CHAPTER 1

SQL Server 2005 Express Overview and Installation

Welcome to *Beginning SQL Server 2005 Express*. This book is centered around the Express Edition of SQL Server 2005, which is a free download from Microsoft. As you are reading this book, I assume that you are interested in learning how to create solutions with Microsoft SQL Server 2005 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 wish 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, etc., will be relatively easy.

This chapter covers the following topics:

- What is SSE?
- How do I know if 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 then look at installing SSE—this section of the chapter covers the following:

- Installing SSE on a Windows XP 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 2005 aimed at several market areas. These include students who cannot afford to purchase the full SQL Server 2005 and any users who wish 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 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, as 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 gives a summary of 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 was a fraction of the cost of its competitors.
1996	SQL Server 6.5	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 engine. 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 MS 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.

Year	Version	Description
2000	SQL Server 2000 An enterprise database	Vastly improved performance scalability and reliability sees SQL Server become a major player in the **enterprise database** market (now 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% 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% of the market). In 2002 the gap had grown, with SQL Server at 45% and Oracle slipping to 27%. *Source: Gartner Report 5/21/2003*
2005	SQL Server 2005	Many areas of SQL Server have been rewritten, such as the ability to load data via a utility called Integration Services, but the greatest leap forward is the introduction of the .NET Framework. This will allow .NET SQL Server–specific objects to be built, giving SQL Server the flexible functionality that Oracle has with its inclusion of Java. This functionality is also available in SSE.

Hardware Requirements

Now that you know a bit about SQL Server, the next big question 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 Pentium III 500 MHz processor, a compatible processor, or similar processing power. However, as with most minimums listed here, Microsoft wholly recommends a faster processor, 1 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 above, and 500 MHz has not been the standard installation for a couple of years now. 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 processor alone that speeds up SQL Server. A large part is also 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 512MB or above, although the minimum RAM is 192MB.

The more memory, the better: I really would recommend a minimum of 512MB. If a process can be held in memory, rather than swapped out to 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, and having sufficient speed will let you develop as fast as possible.

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 over 350MB of space, 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 80GB 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 SSE will run on Windows 2000 Professional Edition and above with Service Pack 4, or all editions of Windows XP Service Pack 2 or above. It will also work on the 64-bit operating systems for Windows XP, as well as the 64-bit editions of Windows Server 2003. So there is plenty of scope for running SQL Server on many operating systems.

The Example

In order to demonstrate SQL Server 2005 Express, together we will develop a system for a financial company that will have features such as banking, share purchase, and regular buying, such as a unit trust savings plan, etc. 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, as 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 Edition with Advanced Services. Although double the size of the vanilla Express Edition and SQL Server Management Studio Express, it does have that extra functionality that we look at in Chapter 14, which discusses 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://msdn.microsoft.com/vstudio/express/sql/register/default.aspx`.

Preparing to Install

First of all, 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 have to have Microsoft .NET Framework 2.0 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 programming languages to interoperate. SQL Server 2005 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, 2006).

You are then presented with the welcome screen shown in Figure 1-1. Click Next.

Figure 1-1. *Beginning the install*

We then come to the System Configuration Check, as you see in Figure 1-2. Its main function is to check that the PC meets the hardware and software requirements. There are certain requirements for certain parts of the installation; for example, SQL Server Reporting Services, a tool for producing reports from SQL Server that was an add-on with SQL Server 2000, requires Internet Information Services (IIS). IIS is a process that runs on Windows XP Professional that provides the ability to run a web server. If you are on Windows XP Home, there will be a warning message indicating IIS has not been installed and later in the process reporting services will not be able to be installed. SQL Server Reporting Services is web based.

