUTIL command-line tool makes managing user instances easier.

## SSEUTIL

SSEUTIL is available as a free download from MSDN at <http://www.microsoft.com/downloads/details.aspx?FamilyID=fa87e828-173f-472e-a85c-27ed01cf6b02&DisplayLang=en>. The tool lets you easily interact with SQL Server. Among other things, it allows you to:

- Connect to the main instance or user instance of SQL Server
- Create, attach, detach, and list databases on the server
- Upgrade database files to match the version of the server
- Execute SQL statements via the console (similar to SQLCMD)
- Retrieve the version of SQL Server that is running
- Enable and disable trace flags (for example, to trace SQL statements sent to the server by any client application)
- List the instances of SQL Server on the local machine or on remote machines
- Checkpoint and shrink a database
- Measure the performance of executing specific queries
- Create and play back lists of SQL commands for the server to execute
- Log all input and output

Although SSEUTIL might appear to be very similar to SQLCMD, it uses different command-line switches to connect to SQL Server. Once you connect to SQL Server Express Edition using SSEUTIL, you can either use a command prompt (which resembles the SQLCMD behavior) or you can request a graphical console window. Either way, you must specify either `-c` for a command prompt or `-consolewnd` to use the graphical user interface, as shown here:

```
c:\>sseutil -s .\sqlexpress -c
```

The default in Express Edition is to call the named instance, `SQLEXPRESS`. SSEUTIL does not require you to specify the `-s` (*servername*) parameter; if you omit it, SSEUTIL assumes that you want to connect to the `SQLEXPRESS` instance on your local box.

Another important point about SSEUTIL and defaults is worth noting. From the preceding information, you might think that we are connecting to the default parent instance of Express Edition. In reality, we are connecting to the server instance, `.\sqlexpress`, but our connection is then redirected to the user instance of the user who is logged in. If we specifically want to connect to the Express Edition parent instance, we must pass the `-m` parameter, as shown here:

```
c:\>sseutil -s .\sqlexpress -m
```

If you do not specify `-m`, SSEUTIL uses the `-child` parameter, which tells SSEUTIL to connect to the user instance of the user who is currently logged in to the box. You can append a

different user name to this parameter if you want to connect to another user instance. To obtain a list of active user instances, you can use the *-childlist* parameter, as shown here:

```
C:\>sseutil -childlist
User                Pipe                ProcessId  Status
-----
DOMAIN\robwal      \\.\pipe\07EB27D8-877E-4F\tsql\query  2168      alive
```

Now that we have discussed how to connect to both the parent and child instances using SSEUTIL, it's time to do something useful with SSEUTIL. Suppose we want to attach the Northwind database to our user instance. We can enumerate the databases that are attached to our instance by using the *-l* parameter:

```
C:\>sseutil -child DOMAIN\robwal -l
Using instance '\\.\pipe\07EB27D8-877E-4F\tsql\query'.

1. master
2. tempdb
3. model
4. msdb
```

SSEUTIL lets you attach or detach a database from both the parent instance and any user instance on your box. Let's attach Northwind, whose database files we'll assume are located in a directory called *C:\databases*.

```
C:\>sseutil -child DOMAIN\robwal -a "C:\databases\northwnd.mdf" Northwind
Using instance '\\.\pipe\07EB27D8-877E-4F\tsql\query'.

Command completed successfully.
```

For completeness, we specified the *-child* parameter, but it does not actually need to be there because it's the default. The *-a* attach parameter takes two values. The first is the location of the *.mdf* file to be attached. The second parameter, which is optional, is the name of the database. Once attached, the database exists on the instance but not on the parent instance, again reinforcing our belief that user instances are truly a separate instance of SQL Server.

Here is a list of databases on the parent Express Edition instance:

```
C:\>sseutil -m -l master
2. tempdb
3. model
4. msdb
5. CustomerOrders
```

Here is a list of databases on the user instance of the logged-in user:

```
C:\>sseutil -child DOMAIN\robwal -l
Using instance '\\.\pipe\07EB27D8-877E-4F\tsql\query'.

1. master
2. tempdb
3. model
4. msdb
5. Northwind
```

If you develop an application that uses user instances, you will find that SSEUTIL is a useful tool for quickly obtaining information and performing tasks against the instance. Alternatively, you can use SQL Server Management Studio Express against a user instance. To do this, you simply obtain the pipename of the user instance. One way to get the pipename is to use the `-childlist` parameter. In the examples in this section, our pipename was `\\.\pipe\07EB27D8-877E-4F\tsql\query`. Open Management Studio Express and, in the Connection dialog box, paste the pipename into the Server Name text box.

## Installing SQL Server Express Edition

Express Edition offers three ways to perform an installation or upgrade: via the Setup Wizard, silently through the use of command-line parameters, or via a configuration file. Using the wizard to quickly get your development or test environment set up is useful if you don't already have Express Edition on your box. When bundling Express Edition with your application, you might find that installing Express Edition via a configuration file provides the best user experience.



**Note** You might already have Express Edition installed if you already have Visual Studio installed on your box. You can check to see whether Express Edition is installed by going to the Services applet in Control Panel and looking for "SQL Server (SQLEXPRESS)." Note that if you installed Express Edition and gave it a different instance name from the default SQLEXPRESS, the name shown in Control Panel might be different.

You can also run the Express Edition setup to install another instance of Express Edition. Express Edition supports up to 50 instances on a single computer.

### Existing Beta Versions of SQL Server 2005

If your box contains previously released versions of SQL Server, you might run into problems if you attempt to upgrade to the released version by simply executing the setup program of the released version. To ensure a successful installation, manually remove all pre-release versions of SQL 2005, Visual Studio 2005, and the .NET

Framework 2.0. If you are lucky enough to be in this situation, it will be beneficial to read through the MSDN article “Uninstalling Previous Versions of Visual Studio 2005,” which can be found at <http://msdn.microsoft.com/vstudio/express/support/uninstall>.

If you want to download SQL Server 2005 Express Edition from the Microsoft Web site, you can go to <http://go.microsoft.com/fwlink/?LinkId=64064>. On this Web page, you will see the prerequisites for installing Express Edition. You need Windows Installer 3.0 and the .NET Framework 2.0. The Web page includes links for downloading these—you must install them before you can run setup for Express Edition, which is the file on this Web page called SQLEXPRESS.exe. If you simply run this executable, it self-extracts and runs through the Setup Wizard. To perform a silent installation or an installation using the configuration file, you can download this file and run SQLEXPRESS.exe /X with the /X attribute; this extracts the files without running the Setup Wizard. If you do not want to extract all the Express Edition installation files, you can also pass installation parameters to SQLEXPRESS.exe.

After we have the Express Edition setup bits on our box, we can either run through the wizard to install or upgrade to SQL Server 2005 Express Edition or we can use a configuration file called Template.ini to silently perform an installation. We will first cover installing Express Edition using the Setup Wizard.

## Using the Setup Wizard to Manually Install Express Edition

Once you have the .NET Framework 2.0 and at least version 3.0 of the Windows Installer, you are ready to run the Setup Wizard for Express Edition. In this example, we'll use the setup for SQL Server Express Edition with Advanced Services (SQLEXPRESS\_ADV.exe).

Setup first installs the SQL Native Client and some additional setup support files. The SQL Native Client is a standalone data access API that is used for both the Object Linking and Embedding Database (OLE DB) and ODBC. It combines the SQL OLE DB provider and the SQL ODBC driver into one native dynamic link library (DLL). It also provides new functionality beyond that supplied by the Microsoft Data Access Components (MDAC).

Next, the familiar Welcome screen appears. Continuing through the wizard brings us to the System Configuration Check page. This page does some rudimentary checks to see whether your operating system is appropriate for Express Edition and whether you have any pending reboot requests from another application, and it looks at a number of other issues that could prevent setup from working correctly.

The next page in the wizard is for registration (Figure 14-12).

Notice the Hide Advanced Configuration Options check box at the bottom of the page. This is selected by default; if we uncheck it, we can change the name of the instance and database

collation, as well as specify some of the service settings for the SQL Server and SQL Server Browser service.

