Beginning SQL Server 2008 Express for Developers: From Novice to Professional

ISBN-13 (pbk): 978-1-4302-1090-0

ISBN-13 (electronic): 978-1-4302-1092-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewer: Vidya Vrat Agarwal
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Kim Wimpsett
Associate Production Director: Kari Brooks-Copony
Production Editor: Kelly Gunther
Compositor: Gina Rexrode
Proofreader: Lisa Hamilton
Indexer: Broccoli Information Management
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The source code for this book is available to readers at http://www.apress.com.

CHAPTER 1

SQL Server 2008 Express Overview and Installation

Welcome to *Beginning SQL Server 2008 Express*. This book is centered around the Express Edition of SQL Server 2008, which is a free download from Microsoft. Because you are reading this book, I assume you are interested in learning how to create solutions with Microsoft SQL Server 2008 Express (SSE) but have no prior knowledge of SSE and probably no experience with any database. The aim of this book is to bring you quickly to a level at which you are developing competently with SSE. This book is specifically designed for beginners or those who at this stage want to use only SSE. But you'll find this book useful for understanding the basics of any relational database, so moving from SQL Server to Oracle, DB2, MySQL, PostgreSQL, and so on, will be relatively easy.

This chapter covers the following topics:

- What is SSE?
- How do I know whether my hardware meets the requirements?
- Can I just confirm that I have the right operating system?
- What can I do with SSE?

We will also look at installing SSE, including the following:

- Installing SSE on a Windows Vista platform
- Options not installed by default
- Where to install SSE physically
- Multiple installations on one computer
- How SSE runs on a machine
- How security is implemented
- Logon IDs, especially the `sa` (system administrator) logon

What Is SSE?

SSE is a functionally reduced version of SQL Server 2008 aimed at several market areas. These include students who cannot afford to purchase the full SQL Server 2008 and any users who want to run SQL Server on lower-spec hardware. For example, if you have a web site based on a SQL Server host, you can develop your system on SSE and then migrate it to your web host without buying the full version.

SSE limits database size to 4GB and memory to 1GB.

■ **Note** If you have more than 1GB of memory, SSE will not use the excess.

SSE can support a wide variety of applications, both for small users and for the largest corporations. However, SSE falls squarely in the smaller application category, appropriate for a small number of users and probably one or two developers. It also targets applications that require a database without a GUI built in, like Microsoft Access, and it has the ability to scale up in size if required. If you start out on SQL Server Express, you can scale your application up to other SQL Server versions with potentially no, or very little, work. As you will see, it is easy to install, because it comes as one complete package for most of its functionality, with a simple install to be performed for the remaining areas if required.

Evolution of SQL Server

SQL Server has evolved over the years into the product it is today. Table 1-1 summarizes this process.

Table 1-1. *The Stages in the Evolution of SQL Server*

Year	Version	Description
1988	SQL Server	Joint application built with Sybase for use on OS/2.
1993	SQL Server 4.2 A desktop database	A low-functionality, **desktop** database, capable of meeting the data storage and handling needs of a small department. The concept of a database that was integrated with Windows and had an easy-to-use interface proved popular.
1995	SQL Server 6.05 A small business database	Major rewrite of the core database engine. First "significant" release, occurring a year after Microsoft splits from Sybase. Improved performance and significant feature enhancements. Still a long way behind in terms of the performance and feature set of later versions, but with this version SQL Server became capable of handling **small e-commerce and intranet** applications, and it was a fraction of the cost of its competitors.

Year	Version	Description
1996	Major point	SQL Server was gaining prominence such that Oracle brought out version 7.1 on the NT platform as direct competition.
1998	SQL Server 7.0 A web database	Another significant rewrite to the core database. A defining release, providing a reasonably powerful and feature-rich database that was a truly viable (and still cheap) alternative for **small-to-medium businesses**, between a true desktop database such as Microsoft Access and the high-end enterprise capabilities (and price) of Oracle and DB2. Gained a good reputation for ease of use and for providing crucial business tools (e.g., analysis services, data transformation services) out of the box, which were expensive add-ons with competing databases.
2000	SQL Server 2000 An enterprise database	Vastly improved performance scalability and reliability saw SQL Server become a major player in the **enterprise database** market (supporting the online operations of businesses such as NASDAQ, Dell, and Barnes & Noble). A big increase in price (although still reckoned to be about half the cost of Oracle) slowed initial uptake, but the excellent range of management, development, and analysis tools won new customers. In 2001, Oracle (with 34 percent of the market) finally ceded its No. 1 position in the Windows database market (worth $2.55 billion in 2001) to SQL Server (with 40 percent of the market). In 2002 the gap had grown, with SQL Server at 45 percent and Oracle slipping to 27 percent. *Source: Gartner Report 5/21/2003*
2005	SQL Server 2005	Many areas of SQL Server were rewritten, such as the ability to load data via a utility called Integration Services, but the greatest leap forward was the introduction of the .NET Framework. This allowed .NET SQL Server–specific objects to be built, giving SQL Server the flexible functionality that Oracle already had with its inclusion of Java. This functionality was also available in SSE.
2008	SQL Server 2008	The aim of SQL Server 2008 is to deal with the many different forms that data can now take. It builds on the infrastructure of SQL Server 2005 by offering new data types such as XML and the use of Language Integrated Query (LINQ). It deals with compact devices and massive database installations. It also offers the ability to set rules within a framework to ensure databases and objects meet criteria that you define, and it offers the ability to report when objects do not meet your criteria.

Hardware Requirements

Now that you know a bit about SQL Server, the next big questions on your list may well be "Do I have a powerful enough computer to run SQL Server Express on? Will this help me refine my decision?"

Judging by today's standards of minimum specification hardware, even the low-cost solutions, the answer will in most cases be "Yes." However, you may have older hardware (technology moves so fast that even hardware bought a couple of months ago can be deemed below minimum specification), so let's take a look at what the minimum recommendations are and how you can check your own computer to ensure that you have sufficient resources.

CPU

The minimum recommended CPU that SQL Server will run on is a 1 GHz processor, a compatible processor, or similar processing power. However, as with most minimums listed here, Microsoft wholly recommends a faster processor, 2 GHz in fact. The faster the processor, the better your SQL Server will perform, and from this the fewer bottlenecks that could surface. Many of today's computers start at 2 GHz or faster, so you should already have a capable processor. If you have a lower-speed processor, try to invest in upgrading it. You will find your development time reduced for it.

However, it is not the processor alone that speeds up SQL Server. A large part also comes down to the amount of memory that your computer has.

Memory

Now that you know you have a fast enough processor, it is time to check whether you have enough memory in the system. Microsoft recommends 1GB or more, although the minimum RAM is 512MB.

The more memory, the better: I recommend a minimum of 1GB. If a process can be held in memory, rather than swapped out to the hard drive or another area while you are running another process, you are not waiting on SQL Server being loaded back into memory to start where it left off. This is called **swapping**, and the more memory, the less swapping could, and should, take place.