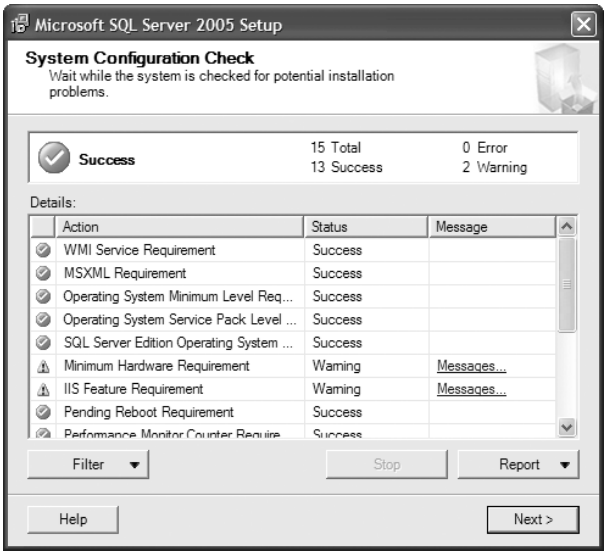

Figure 1-2. *System configuration tool with IIS warning*

Tip You may well see the Minimum Hardware Requirement warning come up whether you meet the requirements or not. Providing you meet the requirements detailed in this book, you should be fine.

You are then required to enter your registration information, as shown in Figure 1-3. Uncheck the tick box detailed as Hide Advanced Configuration Options. This allows you to deal with looking at settings such as how SQL Server runs among other items.

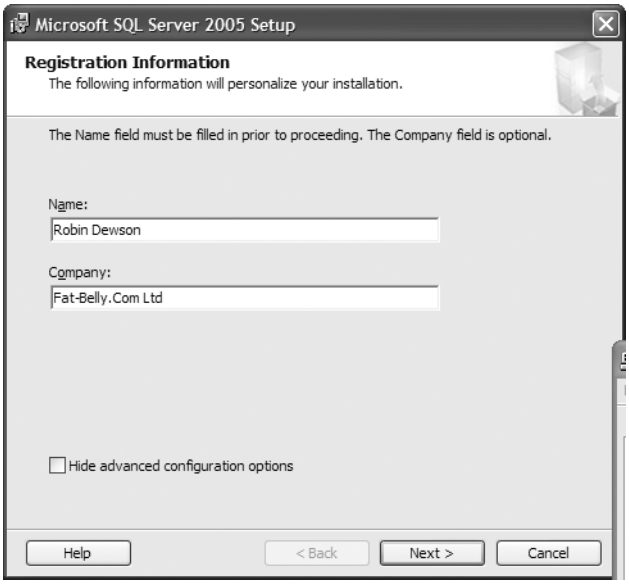

Figure 1-3. *Registration page*

Choosing the Components to Install

We now come to the Components to Install screen, where we have to make some decisions. As you see in Figure 1-4, 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.

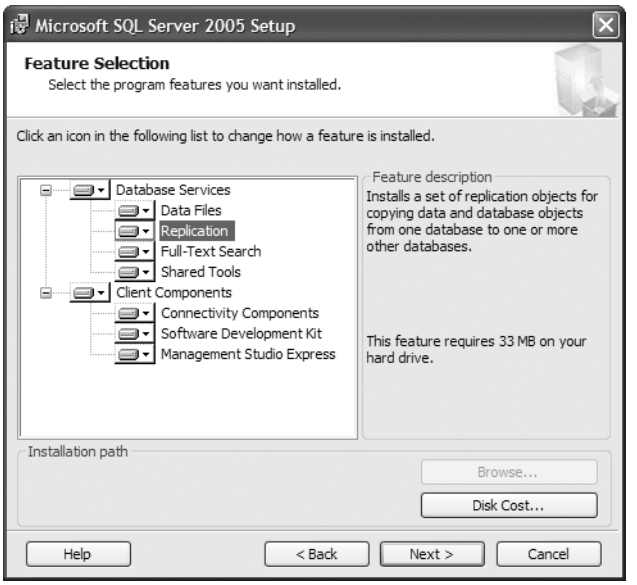

Figure 1-4. *Selecting every component to install*

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

- *Data Files*: This is the core of SSE, and it installs the main engine, data files, etc., to make SQL Server run.
- *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.
- *Shared Tools*: These are tools used by every instance of SQL Server Express.
- *Connectivity Components*: This component allows importing and exporting data using as the data source not only SQL Server, but also Oracle, Excel, and so on.
- *Software Development Kit*: These are tools for working on the workstation. This installs the GUI we use with SQL Server, and it can also install the help feature, Books Online.
- *Management Studio Express*: This is the graphical tool used to work with SSE.