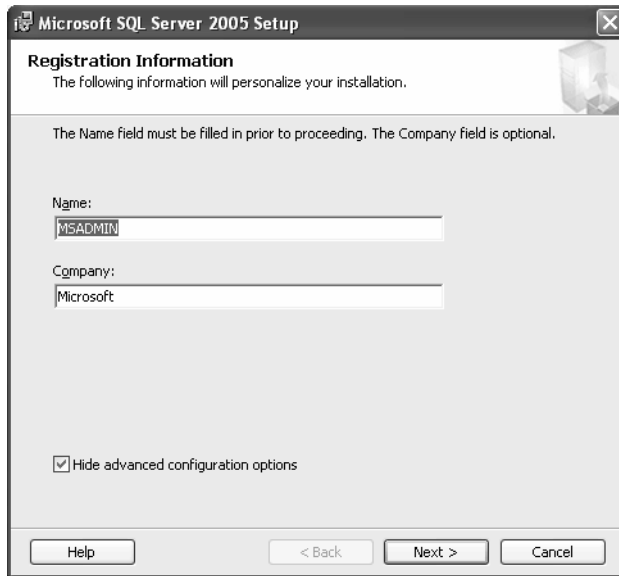


Figure 14-12 Registration Information page of the Setup Wizard

If we uncheck the box and continue with the wizard, we are presented with the Feature Selection page. This page is not an advanced option; users see it whether or not they chose to hide the advanced options. Figure 14-13 shows this page.

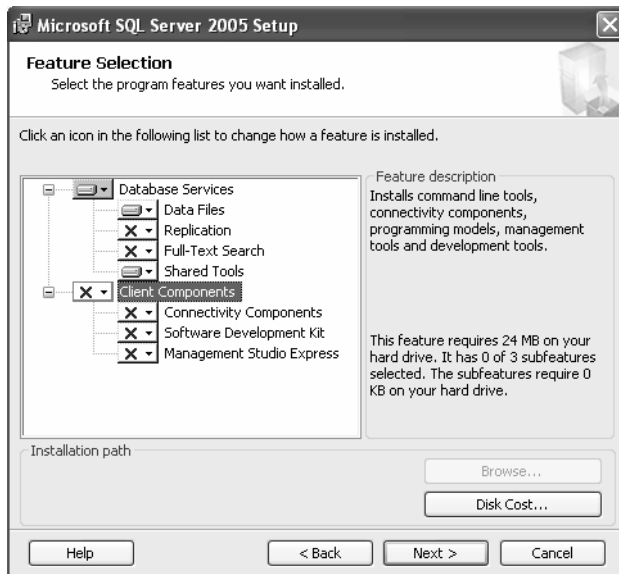


Figure 14-13 Feature Selection page of the Setup Wizard

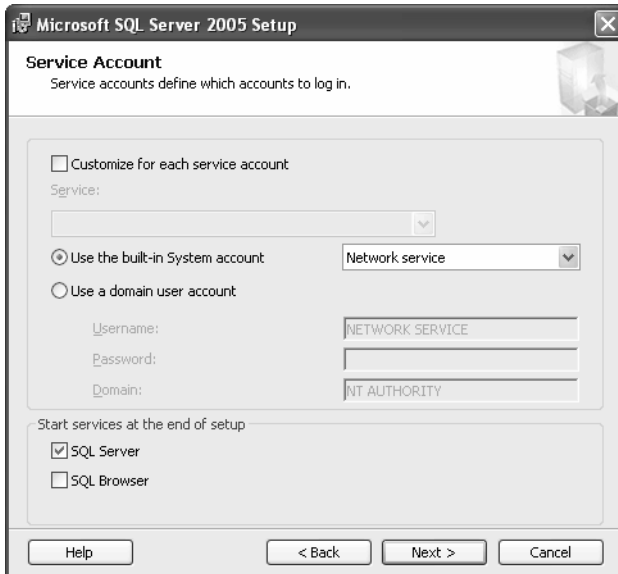


**Note** If you are running the Advanced Services setup and have IIS installed, you will see an option for installing Reporting Services. Choosing to install Reporting Services leads to a few other dialog boxes not listed here. If you choose to install Reporting Services, the wizard will ask if you want to configure Report Server now or later. If you select later, you can always configure it through the Report configuration tool, which is one of the tools provided under the Microsoft SQL Server node on the Start Menu.

Note that by default, only the database engine and its support files (such as the master database) are installed. If you want Replication, Full-Text, or Management Tools, you must explicitly add them using the drop-downs in the tree view.

If we had requested to see the advanced options, we would see the Instance Name page next. This page has two option buttons, one for installing Express Edition as the default instance and one for installing it as a named instance. The default is to install Express Edition as a named instance with the name SQLEXPRESS. This is how all Visual Studio installations of Express Edition create their installations of Express Edition, and it is the recommended installation.

The next advanced page is the Service Account page (Figure 14-14).



**Figure 14-14** Service Account page of the Setup Wizard

The first option you see is the ability to specify credentials for each service account. This is applicable if you are planning to use both the SQL Server service and the SQL Browser service. Most of the time, you can run both services using the Network service account.

Most developers can probably intuit what the “SQL Server” service is. But some of us might not be so sure about the SQL Browser service. In previous versions of SQL Server that supported the idea of “instances,” there needed to be a way for client machines to connect to SQL and ask, “What are all your available instances and their corresponding port numbers?” Traditionally, UDP port 1434 was used for this. As you can imagine, having this inside the server process was not only taxing but there was no way to effectively turn it off. In SQL Server 2005, this functionality has been separated out into a new service called the SQL Browser service.

The next page is the Authentication Mode page. Here you can set Express Edition to support only Windows Authentication or both SQL Authentication and Windows Authentication. If you choose Windows Authentication, you will still have an “sa” account, but that account will be disabled and have a random complex password.

Continuing on with the wizard, if you elected to show advanced pages, you will see the Collation Settings page. This page allows you to specify whether databases should be case sensitive or case insensitive, as well as what kind of sort order the database should use.

The next page asks you if you want to enable User Instances. User Instances is an Express Edition–specific feature—it is not available on any other edition of SQL Server. (User Instances were covered in the earlier section titled “Working with SQL Server Express Edition.”)

At this point, we are almost done with the wizard. The last page that requires us to make some sort of decision is the Error And Usage Report Settings page. This page asks two questions: “Can SQL automatically send error reports to either a central server in your organization or directly to Microsoft?” and “Can SQL send data on feature usage to Microsoft?” Both of these options enable Microsoft to compile a lot of information about the use of SQL Server and common errors. With the second option, no personally identifiable information is sent, so if you are in the mood to contribute to future products, you might want to select this box.

The next page shows a summary of the actions the wizard will perform. After this page, the wizard proceeds and installs Express Edition, including whatever options you have selected. The final page (Figure 14-15) displays some information about the installation, as well as two hyperlinks.

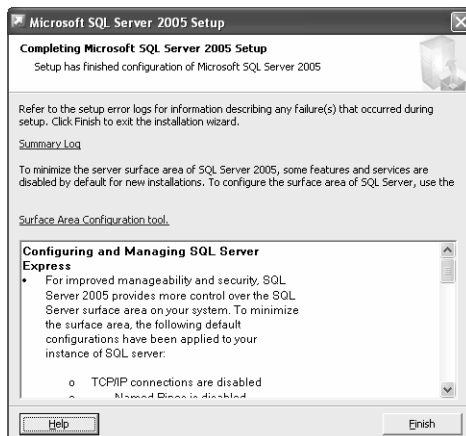


Figure 14-15 The final page of the Setup Wizard

The Summary Log link loads the installation log into Notepad for your reading or troubleshooting enjoyment. The other link launches the Surface Area Configuration tool (described previously in this chapter). If you do not run this tool now, you can always launch it from the Start menu later.

At this point, you are ready to start working with Express Edition.