Together as a whole, CPU speed and memory are the two items crucial to the speed that the computer will run. Having sufficient speed will let you develop as fast as possible.

Insufficient memory won't stop the install, but you will be warned that you need more to actually run SQL Server afterward.

Hard Disk Space

You will need quite a lot just to install SQL Server and the help file called Books Online. Then you will have your data files on top of that. For SQL Server alone, ignoring any data files that you are then going to add on top, you will need more than 350MB of space and then a further 425MB of space for Books Online and the sample databases, if you plan on downloading and installing them. You can reduce this by opting not to install certain options; however, even most notebooks these days come with a minimum 40GB, and even 120GB is not uncommon. Hard disk space is cheap as well, and it is better to buy one disk too large for your needs than have one hard drive that suits now and then have to buy another later, with all the attendant problems of moving information to clear up space on the original drive.

Again, you will need spare space on the drive for the expansion of SQL Server and the databases, as well as room for temporary files that you will also need in your development process. So think big—big is beautiful!

Operating System Requirements

You will find that SQL Server 2008 will run on Windows Vista Home Basic Edition and above, as well as on Windows XP. From the server side, it will work on Windows Server 2003 with Service Pack 2 and Windows Server 2008. It will also work on the 64-bit operating systems for Windows XP Professional, as well as the 64-bit editions of Windows Server 2003 and 2008. So, there is plenty of scope for running SQL Server on many operating systems.

The Example

In order to demonstrate SQL Server 2008 Express, together we will develop a system for a financial company that will have features such as banking, purchasing shares, and regular buying, including a unit trust savings plan, and so on. This is an application that could fit into a large organization or with very minor modifications could be used by a single person to record banking transactions.

The book builds on this idea and develops the example, demonstrating how to take an idea and formulate it into a design with the correct architecture. It should be said, though, that the example will be the bare minimum to make it run, because I don't want to detract from SSE. The book will give you the power and the knowledge to take this example, expand it to suit your financial application needs, and give it the specifics and intricacies that are required to make it fully useful for yourself.

But before we can get to this point, we need to install SSE.

Installation

This chapter will guide you through the installation process of the Express Advanced Edition of SQL Server Express. Although much larger than the plain Express Edition, it does have extra functionality that we look at in Chapter 15, which involves creating reports.

Installation covers a great many different areas:

- Security issues
- Different types of installation—whether this is the first installation and instance of SQL Server Express or a subsequent instance, for development, test, or production
- Custom installations
- Installing only some of the products available

Most of these areas will be covered so that by the end of the chapter you can feel confident and knowledgeable to complete any subsequent installations that suit your needs.

A Standard Installation

Let's now take the time to install SSE on our machines. Microsoft offers SSE download options from `http://www.microsoft.com/sqlserver/2008/en/us/default.aspx`. By then selecting the download link, you should see the Express Advanced Edition listed to install.

Preparing to Install

First ensure that you have logged on to your machine with administrative rights so that you are allowed to create files and folders on your machine, which is obviously required for installation to be successful. You will also need to have Microsoft .NET Framework 3.5 Service Pack 1, Windows Installer 4.5, and Windows PowerShell installed on your machine, which can be downloaded from the Microsoft web site.

Once you start the setup and after you accept the SQL Server End User Agreement, SQL Server then installs some support files. These files are part of SQL Server that will be included in service packs and form part of the installation process. The main files are setup files and the required .NET Framework version if it is not already installed. .NET is a framework that Microsoft created that allows programs written in VB, C#, and other .NET-compliant programming languages to interoperate. SQL Server 2008 uses .NET for some of its own internal work, but also, as a developer, you can write .NET code in any of Microsoft's .NET languages and include this within SQL Server.

■ **Note** Including .NET code is an advanced topic and outside the scope of this book. For more information, try *Pro SQL Server 2005 Assemblies* by Robin Dewson and Julian Skinner (Apress, 2005).

You are then presented with the SQL Server Installation Center shown in Figure 1-1. A number of options appear on the left side of the SQL Server Installation Center. The first option, Planning, is mainly used for checking whether the installation computer meets the requirements for a successful install, as well as other documents concerning security, release notes, and so on. The second option, Installation, you will use to start the SQL Server installation. When you click the New SQL Server standalone installation item at the top of the Installation Center, the SSE installation starts. Click Next.

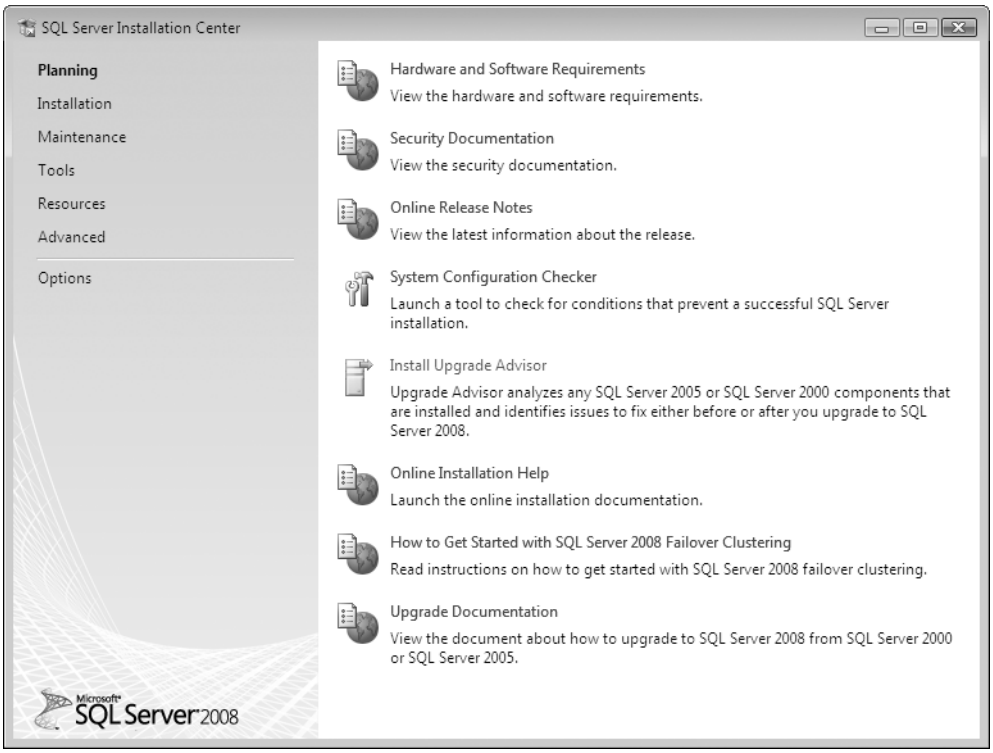

Figure 1-1. *Beginning the install*

The installation moves through several standard screens. I won't detail each screen here, because no interaction is required. These screens are used as part of SQL Server's preinstallation check. So, move past the screens that create some setup support rules, enter your product key, and accept the license; the installation then checks your computer to see whether any items that could be installed might fail and warns you.