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, etc., 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.

Default Instance 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 this, as it gives you an installation with no name and therefore no hint as to its use. As you are still learning, the easiest option to understand is to use the Default Instance, so select Default Instance as shown in Figure 1-5 and then click Next.

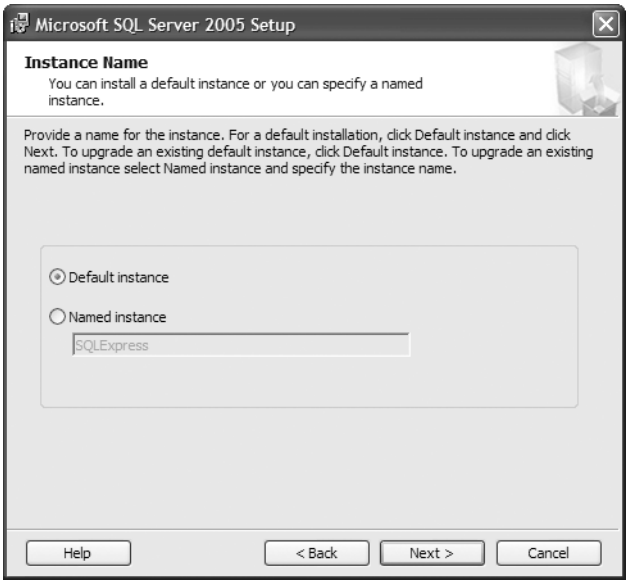

Figure 1-5. *Naming the install instance*

Service Accounts

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, etc., 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 also use 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 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 successful 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. As 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 Built-in System Account/Local System as you see in Figure 1-6. You can also define what services start when the computer is started up. Keep the defaults at the moment as you can always change these later via the Services icon within the Control Panel. Click Next.

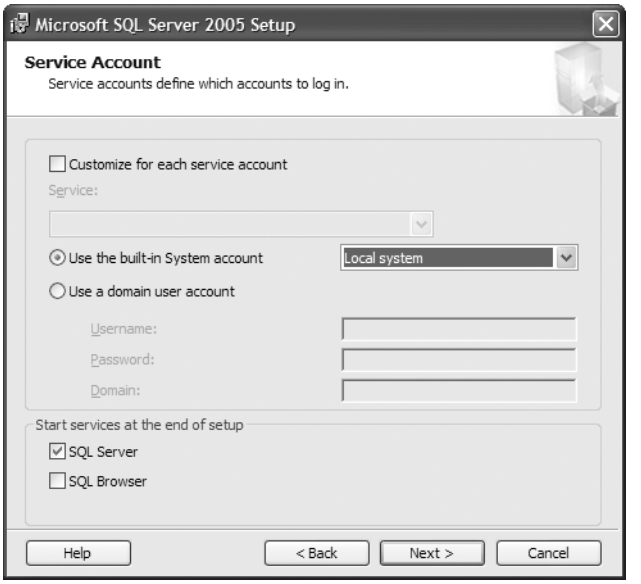

Figure 1-6. *Service account selection*

Authentication Mode

We now come to how we want to enforce security on our SSE installation. As Figure 1-7 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. We 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.