## Installing via Command-Line Parameters or a Configuration File

There are two ways to programmatically install, modify, and remove SQL Server Express Edition components. First, you can call Setup.exe and pass a series of parameters on the command line. Alternatively, you can configure all the parameters within a single file and pass that file as a command-line parameter to Setup.exe. This file is called Template.ini, and it is located in the root directory of SQL Server Express Edition. An example of launching Setup and passing the configuration file is as follows:

```
start /wait setup.exe /qb /settings c:\template.ini
```

For those not familiar with the Windows command line, *start* is an application that opens a new console window. The */wait* parameter tells the console window to wait until the program finishes execution before terminating itself. A complete list of parameters for the *start* command can be obtained by passing */?* as a parameter. The next parameter in the line of code is *setup.exe*. This is the name of the application to launch; in this case, it's SQL Server Express Edition Setup. The rest of the parameters are arguments for the setup program. The Setup.exe argument */qb* tells Setup to run in quiet mode, which requires no user interaction. This mode does provide some visual indicators about the status of the installation.



**Note** Alternatively, you can specify */qn*. The only difference between */qb* and */qn* is that when you use */qn*, there is no visual status indicator.

All errors from Setup in SQL Server are recorded in the setup log files. If you call Microsoft for support on a setup-related issue, the Product Support specialist will probably want you to find these files. By default, the setup log files are located at C:\Program Files\Microsoft SQL Server\90\Setup Bootstrap\LOG\Files. If you encounter problems when developing a custom SQL Server Express Edition installation, these files are a good place to start debugging.

The */settings* parameter in the command-line code tells Setup to obtain all installation information from the file that is defined in the next parameter. In the code example, the Template.ini file is stored in the root of the C drive.



**Important** If you have downloaded SQL Server Express Edition from the Web, you might not see the Setup.exe application if you downloaded a single SQLEXPRESS.exe file. If this is the case, you need to call Setup.exe from the command line to extract the Express Edition files from this compressed executable. To perform this extraction, run `SQLEXPRESS /X` from the command line. A dialog box appears, prompting you for a location to extract the files to. The Express Edition files are copied to the location that you specify. These files include Setup.exe and Template.ini, among many other files and folders.

The Template.ini file is a plain-text file that can be opened by using a text editor such as Notepad. When you open this file, you see a long commented introduction citing examples of how to use the file. The file itself is well documented, and a lot of the options are explained in great detail within the file itself. For that reason, this chapter will not cover all the options. Instead, here are just a few parameters of interest.

- **PIDKEY** This parameter is required for all SQL Server editions except Express Edition.
- **ADDLOCAL** This parameter specifies which components to install. If *ADDLOCAL* is not specified, Setup will fail. You can specify *ADDLOCAL=ALL*, which installs all components. There are some rules around this parameter:
  1. Feature names are case sensitive.
  2. To use *ADDLOCAL*, you must provide a comma-delimited list of features to install, with no spaces between them.
  3. Selecting a parent feature installs only the parent feature, not the child features. Installing a child feature installs the parent and the child. Removing the parent feature removes both the parent and the child.
  4. You can also use *ADDLOCAL* to add components in maintenance mode.

Confused yet? We will give a few examples to demonstrate various installation combinations. First we must explain the possible valid feature names for each edition of SQL Server. Tables 14-4, 14-5, and 14-6 list the feature names unique to Express Edition, Express Edition with Advanced Services, and the Express Edition Toolkit, respectively.

**Table 14-4 Valid *ADDLOCAL* Parameters for Express Edition**

Feature	Parent Feature Name	Child Feature Name
SQL Server Database Services	SQL_Engine	
Data Files		SQL_Data_Files
Replication		SQL_Replication
Client Components	Client_Components	
Connectivity Components		Connectivity
Software Development Kit		SDK

**Table 14-5 Valid *ADDLOCAL* Parameters for Express Edition with Advanced Services**

Feature	Parent Feature Name	Child Feature Name
SQL Server Database Services	SQL_Engine	
Data Files		SQL_Data_Files
Replication		SQL_Replication
Full-Text Search Engine		SQL_FullText
Reporting Services	RS_Server	
Report Manager		RS_Web_Interface
Client Components	Client_Components	
Connectivity Components		Connectivity
Software Development Kit		SDK
Management Studio Express		SQL_SSMSE

**Table 14-6 Valid *ADDLOCAL* Parameters for Express Edition Toolkit**

Feature	Parent Feature Name	Child Feature Name
Client Components	Client_Components	
Connectivity Components		Connectivity
Software Development Kit		SDK
BI Development Studio		SQL_WarehouseDevWorkbench
Management Studio Express		SQL_SSMSE

Even though some of these feature names are self-descriptive, Table 14-7 provides a description of each feature, for completeness of our discussion.

**Table 14-7 Feature Descriptions**

Feature	Description
SQL_Engine	Installs the SQL Server database, including the SQL Server and SQL Browser services.
SQL_Data_Files	Installs core SQL Server databases, including master, the resource database, and tempdb.
SQL_Replication	Installs files necessary for replication support in SQL Server Express Edition.

Table 14-7 Feature Descriptions

Feature	Description
SQL_FullText	Installs files necessary for Full-Text Search support in SQL Server Express Edition.
RS_Server	Installs the Report Server service, which manages, executes, and renders reports.
RS_Web_Interface	Installs a Web-based tool used for managing a report server.
Client_Components	Installs components for communication between clients and servers, including network libraries for ODBC and OLE DB. Also installs applications such as the sqlcmd utility (oSQL replacement), SQL Server Configuration Manager, and the Surface Area Configuration tool.
Connectivity	Installs components for communication between clients and servers, including network libraries for ODBC and OLE DB.
SDK	Installs software development kits containing resources for model designers and programmers. This includes SQL Server Management Objects (SMO) and Replication Management Objects (RMO).
SQL_SSMSE	Installs SQL Server Management Studio Express.
SQL_WarehouseDevWorkbench	Installs BI Development Studio. It also installs Visual Studio Premier Partner Edition if no other edition of Visual Studio is installed on the computer.

With this many options, quite a number of installation combinations are possible. The following are examples of common installation configurations.

- **Install everything**

`ADDLOCAL=A11`

- **Install just the database engine**

`ADDLOCAL=SQL_Engine,SQL_Data_Files,Connectivity`

- **Install just the management tool**

`ADDLOCAL=SQL_SSMSE`

- **REMOVE** This parameter is similar to `ADDLOCAL`, but instead of adding components, it either removes a specific component or completely uninstalls SQL Server Express Edition if you use `REMOVE=ALL`. The following example removes the client components of an existing Express Edition installation:

```
REMOVE=Client_Components.
```

You do not have to specify an instance name because the *Client\_Components* are not instance-specific. If you were removing *SQL\_Replication* support, you would also need to add the following parameter:

```
INSTANCENAME=<<name of the SQL Server Express Instance>>
```

The `INSTANCENAME` parameter is needed whenever the feature is instance-specific.

- **UPGRADE** This parameter is used for upgrading from MSDE to SQL Server 2005 Express Edition. When you use *UPGRADE*, you must also specify the same instance name as the name of the MSDE instance you want to upgrade. This is because it is possible to have up to 16 MSDE instances on a single computer. Here is an example upgrade parameter:

```
UPGRADE=SQL_Engine INSTANCENAME=<<name of the MSDE instance>>
```

When you write custom installation applications, it can be difficult to remember these parameter names for all your projects. To make things easier, you can write a custom wrapper class to encapsulate setting the parameters and to provide a reusable stub for your custom applications. The wrapper class does not expose every option available, but it should give enough direction to suit your own custom installation needs.

## Deploying Express Edition Applications Using a Wrapper

As a custom application developer, you have three options for including SQL Server Express Edition within your application:

- Install Express Edition and then install the custom application.
- Install the custom application and then install Express Edition.
- Create a wrapper that combines the two-step process into a single step.



**Note** An SQL Server Express Edition wrapper cannot be MSI-based because Windows Installer does not support multiple instantiation of the Windows Installer service.

The remainder of this section focuses on creating a wrapper for your custom application. In the example code (Listing 14-6), the wrapper is a simple class that exposes three public methods: *IsExpressInstalled*, *EnumSQLInstances*, and *InstallExpress*. Ideally, you would not need to know if Express Edition or any other instance of SQL Server is already installed on the local computer. This example includes this information in case you want to give the user the flexibility of selecting an existing instance of Express Edition to install your application against instead of always creating a new instance.