Choosing the Features to Install

You now come to the Feature Selection screen, where you have to make some decisions. As you see in Figure 1-2, this installation will have everything installed, because this will be my development instance where developers will be testing every aspect of SQL Server away from any development of projects taking place. This is therefore going to be more of a training environment. Select every item listed.

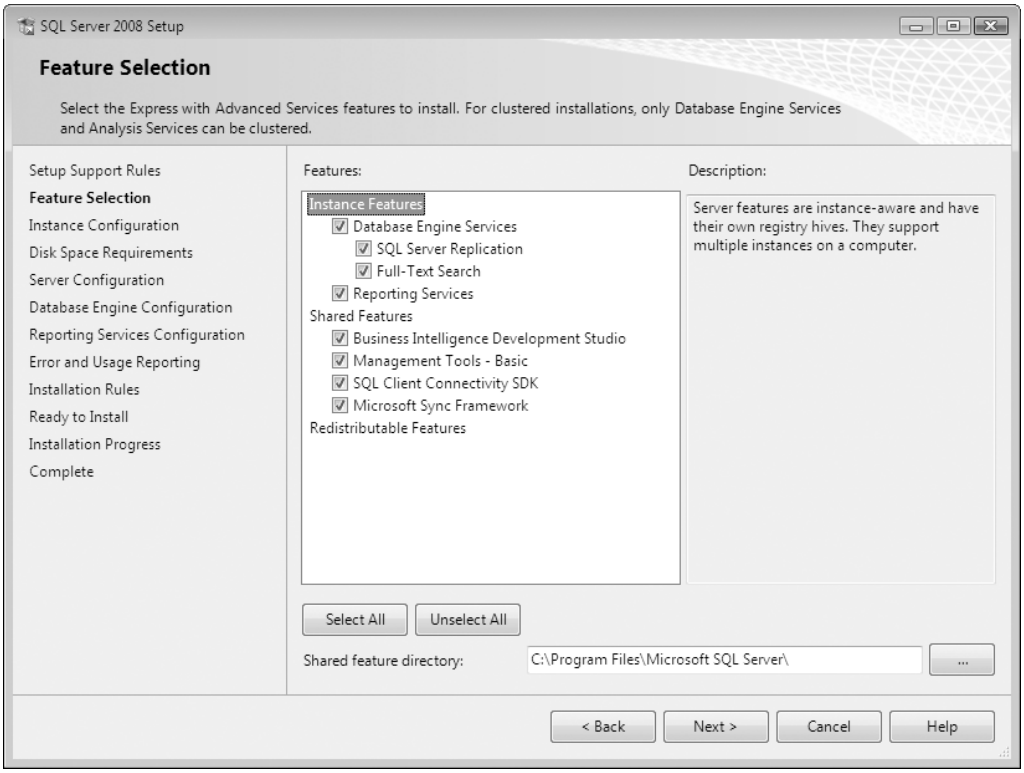

Figure 1-2. *Selecting every feature to install*

Let's briefly take a look at what each of these components are:

Database Engine Services: This is the core of SSE, and it installs the main engine, and so on, to make SQL Server run.

SQL Server Replication: This component, which will not be discussed at length in this book, allows you to replicate changes in one database to another automatically.

Full-Text Search: This allows you to search text-based data types quickly. We don't look at this in this book.

Reporting Services: In Chapter 15 you will see a demonstration of how to build a report using the data we build up as we progress through the book. Reporting Services deals with the processes required to build these reports.

Business Intelligence Development Studio: We will use this tool to build our report; however, if you want to use Analysis Services processing, which is where you are taking your data to perform data analysis, then this software would do this.

Management Tools – Basic: These are basic tools used to interact with SQL Server. The main tool we will use is the SQL Server Management Studio Express; we'll also use a tool for interacting with SQL Server using a command prompt, and we'll use SQL Server PowerShell, which we will take a quick look at in Chapter 7.

SQL Client Connectivity SDK: This is an SDK that you can use to build client connectivity tools.

Microsoft Sync Framework: This allows you to sync your data within SQL Server with offline applications running on devices such as PDAs, handheld PCs, tablet PCs, and so on.

Naming the Instance

As you know, SSE is installed on a computer. It is possible to install SQL Server more than once on one computer. This could happen when you have a powerful server and it has enough resources such as memory, processor, and so on, to cope with two or three different applications running. These different applications want to have their own SQL Server. Each install is called an **instance**. We are now at the stage that we can name the instance of the install. Each instance must have a unique name attached to it, although "no name," known as a **default instance**, is also classified as a unique name.

Naming instances is important as the first step to organizing your environments. For example, you may have an instance for development, another instance for system testing, and finally one for user testing. It is bad practice to share production server hardware with anything but production databases. If you do share and if you have a rogue development action that occurs and crashes the server, you will stop production from running.

The Default Instance option is available, which is what is selected when you are not giving the install a specific name; once you come to install SSE outside of a learning environment, you should avoid selecting the Default Instance option, because it gives you an installation with no name and therefore no hint as to its use. As you are still learning, so select Default instance as shown in Figure 1-3, and either accept the name SSE shows or put in a new name, then click Next. Once you have instances installed on the server, they will be listed in the Instance Configuration window shown in Figure 1-3. You can also see the path detailed for each of the directories for any services selected in the previous step.