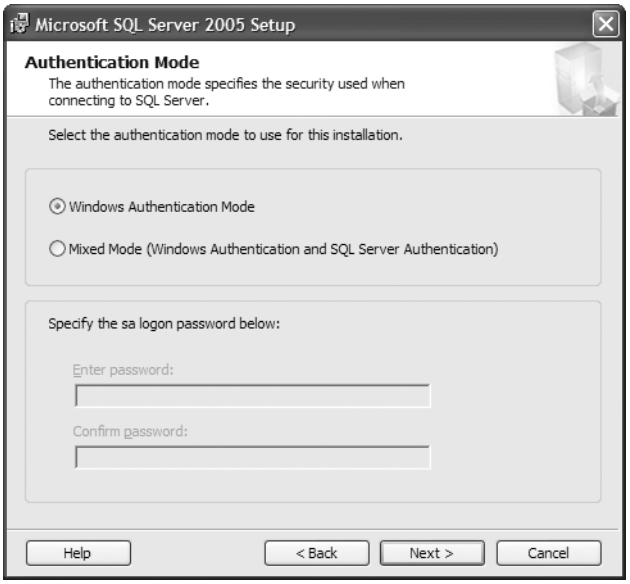

Figure 1-7. *Authentication choices*

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 only be in exceptional circumstances that you would do this, as it will cause extra processing complications when using the same processing in more than one service. Figure 1-8 shows that Windows collation has been chosen.

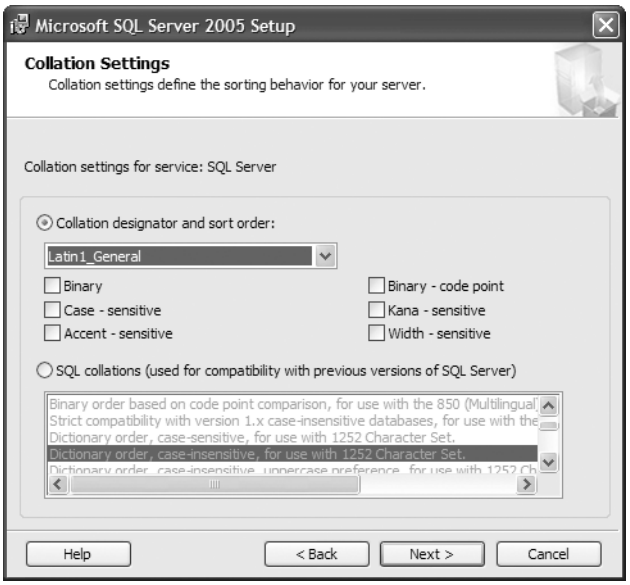

Figure 1-8. *Choosing the collations*

The Reporting Services Database

If you were on an operating system that had IIS, you may have selected Reporting Services to be installed. It is necessary to create a database for the reporting server to use. Depending on your requirements and how heavily used your SQL Server installation is, you may wish your reports to be run out of a separate and purpose-built SQL Server installation. For the moment, we will install Reporting Services on the same SQL Server (see Figure 1-9).

Figure 1-9. *Installing as default Reporting Services*

User Instances

When installing SQL Server Express, we can give the install, also known as an instance, a name. In this book, we are using SQL Server Express with no name, and therefore we are using the Default Instance. If we want to have one instance for development, one for testing, and one for user acceptance testing, we would need to run the SSE setup process again. Then at the instance screen shown previously in Figure 1-5, we would name an instance at that point. However, SSE also allows for databases that have been created to be attached to the server when an SQL client connection is made, if the Enable User Instances option is enabled, as shown in Figure 1-10. What this allows is for developers to attach a database through their code in, for example, C# at run time.

This option allows databases to exist in SQL Server only at run time, which would allow SSE to run a large number of databases. However, if you do go down this route, you'll encounter problems involving authority and the connecting user, whether they have enough permissions to attach the database and what they can execute. Unless you have a real need to attach and detach at run time, leave this option disabled.

Figure 1-10. *Enabling user instances*

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 1-11 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.

The final screen (see Figure 1-12) is displayed when the setup is complete. You can click the Summary Log link to check the install log. There is also a list of recommendations and information in a scrollable text box at the bottom of the screen. In between these two areas is a link to a new tool for SSE, the Surface Area Configuration tool. This tool, which we look at in Chapter 2, deals with enabling or disabling features, services, etc. You don't have to click the link now, as we will access this tool from the Start menu later.