The first step is to create a simple class. This class will contain local variables of most of the command-line switches supported by the SQLEXP.exe installation executable. These switches will be exposed as properties of the class object.



**Important** The following code is only a guideline for installing SQL Server Express Edition with your custom application. It is not complete and does not contain robust error-handling routines.

Listing 14-6 *EmbeddedInstall* class definition

```
public class EmbeddedInstall
{
    #region Internal variables

    //Variables for setup.exe command line
    private string instanceName = "SQLEXPRESS";
    private string installSqlDir = "";
    private string installSqlSharedDir = "";
    private string installSqlDataDir = "";
    private string addLocal = "All";
    private bool sqlAutoStart = true;
    private bool sqlBrowserAutoStart = false;
    private string sqlBrowserAccount = "";
    private string sqlBrowserPassword = "";
    private string sqlAccount = "";
    private string sqlPassword = "";
    private bool sqlSecurityMode = false;
    private string saPassword = "";
    private string sqlCollation = "";
    private bool disableNetworkProtocols = true;
    private bool errorReporting = true;
    private string sqlExpressSetupFileLocation =
System.Environment.GetEnvironmentVariable("TEMP") + "\\sqlexpr.exe";

    #endregion
    #region Properties
    public string InstanceName
    {
        get
        {
            return instanceName;
        }
        set
        {
            instanceName = value;
        }
    }

    public string SetupFileLocation
    {
        get
        {
            return sqlExpressSetupFileLocation;
        }
        set
        {
            sqlExpressSetupFileLocation = value;
        }
    }

    public string sqlInstallSharedDirectory
    {
        get
```

```
        {
            return installSqlSharedDir;
        }
        set
        {
            installSqlSharedDir = value;
        }
    }
    public string SqlDataDirectory
    {
        get
        {
            return installSqlDataDir;
        }
        set
        {
            installSqlDataDir = value;
        }
    }
    public bool AutostartSQLService
    {
        get
        {
            return sqlAutoStart;
        }
        set
        {
            sqlAutoStart = value;
        }
    }
    public bool AutostartSQLBrowserService
    {
        get
        {
            return sqlBrowserAutoStart;
        }
        set
        {
            sqlBrowserAutoStart = value;
        }
    }
    public string SqlBrowserAccountName
    {
        get
        {
            return sqlBrowserAccount;
        }
        set
        {
            sqlBrowserAccount = value;
        }
    }
    public string SqlBrowserPassword
    {
        get
```

```
        {
            return sqlBrowserPassword;
        }
        set
        {
            sqlBrowserPassword = value;
        }
    }
    //Defaults to LocalSystem
    public string SqlServiceAccountName
    {
        get
        {
            return sqlAccount;
        }
        set
        {
            sqlAccount = value;
        }
    }
    public string SqlServicePassword
    {
        get
        {
            return sqlPassword;
        }
        set
        {
            sqlPassword = value;
        }
    }
    public bool UseSQLSecurityMode
    {
        get
        {
            return sqlSecurityMode;
        }
        set
        {
            sqlSecurityMode = value;
        }
    }
    public string SysadminPassword
    {
        set
        {
            saPassword = value;
        }
    }
    public string Collation
    {
        get
        {
            return sqlCollation;
        }
    }
}
```

```
        set
        {
            sqlCollation = value;
        }
    }
    public bool DisableNetworkProtocols
    {
        get
        {
            return disableNetworkProtocols;
        }
        set
        {
            disableNetworkProtocols = value;
        }
    }
    public bool ReportErrors
    {
        get
        {
            return errorReporting;
        }
        set
        {
            errorReporting = value;
        }
    }
    public string SqlInstallDirectory
    {
        get
        {
            return installSqlDir;
        }
        set
        {
            installSqlDir = value;
        }
    }
}

#endregion
```

Now that you have set up the local variables and properties for the class object, you can work on the public methods *IsExpressInstalled*, *EnumSQLInstances*, and *InstallExpress*.

Assuming a local server installation, you can simply look to the local registry to see if Express Edition or any other instance of SQL Server is installed. The method in Listing 14-7 enumerates and checks the *Edition* value for keys under this location, where *X* is an instance of SQL Server:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\MSSQL.X
```

Listing 14-7 *IsExpressInstalled* method

```
public bool IsExpressInstalled()
{
    using (RegistryKey Key =
Registry.LocalMachine.OpenSubKey("Software\\Microsoft\\Microsoft SQL
Server\\", false))
    {
        if (Key == null) return false;
        string[] strNames;
        strNames = Key.GetSubKeyNames();

        //If we cannot find a SQL Server registry key, we
don't have SQL Server Express installed
        if (strNames.Length == 0) return false;

        foreach (string s in strNames)
        {
            if (s.StartsWith("MSSQL."))
            {
                //Check to see if the edition is "Express Edition"
                using (RegistryKey KeyEdition =
Key.OpenSubKey(s.ToString() + "\\Setup\\", false))
                {
                    if ((string)KeyEdition.GetValue("Edition") ==
"Express Edition")
                    {
                        //If there is at least one instance of
SQL Server Express installed, return true
                        return true;
                    }
                }
            }
        }
    }
    return false;
}
```

By using the local registry, you can determine more information about all the SQL Server instances, regardless of edition, that are installed on the local server. Having this information is useful if you want to provide a better installation experience. The method in Listing 14-8 takes a reference and populates a string array for instances, editions, and versions. It returns the number of instances of SQL Server that are installed on the local computer.

Listing 14-8 *EnumSQLInstances* method

```
public int EnumSQLInstances(ref string[] strInstanceArray, ref string[] strEditionArray,
ref string[] strVersionArray)
{
    using (RegistryKey Key = Registry.LocalMachine.OpenSubKey("Software\\
Microsoft\\Microsoft SQL Server\\", false))
    {
```

```

        if (key == null) return 0;
        string[] strNames;
        strNames = Key.GetSubKeyNames();

        //If we can not find a SQL Server registry key, we return 0 for none
        if (strNames.Length == 0) return 0;

        //How many instances do we have?
        int iNumberOfInstances = 0;

        foreach (string s in strNames)
        {
            if (s.StartsWith("MSSQL."))
                iNumberOfInstances++;
        }

        //Reallocate the string arrays to the new number of instances
        strInstanceArray = new string[iNumberOfInstances];
        strVersionArray = new string[iNumberOfInstances];
        strEditionArray = new string[iNumberOfInstances];
        int iCounter = 0;

        foreach (string s in strNames)
        {
            if (s.StartsWith("MSSQL."))
            {
                //Get Instance name
                using (RegistryKey KeyInstanceName =
                    Key.OpenSubKey(s.ToString(), false))
                {
                    strInstanceArray[iCounter] =
                        (string)KeyInstanceName.GetValue("");
                }

                //Get Edition
                using (RegistryKey KeySetup =
                    Key.OpenSubKey(s.ToString() + "\\Setup\\", false))
                {
                    strEditionArray[iCounter] =
                        (string)KeySetup.GetValue("Edition");
                    strVersionArray[iCounter] =
                        (string)KeySetup.GetValue("Version");
                }

                iCounter++;
            }
        }
        return iCounter;
    }
}

```

Now you can install SQL Server Express Edition. First convert the properties of the class into a command-line argument that can be passed to the SQLEXPRESS.exe installation application. The *BuildCommandLine* method performs this task, as shown in Listing 14-9.

Listing 14-9 *BuildCommandLine* method