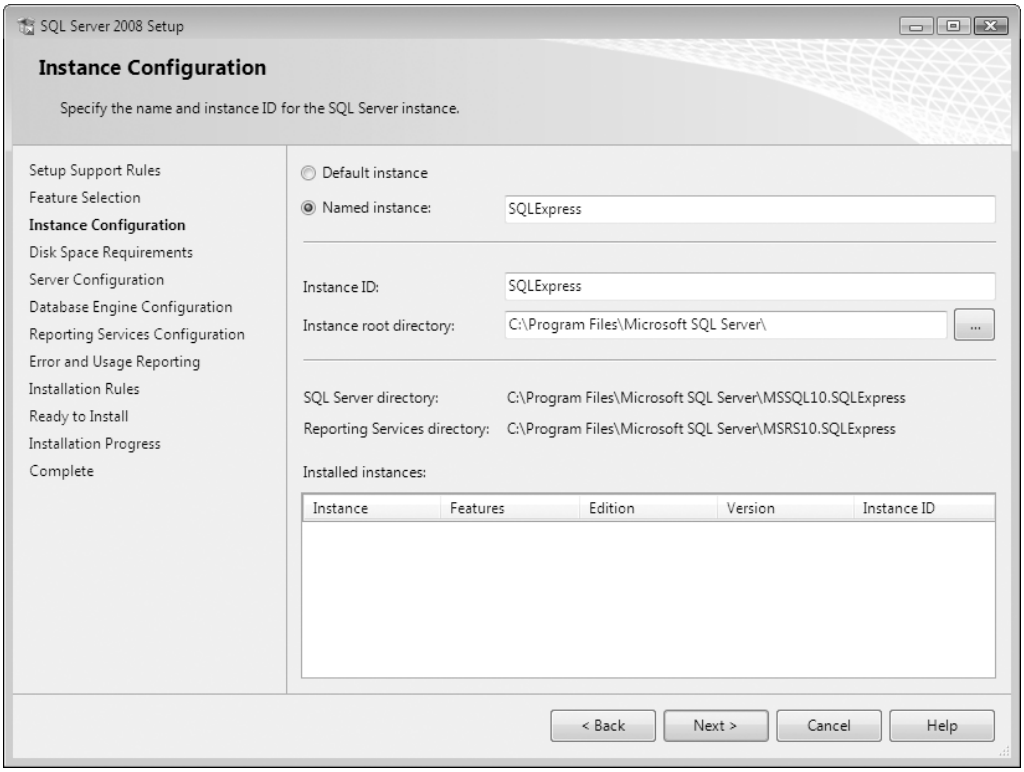

Figure 1-3. *Naming the install instance*

Service Accounts

Once you skip past the Disk Space Requirements calculations, you then come to the SQL Server Configuration screen. SSE requires you to log in to Windows before starting, just as you need to log in to Windows before using the system. SQL Server, Reporting Services, and so on, can run without you or anyone being logged in to the computer the install took place on. They can run just so long as the computer has fired up successfully. This is normal when SQL Server is installed on a server that is held in a remote location like a server room.

However, nothing can run on Windows without having some sort of login. If you do log in to the computer, as you likely will while working through this book because SSE will be running on a home or local system, you can use this Windows user ID for SQL Server to log in and start its service. This is known as a **local system account**.

On the other hand, you can create a Windows login that exists purely for SSE. This could exist for several reasons. For example, your Windows account should be set up so that the password expires so many days after being set, or it locks out after a number of incorrect password attempts. This is to protect your computer and the network, among many other areas. However, SSE should use a separate account that also has an expiring password and the ability to lock the account after a number of unsuccessful attempts. This kind of non-user-specific, "generic" account removes the link between SSE and a person within an organization.

Where you are installing your SSE will have an effect on the decision you make. Because you are just learning SSE, let's not make it complicated at this point; I assume that this install will be for you to use to learn SSE. Therefore, just select Use the NT AUTHORITY\SYSTEM account for the database service and the NT AUTHORITY/LOCAL SERVICE for the other three options, as you see in Figure 1-4. You can also define what services start when the computer is started up. Keep the defaults at the moment because you can always change these later via the Configuration Manager, which is found in Configuration Tools. For now, just click Next to proceed to the next step.

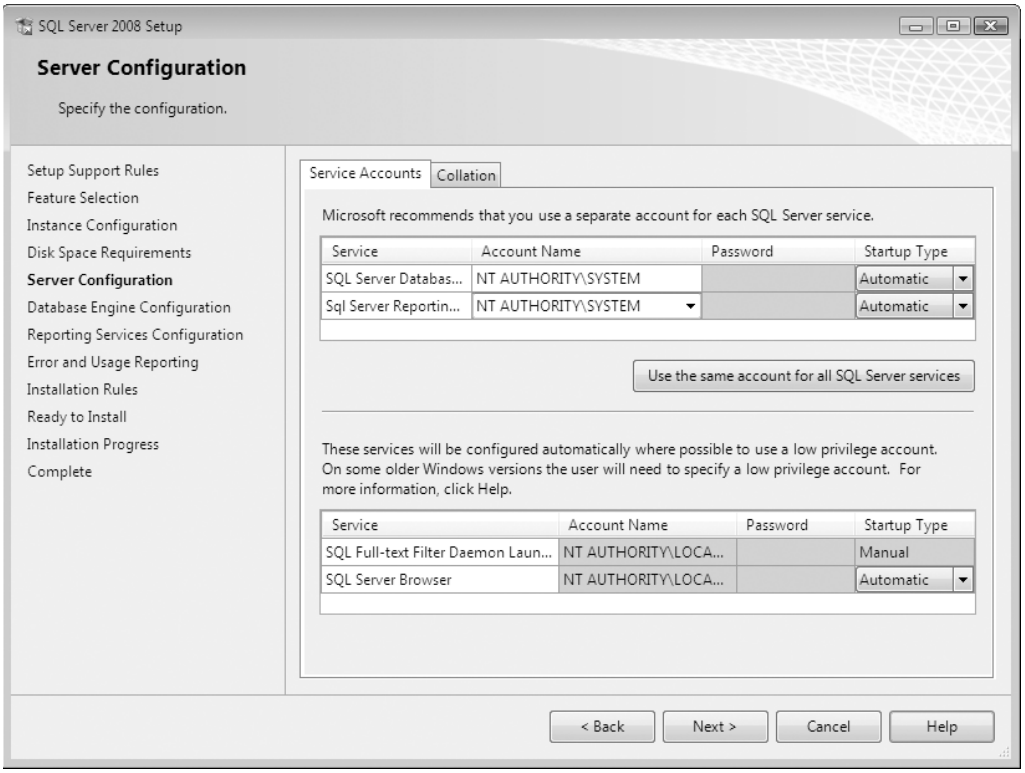

Figure 1-4. *Service account selection*

Authentication Mode

You now come to how you want to enforce security on your SSE installation. As Figure 1-5 shows, there are two choices: Windows authentication mode and mixed mode. You will learn more about modes later in the chapter but very, very simply, Windows authentication mode denotes that you are going to use Windows security to maintain your SQL Server logins, whereas mixed mode uses either Windows security or an SSE-defined login ID and password. You also need to define a password for a special login ID, called `sa`. Again, you will learn more about this in a moment, but for now you must enter a valid password. Use a meaningful and impossible-to-guess password.