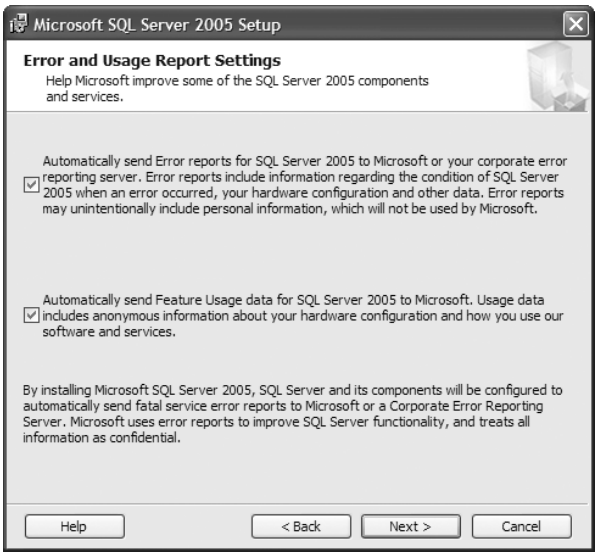

Figure 1-11. *Error and usage report settings*

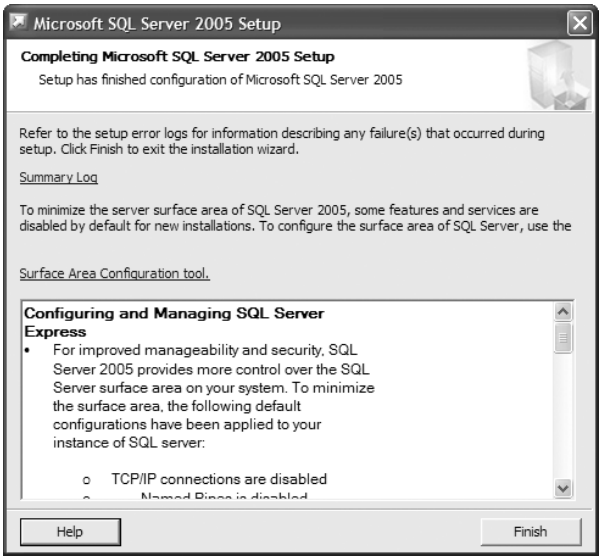

Figure 1-12. *Setup complete details*

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

Security

To discuss the Service Account dialog box that we came across in the installation properly (refer back to Figure 1-6), we need to delve into the area of Windows security.

In this section, we will first examine the concept of Windows services as opposed to programs, and then move on to discussing different types of authentication we 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, is either loaded in memory and running, or 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 a user to be logged in to Windows in order for him or her 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. Providing 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, as 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 was running as a program, some sort of decision has 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.

It's time to move on to the options we 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 that has to be made, concerns the authentication mode you wish 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 2000/2003/XP machine, a username must be supplied. There is no way around this (unlike in Windows 9*x*/ME where a username is 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 us 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 we have logged in to Windows, SQL Server uses a trusted connection. This means SQL Server is trusting that the username and password have been validated as we just mentioned. If, however, the username does not exist, you won't be able to log on to that machine. Someone else can log on to your machine with his or her user ID and password, and although he or she might be able to get to SQL Server by finding the executable on the C drive, SQL Server will first of all check to see whether that user has a valid login within SQL Server. If the login isn't valid, it 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, the user may well be able to at least connect to SQL Server.

SSE on Windows XP Home works slightly differently with its security compared to Windows XP Pro, 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.

We are also in a bit of a catch-22 situation here. You need to know about security for your install process, but to demonstrate it fully means working with SQL Server Management Studio Express, which the next chapter covers. We 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.
2. Select Start ➤ Control Panel ➤ User Accounts.
3. When the Users and Passwords dialog box comes up, click Create a New Account.
4. Once the Name the New Account dialog box comes up, enter the username *AJMason*, as shown in Figure 1-13. When done, click Next.
5. Ensure that the account type specified is Limited (see Figure 1-14). This means that it will not have administrator privileges.