```
private string BuildCommandLine()
{
    StringBuilder strCommandLine = new StringBuilder();

    if (!string.IsNullOrEmpty(installSqlDir))
    {
        strCommandLine.Append("INSTALLSQLDIR=").Append(installSqlDir)
            .Append("\");
    }

    if (!string.IsNullOrEmpty(installSqlSharedDir))
    {
        strCommandLine.Append("INSTALLSQLSHAREDDIR=").Append(installSqlSharedDir)
            .Append("\");
    }

    if (!string.IsNullOrEmpty(installSqlDataDir))
    {
        strCommandLine.Append("INSTALLSQLDATADIR=").Append(installSqlDataDir)
            .Append("\");
    }

    if (!string.IsNullOrEmpty(addLocal))
    {
        strCommandLine.Append(" ADDLOCAL=").Append(addLocal).Append("\");
    }

    if (sqlAutoStart)
    {
        strCommandLine.Append(" SQLAUTOSTART=1");
    }
    else
    {
        strCommandLine.Append(" SQLAUTOSTART=0");
    }

    if (sqlBrowserAutoStart)
    {
        strCommandLine.Append(" SQLBROWSERAUTOSTART=1");
    }
    else
    {
        strCommandLine.Append(" SQLBROWSERAUTOSTART=0");
    }

    if (!string.IsNullOrEmpty(sqlBrowserAccount))
    {
        strCommandLine.Append("SQLBROWSERACCOUNT=").Append(sqlBrowserAccount)
            .Append("\");
    }

    if (!string.IsNullOrEmpty(sqlBrowserPassword))
```

```
{
    strCommandLine.Append("SQLBROWSERPASSWORD=\\")
        .Append(sqlBrowserPassword).Append("\\");
}

if (!string.IsNullOrEmpty(sqlAccount))
{
    strCommandLine.Append(" SQLACCOUNT=\\").Append(sqlAccount)
        .Append("\\");
}

if (!string.IsNullOrEmpty(sqlPassword))
{
    strCommandLine.Append(" SQLPASSWORD=\\").Append(sqlPassword)
        .Append("\\");
}

if (sqlSecurityMode == true)
{
    strCommandLine.Append(" SECURITYMODE=SQL");
}

if (!string.IsNullOrEmpty(saPassword))
{
    strCommandLine.Append(" SAPWD=\\").Append(saPassword).Append("\\");
}

if (!string.IsNullOrEmpty(sqlCollation))
{
    strCommandLine.Append(" SQLCOLLATION=\\").Append(sqlCollation)
        .Append("\\");
}

if (disableNetworkProtocols == true)
{
    strCommandLine.Append(" DISABLENETWORKPROTOCOLS=1");
}
else
{
    strCommandLine.Append(" DISABLENETWORKPROTOCOLS=0");
}

if (errorReporting == true)
{
    strCommandLine.Append(" ERRORREPORTING=1");
}
else
{
    strCommandLine.Append(" ERRORREPORTING=0");
}

return strCommandLine.ToString();
}
```

Now you can create the *InstallExpress* method, as shown in Listing 14-10.

**Listing 14-10** *InstallExpress* method

```
public bool InstallExpress()
{
    //In both cases, we run Setup because we have the file.
    Process myProcess = new Process();
    myProcess.StartInfo.FileName = sqlExpressSetupFileLocation;
    myProcess.StartInfo.Arguments = "/qb " + BuildCommandLine();
    /*      /qn -- Specifies that setup run with no user interface.
           /qb -- Specifies that setup show only the basic
user interface. Only dialog boxes displaying progress information are
displayed. Other dialog boxes, such as the dialog box that asks users if
they want to restart at the end of the setup process, are not displayed.
*/
    myProcess.StartInfo.UseShellExecute = false;

    return myProcess.Start();
}
```

Finally, we can create the sample application that calls the wrapper class and installs Express Edition, as shown in Listing 14-11.

**Listing 14-11** Main application

```
class Program
{
    static void Main(string[] args)
    {
        EmbeddedInstall EI = new EmbeddedInstall();

        if (args.Length > 0)
        {
            int i = 0;
            while (i < args.Length)
            {
                if ((string)args[i].ToUpper() == "-v")
                {
                    string[] strInstanceArray = new string[0];
                    string[] strVersionArray = new string[0];
                    string[] strEditionArray = new string[0];

                    int iInstances = EI.EnumSQLInstances(ref strInstanceArray,
                        ref strEditionArray, ref strVersionArray);
                    if (iInstances > 0)
                    {
                        for (int j = 0; j <= iInstances - 1; j++)
                        {
                            Console.WriteLine("SQL Server Instance:
                                \" + strInstanceArray[j].ToString() + "\" -- " +
```

```

strEditionArray[j].ToString() + " (" + strVersionArray[j].ToString() +
    ")");
        }
    }
    else
    {
        Console.WriteLine("No instance of SQL Server
            Express found on local server.\n\n");
    }

    return;
}

if ((string)args[i].ToUpper() == "-I")
{
    if (EI.IsExpressInstalled())
    {
        Console.WriteLine("An instance of SQL Server
            Express is installed.\n\n");
    }
    else
    {
        Console.WriteLine("There are no SQL Server
            Express instances installed.\n\n");
    }

    return;
}

i++;
}
}

Console.WriteLine("\nInstalling SQL Server 2005 Express Edition\n");

EI.AutostartSQLBrowserService = false;
EI.AutostartSQLService = true;
EI.Collation = "SQL_Latin1_General_Cp1_CS_AS";
EI.DisableNetworkProtocols = false;
EI.InstanceName = "SQLEXPRESS";
EI.ReportErrors = true;
EI.SetupFileLocation = "C:\\Downloads\\sqlexpr.exe";
    //Provide location for the Express setup file
EI.SqlBrowserAccountName = ""; //Blank means LocalSystem
EI.SqlBrowserPassword = ""; // N/A
EI.SqlDataDirectory = "C:\\Program Files\\Microsoft SQL Server\\";
EI.SqlInstallDirectory = "C:\\Program Files\\";
EI.SqlInstallSharedDirectory = "C:\\Program Files\\";
EI.SqlServiceAccountName = ""; //Blank means LocalSystem
EI.SqlServicePassword = ""; // N/A
EI.SysadminPassword = "ThISISALongPaSswOrd1234!"; //<<Supply
    a secure sysadmin password>>
EI.UseSQLSecurityMode = true;

```

```
        EI.InstallExpress();

        Console.WriteLine("\nInstalling custom application\n");

        //
        If you need to run another MSI install, remove the following comment lines
        //and fill in information about your MSI

        /*Process myProcess = new Process();
        myProcess.StartInfo.FileName = "";//
        <<Insert the path to your MSI file here>>
        myProcess.StartInfo.Arguments = ""; //
        <<Insert any command line parameters here>>
        myProcess.StartInfo.UseShellExecute = false;
        myProcess.Start();*/