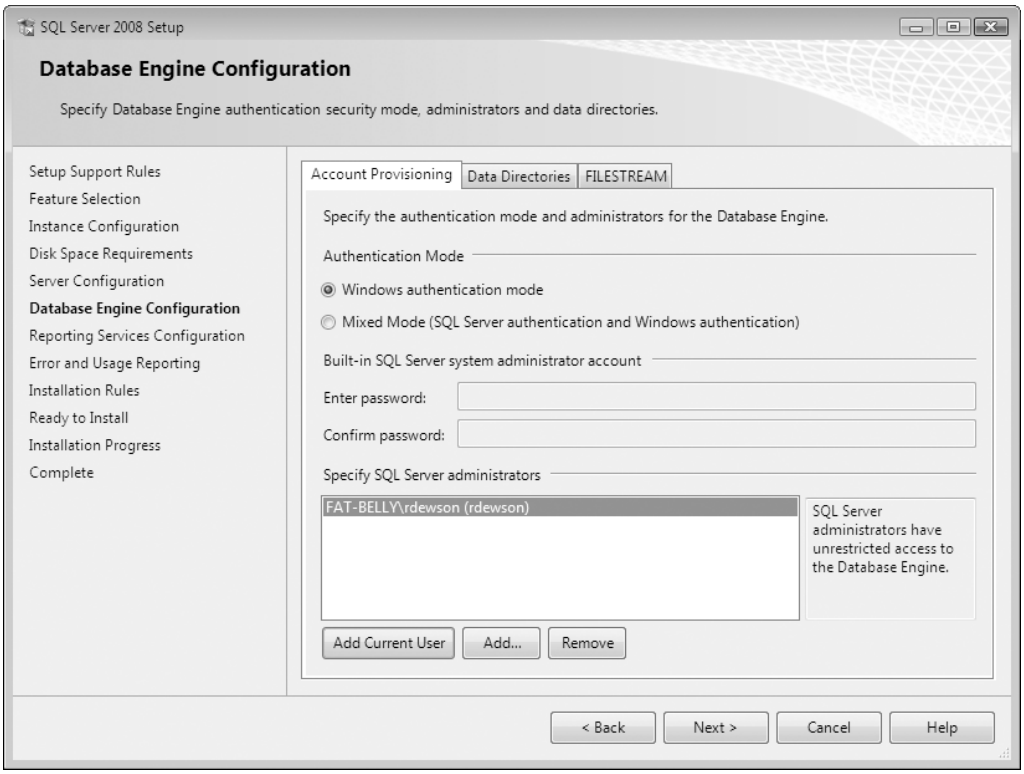

Figure 1-5. *Authentication choices*

It is also necessary to define a SQL Server Administrator account. This is a special account that you can use to log on with during dire emergencies, such as when SQL Server refuses connections. This special account allows you to log on, debug the situation, and get SQL Server back up and running. Normally, this Administrator account name would also be a server account ID. But for now, use the account you have used to log on to your computer.

Defining the Data Directories

The Data Directories tab, First as shown in Figure 1-6, is where you define where SQL Server stores its data, backup directories, and temporary database by default. For each instance, you define where the relevant folders should reside. As stated earlier, you can have several installations on one physical server, and it is not uncommon for a physical server to hold more than one production installation of SQL Server. For example, there could be an installation for customer accounts, another for product control, and so on. Each instance would have its data held in a different data directory. This includes any temporary databases that are created and any log files that are generated. So although the physical server is shared, each installation is isolated. Because you are dealing with only one instance within this book, leave these settings as they are.

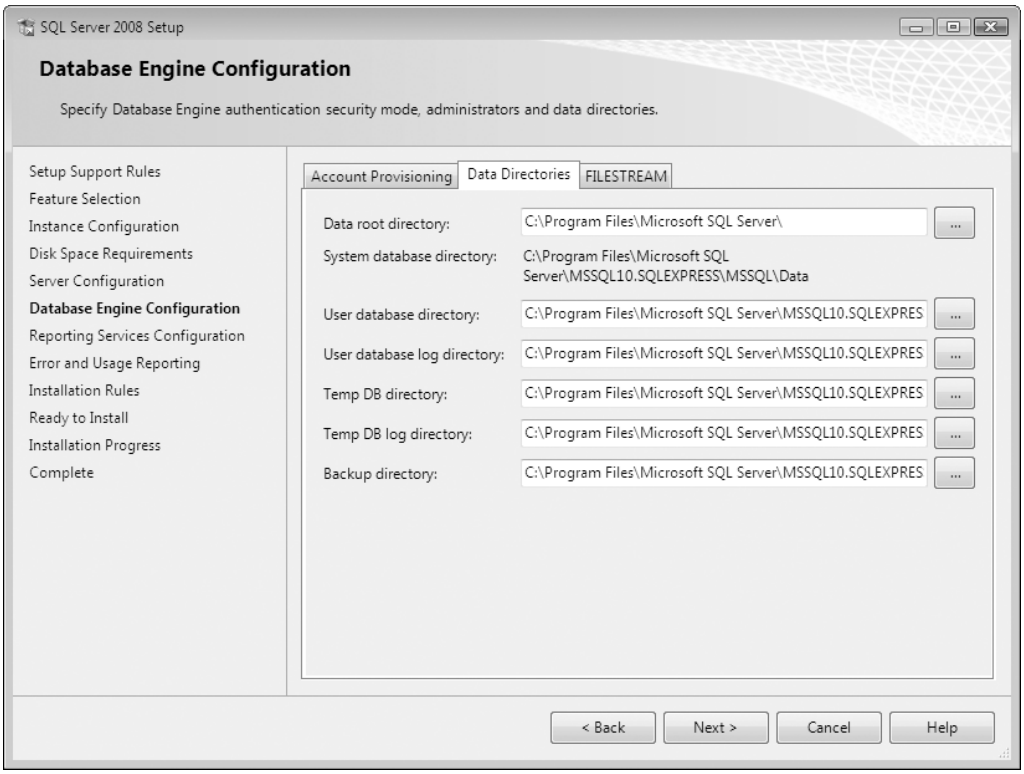

Figure 1-6. *Database data directory locations*

The FILESTREAM tab, which you can see in Figure 1-6 next to the Data Directories tab, allows you to define another type of data directory, but ignore that tab for now. However, to give you a small insight, FILESTREAM is used when dealing with large amounts of data and where it resides on the physical server.

Collation Settings

Collation settings specify how sorting and comparison of rows of data are dealt with within SSE. For example, a collation setting will inform SSE of details such as whether the system is case sensitive. It is possible to have different collation settings for each type of SQL service, shown in the earlier setup process, so you could sort one way for Analysis Services different from that defined for your main SSE installation. It would be only in exceptional circumstances that you would do this, because it will cause extra processing complications when using the same processing in more than one service.

Creating the Reporting Services Database

Because you selected Reporting Services to be installed, you need to create a database for the reporting server to use. There are three different possible installation options for Reporting

Services: native, SharePoint, and installed but not configured. If you select the last option, SQL Server Reporting Services will be installed on the server but will not be configured. This is ideal if you're setting up a specific server just for the reporting options. Once it's installed, you would then have to create a reporting database. The native mode configuration, as shown in Figure 1-7, is the simplest and the one you will be using. It installs Reporting Services and also creates the necessary databases within your SQL Server. It will be available only if you are installing on a local instance rather than a remote instance and if Reporting Services is also on that local instance. Default values are used for the service account, the report server URL on the local instance (which will be localhost), the Report Manager URL, and the name of the Reporting Services database.