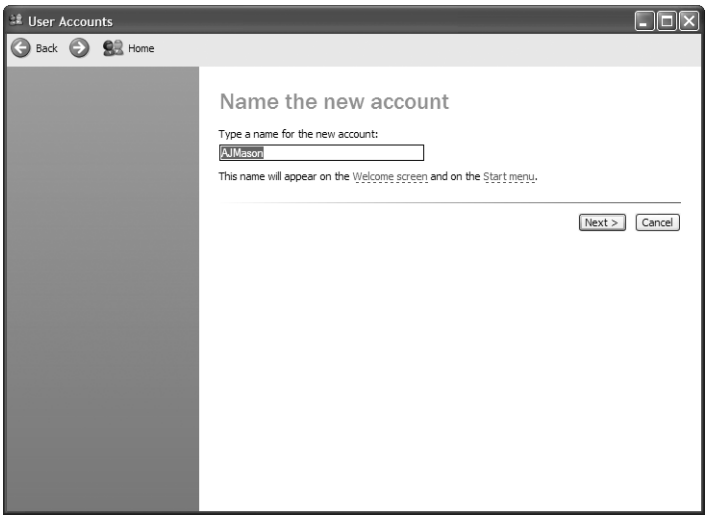

Figure 1-13. *Creating a new user account*

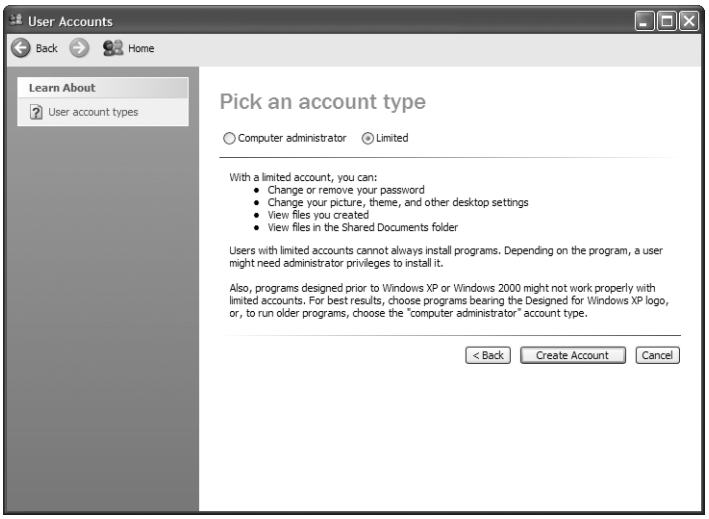

Figure 1-14. *Selecting the new user's account type*

6. Stay in the User Accounts dialog box, as you want to add a second username. Repeat the preceding process using the following details:

   Username: VMcGlynn

   Account type: Computer Administrator

7. Log off from Windows and then log on using the first ID that you created: AJMason.

8. Once logged on, start up SQL Server Management Studio Express by selecting Start ➤ All Programs ➤ Microsoft SQL Server 2005 ➤ SQL Server Management Studio Express. This is where Windows XP Home differs from other operating systems. The remaining steps will only be seen on Windows XP Professional.

9. Examine the error message that appears, which should resemble what you see in Figure 1-15. AJMason as a login has not been defined within SQL Server specifically and does not belong to a group that allows access. The only group at the minute is a user who is in the `Administrators` Windows group. Recall that AJMason is a Limited user.

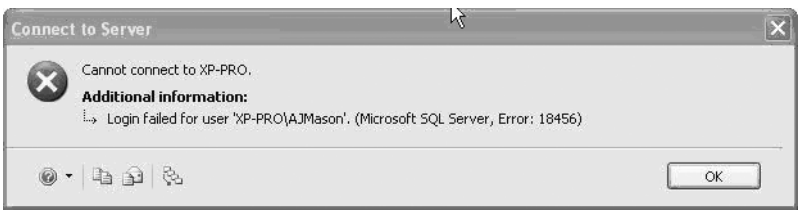

Figure 1-15. *Failed login to server*

10. We will now try out the other user we created. Close down SQL Server, log off Windows, and log on using the second ID—VMcGlynn. Once logged in, start up SQL Server Management Studio Express and connect to your server. This time the login will work.