    }
```

## Deploying Express Edition Applications Using ClickOnce

ClickOnce is a new feature that is part of the .NET Framework 2.0. ClickOnce lets you deploy Windows-based client applications to a computer by placing the application files on a Web or file server that is accessible to the client, and then providing the user with a link. This lets users download and run applications from centrally managed servers without requiring administrator privileges on the client machine.

In this section, we will illustrate the ClickOnce/SQL Server Express Edition experience by developing a simple Windows Forms application. This application uses the AdventureWorks sample database, which can be downloaded from <http://go.microsoft.com/fwlink/?linkid=65209>.

This example demonstrates how to create a single WinForm for viewing the departments in the HumanResources.Department table in the AdventureWorks database.

To create a Windows Form that displays the Department table:

1. Launch Visual Studio.
2. Create a new Windows Application project.
3. When the Form1 Designer opens, add a reference to the AdventureWorks database.
4. Right-click the Project node in the Solution Explorer pane, and then select both Add and Existing Item. Navigate to the AdventureWorks database and click OK.
5. In the Data Source Configuration Wizard, under the Tables node, select the Department table, and then continue with the wizard.
6. When the wizard finishes, you will notice the AdventureWorks.mdf database icon in the Solution Explorer pane and a new AdventureWorks connection in Database Explorer.

Database Explorer lets you perform database operations such as creating new tables, querying and modifying existing data, and other database development functions.

7. Add the *DataGridView* control to the WinForm. This grid control is located in the tool box. When you drag the grid control onto the design surface, you have the option of selecting the AdventureWorks dataset that you created when you ran the Data Source Configuration Wizard. This dialog box is shown in Figure 14-16.



**Figure 14-16** *DataGridView* task properties

When a data source is configured, you should be able to run the application and have the grid control display the values for the Department table, as shown in Figure 14-17.

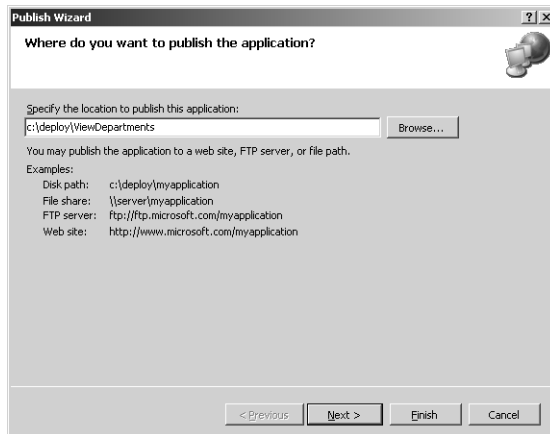
DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and D...	6/1/1998
2	Tool Design	Research and D...	6/1/1998
3	Sales	Sales and Market...	6/1/1998
4	Marketing	Sales and Market...	6/1/1998
5	Purchasing	Inventory Manag...	6/1/1998
6	Research and D...	Research and D...	6/1/1998
7	Production	Manufacturing	6/1/1998
8	Production Control	Manufacturing	6/1/1998
9	Human Resources	Executive Gener...	6/1/1998
10	Finance	Executive Gener...	6/1/1998
11	Information Servi...	Executive Gener...	6/1/1998
12	Document Control	Quality Assurance	6/1/1998

**Figure 14-17** Department table enumerated using the *DataGridView* control

You can now deploy this application using ClickOnce.

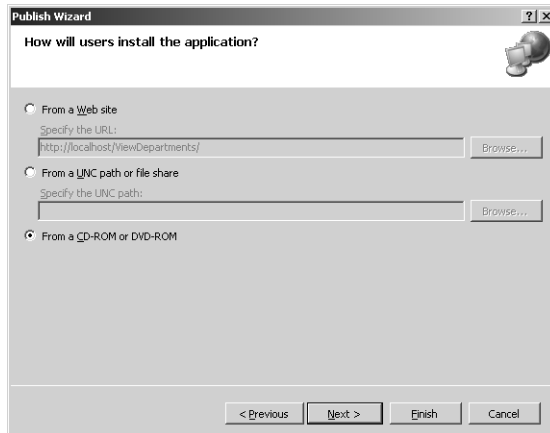
To deploy the application by using ClickOnce:

1. To publish the application, select Publish from the Build menu. The Publish Wizard opens, as shown in Figure 14-18.



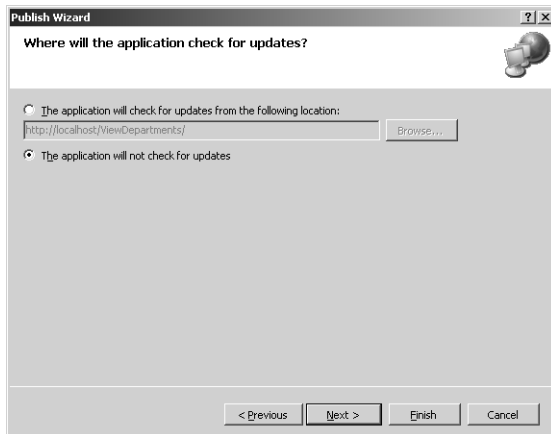
**Figure 14-18** The Publish Wizard's Where Do You Want To Publish The Application? page

2. On the first page of the wizard, you specify where the compiled bits should physically be placed. In the Specify The Location To Publish This Application box, enter **C:\deploy\ViewDepartments**. Click Next.
3. On the next page of the wizard (Figure 14-19), you specify the location from which users will install the application. Select **From A CD-ROM Or DVD-ROM**. Click Next.



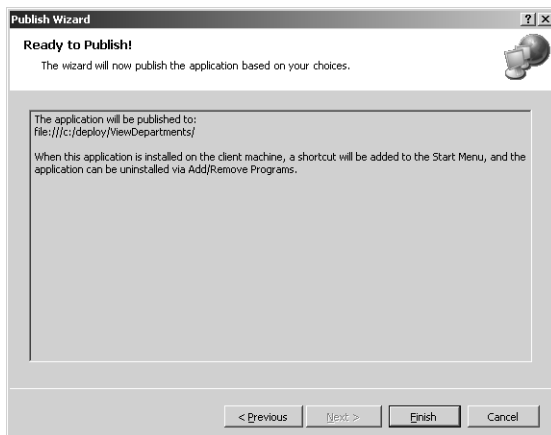
**Figure 14-19** The Publish Wizard's How Will Users Install The Application? page

4. On the next page of the wizard (Figure 14-20), you specify whether the application will check for updates.



**Figure 14-20** The Publish Wizard's Where Will The Application Check For Updates? page

5. ClickOnce gives applications the ability to look for updates at certain times, such as when the application starts or whenever the application developer chooses to call the appropriate update APIs. (There are some issues when you use this feature with a database, which we will discuss later in this chapter.) For this example, select The Application Will Not Check For Updates. Click Next.
6. The last page of the wizard (Figure 14-21) displays summary information and notifies you that, because you are writing to a CD or DVD-ROM, Setup will install a shortcut and entry in Add Or Remove Programs for your application. Click Finish.



**Figure 14-21** The last page of the Publish Wizard

You can write an application that will live on the application server only and will never be installed on the client machine. Regardless, ClickOnce will prompt the user to install any missing prerequisites, such as the .NET Framework 2.0 or SQL Server Express Edition, as shown in Figure 14-22.

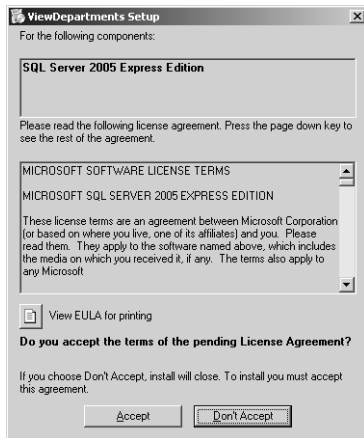


Figure 14-22 Prerequisites not installed when user launches application



**Note** Whether the application itself is designed to be run on demand from an application server or to be installed locally, SQL Server Express Edition is always installed on the local machine if the custom application requires it.

When the Publish Wizard finishes, new files are placed in the deployment directory. These files include the compressed data files and the setup installer application. You might want to copy these files to a CD and distribute them to your users, to provide them with the necessary information about applications that use SQL Server Express Edition.



**Important** A user who is not an administrator on the local machine cannot install the .NET Framework or SQL Server Express Edition. In this case, system administrators should deploy these components first. They can do this manually or by using a distributed software management system such as Microsoft Systems Management Server.

## Updating ClickOnce Deployments That Use Express Edition

Let's say the user has successfully installed your application. The user had all necessary prerequisites installed, and the application is running successfully.

The user has entered data into version 1.0 of the database and now the developers have come out with version 2.0. This new version has an additional column named Location in the Departments table. This new column stores the geographical location of the department. When the developer deploys version 2.0, the new version of the database is pushed down to the client, and the previous version is automatically moved to a separate folder named Pre. The developer must now write a database migration script to move all the data from the 1.0 version in the Pre folder to the new database. Because Visual Studio does not have any tools to support this migration, it is completely up to the developer to perform the migration.

Otherwise, none of the data that was entered in version 1.0 will be accessible to the application. Additionally, if the developer publishes an interim version (for example, 2.1) to reconcile this migration problem, or if the developer accesses the .mdf file by simply viewing the structure in Server Explorer, ClickOnce will see that the date and time stamp has changed and deploy version 2.1 of the database. This moves version 2.0 of the database to the Pre folder and deletes version 1.0 of the database. This results in complete data loss and a poor customer experience.

To avoid this, Visual Studio should not include the database files when the application is deployed. Instead, it should provide installation scripts to create the database. Also, when you perform a ClickOnce update, you must write and call a separate update script. The ViewDepartments example from the previous section is used in this section to help clarify the workaround solution.

ViewDepartments is a single WinForm application that connects to the AdventureWorks database and enumerates the Departments table. When you developed this application, you pointed Visual Studio to the AdventureWorks .mdf file, which created a new data source. As the application functions now, if you were to use ClickOnce to deploy the application, the application would always include the AdventureWorks .mdf file and cause the overwrite problems mentioned previously.

To avoid unwanted data loss in your application:

1. In the Solution Explorer pane, click the AdventureWorks database icon, as shown in Figure 14-23. In the Properties pane, select Do Not Copy for the Copy To Output Directory property.

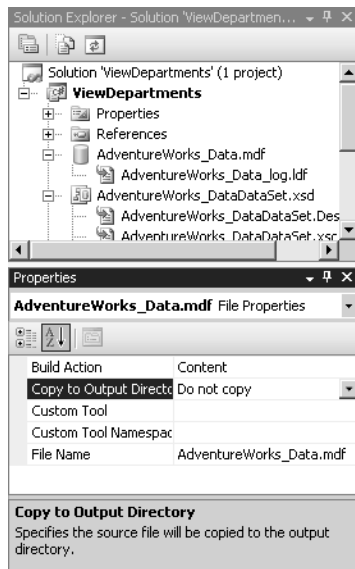


Figure 14-23 Copy To Output Directory property for the AdventureWorks database

2. Choose Properties from the Project menu to open the Project Properties panel. On the Publish tab, click Application Files. This launches a dialog box that contains a list of all the files in the solution. Change the Publish Status to Exclude for the .MDF and .LDF files of the AdventureWorks database, as shown in Figure 14-24.

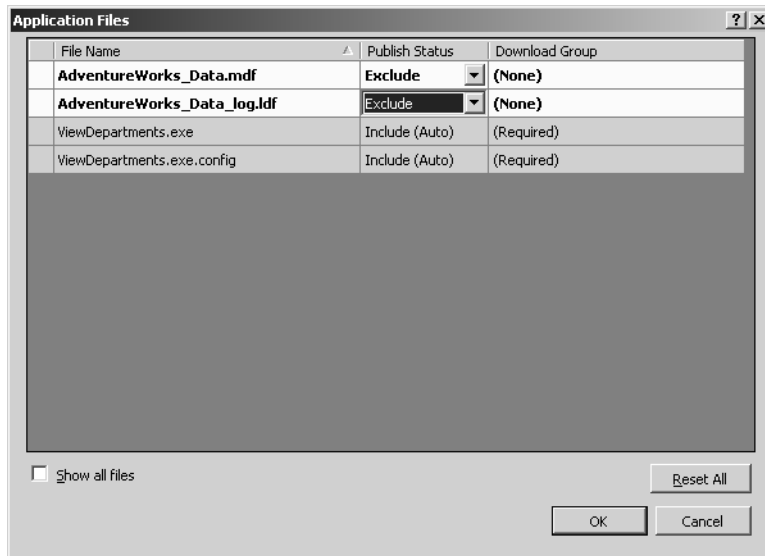


Figure 14-24 Excluding database files

Next you script the creation of the AdventureWorks database. You can script a database in many ways. In SQL Server Management Studio, you can right-click the database in Object Explorer and create the entire script there. Or you can use the Generate SQL Server Scripts Wizard for more scripting options. If you do not have a license for this tool or any other scripting tool, you can easily create a small program that uses the SQL Server Management Objects (SMO) object model to create a script by using the *Scripter* class.



**Note** If you installed SQL Server Express Edition with the developer components, the SMO DLLs are located by default in C:\Program Files\Microsoft SQL Server\90\SDK\Assemblies.

Listing 14-12 shows a modified AdventureWorks creation script that creates and populates the Departments table.

Listing 14-12 AdventureWorks creation script