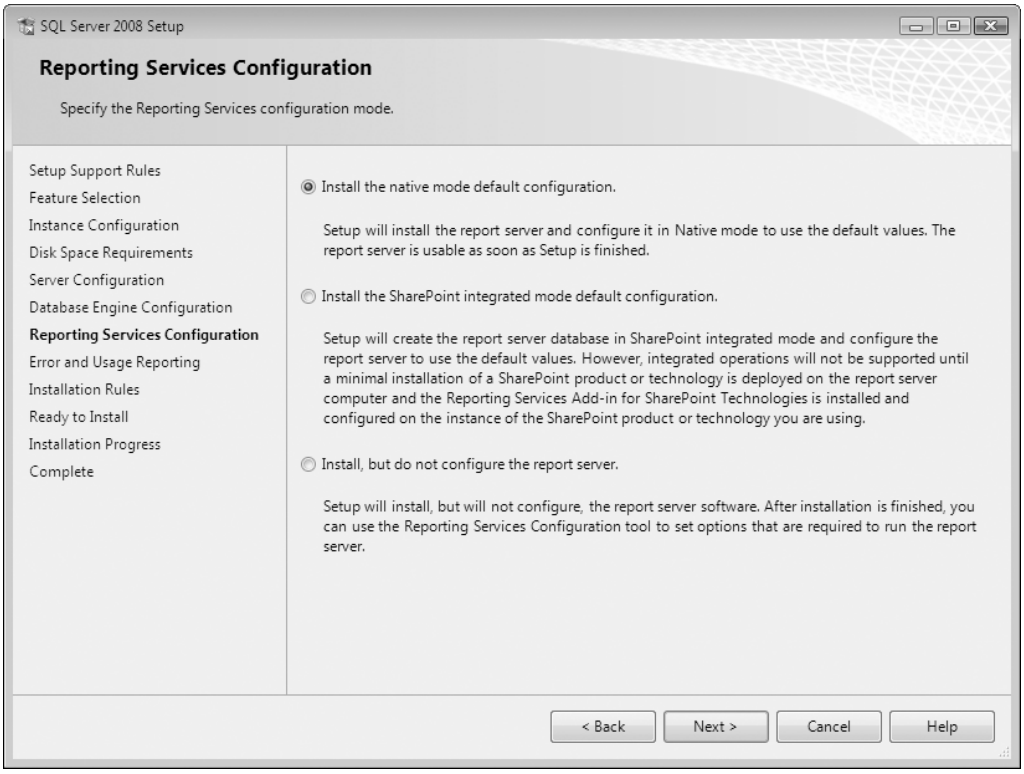

Figure 1-7. *Installing as default Reporting Services*

If you have a SharePoint installation and you want Reporting Services to use this architecture, then select this option, which allows you to use SharePoint's functionality. This is outside the scope of this book.

Error and Usage Report Settings

Within SSE, it is possible for any errors to be automatically reported and sent to Microsoft. These include fatal errors where SSE shuts down unexpectedly. It is recommended that you keep the error settings shown in Figure 1-8 enabled. No organizational information will be

sent; therefore, your data will still be secure. This is similar to sending reports when Excel crashes, for example. It is better to have these options enabled.

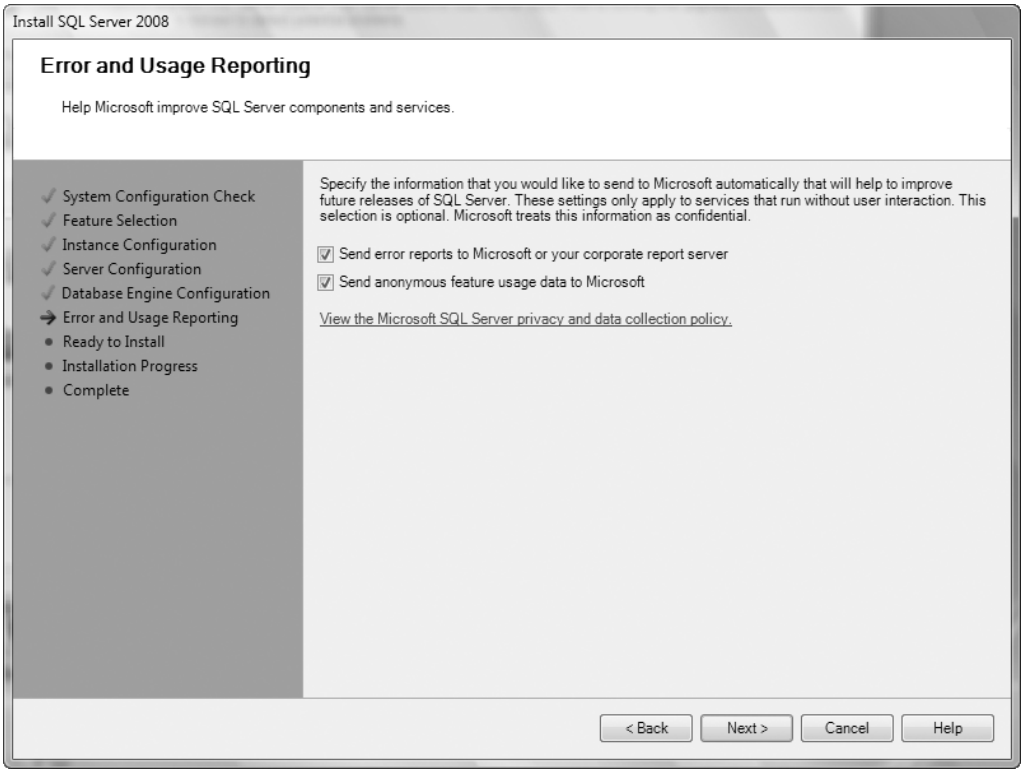

Figure 1-8. *Error and usage report settings*

And that is it; you are now ready to install.

Security

To discuss the Service Account dialog box that you came across in the installation properly (refer to Figure 1-4), you need to delve into the area of Windows security.

In this section, you will first examine the concept of Windows services as opposed to programs and then move on to discussing different types of authentication you can choose when installing SQL Server.

Services Accounts

SSE runs as a Windows service. So, what is a service? A good example of a **service** is any antivirus software that runs continuously from when the user restarts a computer to the point that the computer shuts down. A **program**, on the other hand, either is loaded in memory and running or is not started. So, what is the advantage of running a service? When you have a unit of work that can run as a service, Windows can control a great deal more concerning that

process. A service can be set to start automatically before any user has even logged on; all other programs require users to be logged in to Windows in order for them to start.

A service also has absolutely no user interface. There will be no form to display and no user input to deal with at run time. The only interaction with the process runs either through a separate user interface, which then links to the service but is a totally separate unit of work (for example, SQL Server Management Studio Express), or from Windows management of that service itself. Any output that comes from the service must go to the Event Log, which is a Windows area that stores any notification from the services that Windows runs.

Having no interface means that the whole process can be controlled without human intervention. Provided the service is designed well, Windows can take care of every eventuality itself and can also run the service before anyone has even logged in to the computer.

In most production environments, SSE will be running on a remote server, one probably locked away in a secure and controlled area, possibly where the only people allowed in are hardware engineers. There probably isn't even a remote access program installed, because this could give unauthorized access to these computers. SSE will run quite happily and, with any luck, never give an error. But what if one day there is an error? If SSE were running as a program, some sort of decision would have to be taken. Even if SSE crashed, there at least has to be some sort of mechanism to restart it. This means another process needs to be run, a monitoring process, which in itself could result in a whole ream of problems. However, as a service, SSE is under Windows' control. If a problem occurs, whether with SSE, Windows, or any outside influence, Windows is smart enough to deal with it through the services process.