We have created two usernames: one that has restricted access (AJMason), the other with administration rights (VMcGlynn). However, neither of these specific usernames exist within SQL Server itself: after all, we haven't entered them and they haven't appeared as if by magic. So why did one succeed and one fail?

The Windows security model has ensured that both IDs are valid. If the ID or password were incorrect, there is no way that 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 1-16, neither AJMason nor VMcGlynn exist.

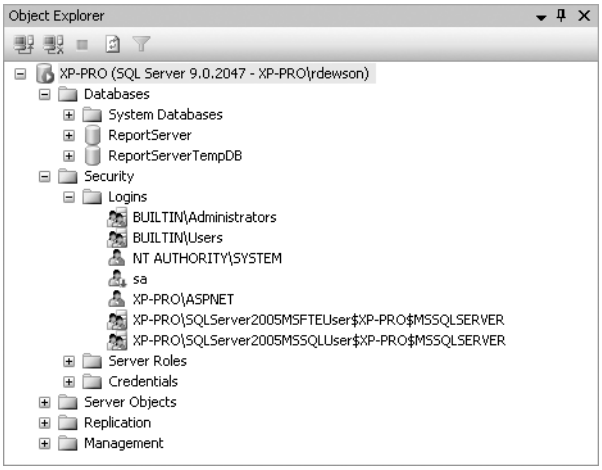

Figure 1-16. *Object Explorer for SQL Server*

However, you can see that there is a Windows group called `BUILTIN\Administrators`. This means that any username that is part of the `Administrators` group will have the capacity to log on to this SQL Server. Hence avoid if possible setting up users as administrators of their own PCs.

In a production environment, it may be advisable to remove this group from the system if you do allow users to be administrators. As VMcGlynn is a member of the `Administrators` group, then this username will also be a member of the `BUILTIN\Administrators` group.

Windows XP Home

Windows XP Home will place both AJMason and VMcGlynn in the same group, `BuiltIn\Users`. This group exists within SQL Server, and therefore all users you ever define for your computer will be able to connect. They will not be able to do anything on any database created that they have not been given access to, but you should instantly realize that the security is slightly more lax. The main reason behind this is the lesser complexity of Windows Security in Windows XP Home compared to versions of the software that are more likely to be running on networks and so on. I discuss this more in Chapter 4.

Mixed Mode

If we installed SQL Server with mixed mode, this means we 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 Windows taking the ID 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.

There is also another argument for mixed mode. We may wish for some reason (for example, for auditing purposes) that the user log on to SQL Server using a different username from that of his or her Windows account. We could be working on a large SQL Server development project that will have developers joining and leaving the team as the need arises. In this case, it might be necessary to create temporary usernames, as opposed to permanent IDs linked to the developers' Windows usernames. In SQL Server, we could create usernames of Developer1, Developer2, etc. These usernames can have different access rights within SQL Server. Another situation would be the case of an Internet-based application where there is just no way we could create a username for every visitor to our site. Therefore, we would create a generic login ID using a specific ID created for the web site. Whatever the reason, there is a need to have usernames not linked with the Windows usernames.

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. You saw during the installation process that you would be forced to include a password for this account if you were installing with SQL Server Authentication enabled because it is such a powerful login that exists in every SQL Server installation. 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. He or she 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 2005 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 than 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. We need to contact that person to check whether he or she can stop the process. If that person logs in as sa, we 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 we can track. By

restricting the sa login so that people have to use their own accounts, we can ensure a much higher degree of system monitoring and integrity.

There will be times that we want mixed mode authentication; it is perfectly acceptable to wish this. Internet providers use mixed mode, as 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 we do? Well, we create a logon ID that will have the access privileges we wish; in other words, the ability to just see the data and work with the data that we 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 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 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 2005 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.