```
USE [master]
GO
CREATE DATABASE [Adventureworks] ON PRIMARY
( NAME = N'Adventureworks_Data', FILENAME = N'C:\Program Files\Microsoft
SQL Server\MSSQL.1\MSSQL\Data\Adventureworks_Data.mdf' , SIZE = 167936KB
, MAXSIZE = UNLIMITED, FILEGROWTH = 16384KB )
```

```

LOG ON
( NAME = N'Adventureworks_Log', FILENAME = N'C:\Program Files\Microsoft
SQL Server\MSSQL.1\MSSQL\Data\Adventureworks_Log.ldf' , SIZE = 2048KB ,
MAXSIZE = 2048GB , FILEGROWTH = 16384KB )
COLLATE SQL_Latin1_General_CP1_CI_AS
GO
EXEC dbo.sp_dbcmptlevel @dbname=N'AdventureWorks', @new_cmptlevel=90
GO
USE [Adventureworks]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TYPE [dbo].[Name] FROM [nvarchar](50) NULL
GO
EXEC sys.sp_executesql N'CREATE SCHEMA [HumanResources] AUTHORIZATION [dbo]'
GO
CREATE TABLE [HumanResources].[Department](
    [DepartmentID] [smallint] IDENTITY(1,1) NOT NULL,
    [Name] [dbo].[Name] NOT NULL,
    [GroupName] [dbo].[Name] NOT NULL,
    [ModifiedDate] [datetime] NOT NULL CONSTRAINT
[DF_Department_ModifiedDate] DEFAULT (getdate()),
    CONSTRAINT [PK_Department_DepartmentID] PRIMARY KEY CLUSTERED
(
    [DepartmentID] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Engineering','Research and Development')
GO
insert into [HumanResources].[Department](Name,Groupname) values('Tool
Design','Research and Development')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Sales','Sales and Marketing')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Marketing','Sales and Marketing')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Purchasing','Inventory Management')
GO
insert into [HumanResources].[Department](Name,Groupname) values('Research
and Development','Research and Development')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Production','Manufacturing')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Production Control','Manufacturing')

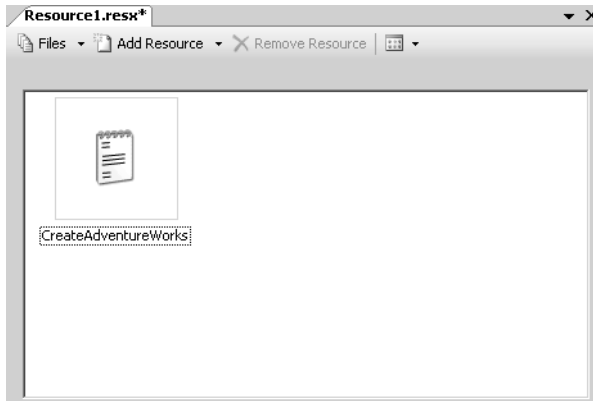
```

```
GO
insert into [HumanResources].[Department](Name,Groupname) values('Human
Resources','Executive General and Administration')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Finance','Executive General and Administration')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Information Services','Executive General and Administration')
GO
insert into [HumanResources].[Department](Name,Groupname) values('Document
Control','Quality Assurance')
GO
insert into [HumanResources].[Department](Name,Groupname) values('Quality
Assurance','Quality Assurance')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Facilities and Maintenance','Executive General and
Administration')
GO
insert into [HumanResources].[Department](Name,Groupname) values('Shipping
and Receiving','Inventory Management')
GO
insert into [HumanResources].[Department](Name,Groupname)
values('Executive','Executive General and Administration')
GO
--This next table is used to identify the version of the database
CREATE TABLE Adventureworks..AppInfo
(Property nvarchar(255) NOT NULL,
Value nvarchar(255))
GO
INSERT INTO Adventureworks..AppInfo Values('Version','1.0.0.0')
GO
```

Because the actual .mdf file is not included in this solution, you must define and synchronize versions of the database that the application is connected to. An easy workaround is to add the AppInfo table to the AdventureWorks database. When you start the application, it should first check to see if the versions match. If they don't, the application should either run an upgrade script or fail. This is explained in more detail next.

To implement a version check, you add a resource file to your project and then store the script as an embedded resource within the application:

1. Right-click the project in the Solution Explorer pane and select Add, and then select New Item. Select Resource File, and then click Add. This launches the Resource File document window shown in Figure 14-25.
2. You can add the SQL scripts as separate strings or as text files; for simplicity, we'll store the Create and Update scripts as separate files within this resource. From the Add Resource drop-down menu, select Add Existing File. Locate the creation script we produced earlier and add this file.



**Figure 14-25** Resource document window showing our creation script

3. Next we'll create an upgrade script for the AdventureWorks database. Although you might not have to upgrade your application right away, you should also include the upgrade script to upgrade your database to version 1.0.0.3.