If you do log in to the computer, as you likely will while working through this book, then you can specify that SQL Server should use your Windows user ID to also log in and start its service. The Windows user ID that SQL Server uses is known as a **local system account**.

On the other hand, you can create a Windows login that exists purely for SQL Server. You could do that for several reasons. For example, your Windows account should be set up so that the password expires after so many days or so that it locks out after a number of incorrect password attempts. This is to protect your computer and the network, amongst many other things. However, SQL Server should use a separate account that also has an expiring password and the ability to lock the account after a number of unsuccessful attempts. This kind of non-user-specific, "generic" account removes the link between SQL Server and a person within an organization. If you are looking at the domain account option, as shown earlier in Figure 1-4, this account is likely to be in a network environment or a production environment. There is an option to define a different account for each service. That ability is crucial when in a corporate environment because of the security implications that you must deal with.

SQL Server has several different processes that exist for different work. There is an executable named `sqlserver.exe` that is used to run SQL Server itself. You'll see that process listed in Task Manager as `MSSQL$instancename`. Including the instance name as part of the process name allows each instance to be unaffected if a different instance is stopped. Another executable named `sqlagent.exe` represents the SQL Server Agent, which runs processes such as batch jobs and is seen in Task Manager as `SQLAgent$instancename`. So even if you stop `MSSQL$instancename`, the `SQLAgent$instancename` process will continue. This behavior reinforces the fact that the SQL Server and SQL Server Agent processes should be on different accounts. Finally, SQL Server Agent, because it runs batch processes and complex tasks including working with other servers, needs a more powerful domain account than the SQL Server process. SQL Server Agent will need access to tape drives for backups, for example.

Your network administrator may have created these accounts and will know which account is best to use or best to create for these tasks.

It's time to move on to the options you are given during installation regarding authentication mode.

Looking at the Authentication Mode

Probably the most crucial information in the whole setup process, and also the biggest decision you have to make, concerns the authentication mode you want to apply to your server. As we saw earlier in the setup process, there are two choices: Windows authentication mode and mixed mode.

Windows Authentication Mode

To log on to a Windows 2003/XP/Vista machine, the user must supply a username. There is no way around this (unlike in Windows 9*x*/ME where a username was optional). So, to log on to Windows, the username and password have to be validated within Windows before the user can successfully log in. When this is done, Windows is actually verifying the user against username credentials held within the domain controller or, if you are running SQL Server on a standalone machine at home, the credentials held locally. These credentials check the access group the user belongs to (the user rights). The user could be an administrator, who has the ability to alter anything within the computer, all the way down to a basic user, who has very restricted rights. This then gives you a trusted connection; in other words, applications that are started up after logging in to Windows can trust that Windows has verified that the account has passed the necessary security checks.

Once you have logged in to Windows, SQL Server uses a trusted connection when working with Windows authentication mode. This means SQL Server is trusting that the username and password have been validated as I just mentioned. If, however, the username you give does not exist, you won't be able to log on to that machine. If the login isn't valid, SQL Server will check the Windows group that the user belongs to and check its security to see whether that group is set up to access SQL Server. If that user has administration rights to your computer, then the user may well be able to at least connect to SQL Server.

Someone else can also log on to your machine—provided they have access to it—with their user ID and password. Although they might be able to get to SQL Server by finding the executable on the C drive, SQL Server will first check to see whether that user has a valid login within SQL Server. If the login isn't valid, SQL Server will check the Windows group that the user belongs to and then check this user using its security model to see whether that group is set up to access SQL Server. If that user has administration rights to your computer, the user may well be able to at least connect to SQL Server.

SSE on Windows Vista Home works slightly differently with its security compared to Windows Vista Ultimate, Windows XP, Windows Server 2003, and so on. We look at this in detail in Chapter 4, but for now, in the upcoming text, it is important to compare the two different scenarios in general terms.

You are also in a bit of a catch-22 situation here. You need to know about security for your install process, but demonstrating security fully means working with SQL Server Management Studio Express, which the next chapter covers. I will keep the discussion simple here, so let's look at an example involving security now.

Try It Out: Windows Authentication Mode

1. Ensure that you are logged on to your machine as an administrator. If you are on a local computer, chances are that your login is in fact an administrator ID. If this computer is on a network and you are unsure about your access rights, ask your PC support desk to help you out with the ID and password. On Vista, you may need to change your user control access to avoid many dialog boxes confirming you want to proceed with each step.
2. Select Start ➤ Control Panel ➤ Users ➤ Manage Another Account ➤ Create a New Account, which is a link toward the bottom of the screen.
3. Once the Name the New Account dialog box comes up, enter the username **AJMason**, as shown in Figure 1-9. Ensure that the account type specified is Standard. This means it will not have administrator privileges. When done, click Create Account.

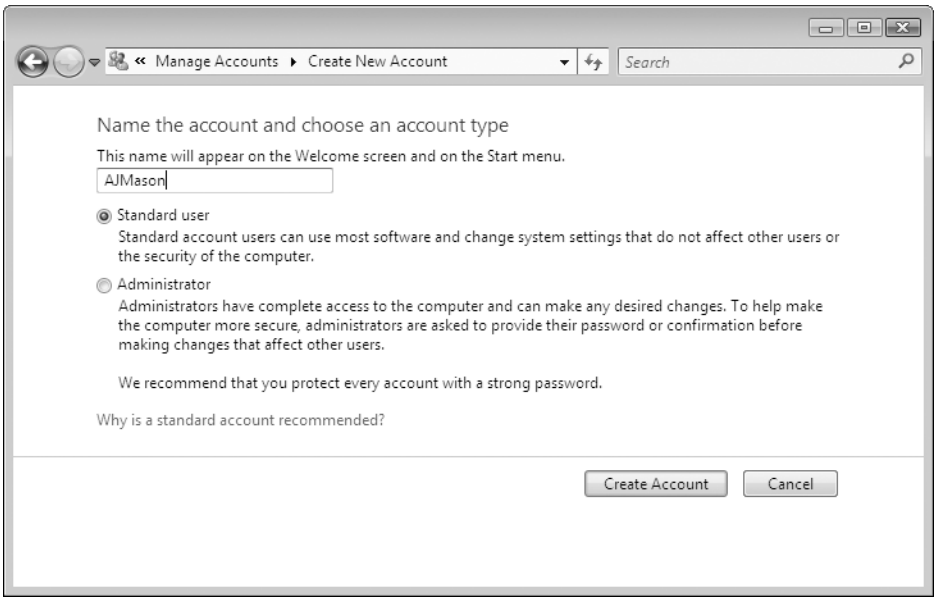

Figure 1-9. *Creating a new user account*

4. Complete steps 1–3 again because you want to add a second username using the following details:

   Username: VMcGlynn

   Account type: Computer Administrator

5. Log off of Windows, and then log on using the first ID you created, AJMason.
6. Once logged on, start up SQL Server Management Studio Express by selecting Start ➤ All Programs ➤ Microsoft SQL Server 2008 ➤ SQL Server Management Studio Express. You may need to populate the dialog box with the server name of the install. Click Browser for more, then select Database Engine, and finally select the install. You will go through this in more detail in Chapter 2.