```
USE [Adventureworks]
GO
ALTER TABLE [HumanResources].[Department]
ADD Location char(2)
GO
UPDATE Adventureworks..AppInfo set value='1.0.0.3' where
Property='Version'
GO
```

4. Save this script as UpgradeAdventureWorks.sql. Add it to the resource file, as described earlier.
5. Modify the application to check versions and run any necessary scripts.



**Note** In the previous example, the *Form\_Load* method contains code that was autogenerated when we assigned the AdventureWorks dataset via the UI:  
`this.departmentTableAdapter.Fill(this.adventureWorks_DataDataSet.Department);`  
 You should remove or comment out this code because you want to perform the database version check first.

You should also set the *DataSource* property (which was pre-populated in the grid control when you used the UI to bind the grid to the data source) to *None* (Figure 14-26).

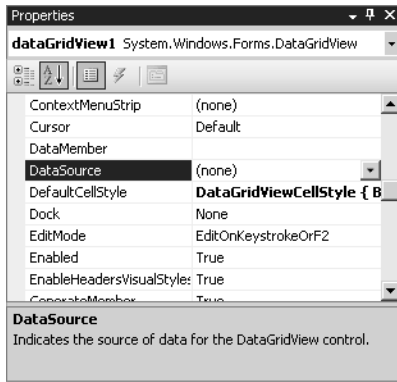


Figure 14-26 *DataSource* property autogenerated by Visual Studio

Listing 14-13 shows the complete code for the *Form1* class:

Listing 14-13 *Form1.cs*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Text.RegularExpressions;

namespace ViewDepartments
{
    public partial class Form1 : Form
    {
        enum VersionCheck { Failed = 0, Equal, DatabaseIsMoreNew,
            DatabaseIsOlder, DatabaseNotFound };

        private SqlConnection sqlCon = new SqlConnection();
        private SqlCommand sqlCmd = new SqlCommand();

        public Form1()
        {
            InitializeComponent();

            if (SetupDatabase() == false)
            {
                return;
            }

            PopulateGrid();
        }
    }
}
```

```
public bool SetupDatabase()
{
    bool bContinue = false;

    //Create a connection to SQL Server
    try
    {
        sqlCon.ConnectionString = "Server=
        .\\sqlexpress;Integrated Security=true";
        sqlCon.Open();
    }
    catch (SqlException sql_ex)
    {
        MessageBox.Show("Fail to connect to SQL Server Express\n"
+ sql_ex.Number.ToString() + " " + sql_ex.Message.ToString());
        return bContinue;
    }

    //Now that you are connected to Express, check the database versions

    switch (CheckVersion())
    {
        case (int)VersionCheck.Equal:
            {
                bContinue = true;
                break;
            }
        case (int)VersionCheck.Failed:
            {
                bContinue = false;
                break;
            }
        case (int)VersionCheck.DatabaseIsOlder:
            {
                //Run the upgrade script
                bContinue = RunScript(Resource1.UpdateAdventureworks.ToString());
                break;
            }
        case (int)VersionCheck.DatabaseIsMoreNew:
            {
                bContinue = false;
                break;
            }
        case (int)VersionCheck.DatabaseNotFound:
            {
                //Run the creation script
                bContinue = RunScript(Resource1.CreateAdventureworks.ToString());
                break;
            }
        default:
            {
                bContinue = false;
                break;
            }
    }
}
```

```
        }

    }

    return bContinue;

}

public bool RunScript(string strFile)
{
    string[] strCommands;
    strCommands = ParseScriptToCommands(strFile);
    try
    {
        if (sqlCon.State != ConnectionState.Open) sqlCon.Open();

        sqlCommand.Connection = sqlCon;

        foreach (string strCmd in strCommands)
        {
            if (strCmd.Length > 0)
            {
                sqlCommand.CommandText = strCmd;
                sqlCommand.ExecuteNonQuery();
            }
        }
    }
    catch (SqlException sql_ex)
    {
        MessageBox.Show(sql_ex.Number.ToString() + " " +
            sql_ex.Message.ToString());
        return false;
    }

    return true;
}

public int CheckVersion()
{
    //Get Version information from application
    Version v=new Version(Application.ProductVersion.ToString());

    try
    {

        string strResult;

        //Verify that the Adventureworks Database exists
        sqlCommand = new SqlCommand("select count(*) from
            master..sysdatabases where name='Adventureworks'",sqlCon);
        strResult = sqlCommand.ExecuteScalar().ToString();
        if (strResult == "0")
        {
            sqlCon.Close();
            return (int)VersionCheck.DatabaseNotFound;
        }
    }
}
```

```

        }

        sqlCmd = new SqlCommand("SELECT value from
Adventureworks..AppInfo where property='version'", sqlCon);
        strResult=(string)sqlCmd.ExecuteScalar();

        Version vDb = new Version(strResult);

        sqlCon.Close();

        if (vDb == v)
            return (int)VersionCheck.Equal;

        if (vDb > v)
            return (int)VersionCheck.DatabaseIsMoreNew;

        if (vDb < v)
            return (int)VersionCheck.DatabaseIsOlder;

    }
    catch (SqlException sql_ex)
    {
        MessageBox.Show(sql_ex.Number.ToString() + " " +
            sql_ex.Message.ToString());
        return (int)VersionCheck.Failed;
    }
    catch (Exception system_ex)
    {
        MessageBox.Show(system_ex.Message.ToString());
        return (int)VersionCheck.Failed;
    }
}

return (int)VersionCheck.Failed;

}

public string[] ParseScriptToCommands(string strScript)
{
    string[] commands;
    commands = Regex.Split(strScript, "GO\r\n", RegexOptions.IgnoreCase);
    return commands;
}

public void PopulateGrid()
{
    String strCmd = "Select * from [Adventureworks].[HumanResources].[Department
]";

    SqlDataAdapter da;

    da = new SqlDataAdapter(strCmd, sqlCon);
    DataSet ds = new DataSet();
    da.Fill(ds, "Departments");
    dataGridView1.DataSource = ds;
    dataGridView1.DataMember = "Departments";
}

```

```
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        // TODO: This line of code loads data into the
        'adventureworks_DataDataSet.Department' table. You can move or remove this
        line as necessary.
        //this.departmentTableAdapter.Fill(this.adventureworks_DataDataSet.
        Department);
    }
}
}
```

In the code for `Form1.cs`, a call is made to `SetDatabase()`. This function first attempts to make a connection to SQL Server Express Edition. When that call succeeds, it calls into the `CheckVersion()` method, which checks to see if the AdventureWorks database exists. If it does, the method obtains the version number from the `AppInfo` table. If the AdventureWorks database does not exist, the creation script located in the resource file is executed. If the database version is earlier than the application version, the upgrade script is run.



**Note** The version that is compared against the database comes from the *File Version* property of the project. This property can be set within the Assembly Information dialog box (which is accessible from the Application tab in Project Properties).

When you first execute this application against a blank SQL Server Express Edition database, it creates the AdventureWorks database, and you see the four columns of the Departments table. The next time you execute this application, it will be upgraded to include another column in the table named Location.

## Summary

SQL Server Express Edition uses the same database engine as all the other editions of SQL Server, but with memory, disk, and some feature restrictions. Even with these restrictions, it is possible to develop and deploy a variety of Express Edition applications.

SQL Server Express Edition with Advanced Services, which was released simultaneously with Service Pack 1, allows users to develop reports and issue full-text queries against Express Edition databases. This upgrade comes with a stripped-down version of SQL Server Management Studio. The Advanced Services Toolkit also includes a graphical report designer used to create reports for Report Server. With all these features and functionality, SQL Server Express Edition provides users with a powerful relational database engine—for free.