You have created two usernames: one that has Standard access (AJMason) and one that has administration rights (VMcGlynn). However, neither of these specific usernames exists within SQL Server itself: after all, you haven't entered them, and they haven't appeared as if by magic.

The Windows security model has ensured that both IDs are valid. If the ID or password were incorrect, there would be no way you could be logged in to Windows. Therefore, when you try to connect to SQL Server, the only check that is performed is whether the user has access to SQL Server either via membership of an operating system group or through the specific logged-in user account. As you can see in Figure 1-10, neither AJMason nor VMcGlynn exists.

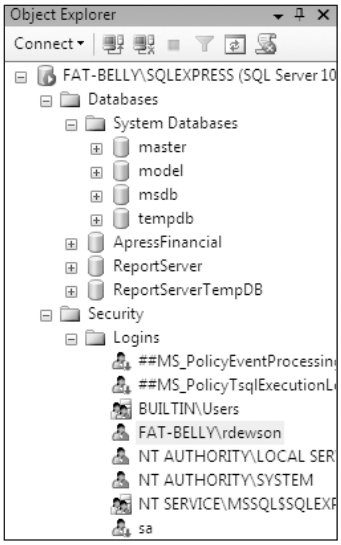

Figure 1-10. *Object Explorer for SQL Server*

However, you can see that there is a Windows group called `BUILTIN\Users`. This means that any username that is part of the `Users` group will have the capacity to log on to this SQL Server. This is not ideal and perhaps not what was expected or desired. Therefore, we need to deal with removing this group and replacing it with a more meaningful setup; otherwise, any login on the computer that SQL Server is installed on could access SQL Server. This is what will be demonstrated in Chapter 4 when you will learn more about securing your SQL Server installation.

Mixed Mode

If you installed SQL Server with mixed mode, this means you can use either Windows authentication, as has just been covered, or SQL Server authentication.

How does mixed mode differ from Windows authentication mode? To start with, you need to supply a user ID and password to connect rather than SQL Server taking the Windows ID, or the group the user belongs to, of the logged-in account. There is no assumption that the username supplied is a valid ID. Using mixed mode is also appropriate in many cases when working with Internet service providers (ISPs). To clarify this, if you are working on your

remote data at a local client machine, the remote machine would need to know the credentials of your login, and the easiest method is to use SQL Server authentication. Do not get confused here, though. If you want to work with your data at your ISP, the ISP may provide some sort of tool, or you may use SQL Server Management Studio Express to connect to your data. You would then be able to do what you want. The web site code, if written in ASP.NET, will use a Windows account to log in, so although you may lock out your SQL Server mixed mode account, it should not stop your web site from working.

You will learn how to add usernames to SQL Server (as opposed to adding Windows users) when I talk about security in Chapter 4.

This leaves one area of security that needs to be discussed here: the sa login.

The sa Login

The sa login is a default login that has full administration rights for SQL Server. If you selected mixed mode authentication during the installation process, you would be forced to include a password for this account. This is because the sa user ID is such a powerful login. It also exists in every SQL Server installation; therefore, any hacker knows that this user ID exists and so will try to connect to the server using it. Prior to SQL Server 2005 when creating a password became compulsory, many installations had the password blank, therefore allowing hackers instant access. If you logged in to SQL Server as sa, you will have full control over any aspect of SQL Server. SQL Server inserts this ID no matter which authentication mode you install. If you have a Windows account defined as sa—for example, for Steve Austin—then this user will be able to log in to the server if you have set the server up to implement Windows authentication mode without any further intervention on his part. Try to avoid login IDs of sa.

In a mixed mode installation, sa will be a valid username and validated as such. As you can guess, if any user gets ahold of this username and the password, that user will have full access to view and amend or delete any item of data. At worst, the user could corrupt any database, as well as corrupt SQL Server itself. The user could even set up tasks that e-mail data to a remote location as it is being processed.

It is essential to set up a strong password on the sa account in the Authentication Mode screen if you choose mixed mode. It is a major improvement in SQL Server 2008 that you are now forced to enter a password, although it is possible to set up a very easily guessed password. Do not use passwords such as *password* or *adminpwd*, for example. Always keep the password safe, but also make a note of it in a safe place. If you forget the sa password and this is the only administration ID that exists, you will need to reinstall SQL Server to get out of this problem. A good password is one that mixes numbers and letters but doesn't always include letters that can be made into numbers and numbers into letters in all cases. For example, *pa55word* is just as easy to guess as *password*. Or *4pr355* for Apress.

There is also another reason not to log on to SQL Server with the sa username. At times it will be essential to know who is running a particular query on a SQL Server database. In a production database, someone may be running an update of the data, which is filling up the disk space or filling up the transaction log. You need to contact that person to check whether he or she can stop the process. If that person logs in as sa, you will have no idea who he or she is. However, if that person logs on with an identifiable name, he or she will have an ID in SQL Server, which you can track. By restricting the sa login so that people have to use their own accounts, you can ensure a much higher degree of system monitoring and integrity.

There will be times that you want mixed mode authentication; it is perfectly acceptable to want this. Internet providers use mixed mode, because many applications may be on one web server. If this ISP is a reseller (in other words, many people around the globe use the one computer), you will not want these people to have the ability to see your data. We have also decided not to have `sa` as an administration logon at this point. So, what do you do? Well, you create a logon ID that will have the access privileges you want; in other words, it will have the ability to just see the data and work with the data you need, and no more. The ISP may require you to supply a user ID and password that it uses to create an account on its SQL Server instance. You will encounter more about this in Chapter 4.

■ **Note** Regardless of the authentication mode, it is important that you always supply a strong password.

Summary

By this point, you should understand the small differences between each version of SQL Server. You should also know how to check your computer to see whether it is suitable for a SQL Server installation.

By following the steps given earlier, you should have a successful installation of SQL Server on your computer. You may even have completed the installation twice so that you have a development server installation as well as a test server installation. This is a good idea and is something to consider if you have only one installation so far. Whether you are working in a large corporation or are a "one-man band," keeping your production and development code separate leads to greatly reduced complications if, when developing, you need to make a production fix.

This chapter introduced you to security in SQL Server so that you can feel comfortable knowing which way you want to implement this and how to deal with different usernames. You may not have any data yet, but you want to ensure that when you do, only the right people get to look at it!

You are now ready to explore SQL Server 2008 Express Edition. One of the best ways of managing SQL Server is by using SQL Server Management Studio Express, which will be discussed in the next chapter.