Beginning SQL Server 2008 for Developers: From Novice to Professional

Copyright © 2008 by Robin Dewson

ISBN-13 (pbk): 978-1-59059-958-7

ISBN-10 (pbk): 1-59059-958-6

ISBN-13 (electronic): 978-1-4302-0584-5

ISBN-10 (electronic): 1-4302-0584-9

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

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`.

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

CHAPTER 1

SQL Server 2008 Overview and Installation

Welcome to *Beginning SQL Server 2008 for Developers*. This book has been written for those who are interested in learning how to create solutions with Microsoft SQL Server 2008, but have no prior knowledge of SQL Server 2008. You may well have had exposure to other databases, such as MySQL, Oracle, or Microsoft Access, but SQL Server uses different interfaces and has a different way of working compared to much of the competition. The aim of this book is to bring you quickly up to a level at which you are developing competently with SQL Server 2008. This book is specifically dedicated to beginners and to those who at this stage wish to use only SQL Server 2008. It is also for those developers who have SQL Server 2005 experience and want a quick method to get up to speed on SQL Server 2008. You may find this book useful for understanding the basics of other databases in the marketplace, especially when working with T-SQL. Many databases use an ANSI-standard SQL, so moving from SQL Server to Oracle, Sybase, etc., after reading this book will be a great deal easier.

This chapter covers the following topics:

- Why SQL Server 2008?
- 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 SQL Server 2008?

We will also then look at installing our chosen edition and cover the following:

- Installing SQL Server 2008 on a Windows XP platform
- Options not installed by default
- Where to install SQL Server physically
- Multiple installations on one computer
- How SQL Server runs on a machine
- How security is implemented
- Logon IDs for SQL Server, especially the `sa` (system administrator) logon

Why SQL Server 2008?

The following discussion is my point of view, and although it no doubt differs from that of others, the basis of the discussion holds true. SQL Server faces competition from other databases, not only from other Microsoft products such as Microsoft Access and Microsoft Visual FoxPro, but also from competitors such as Oracle, Sybase, DB2, and Informix, to name a few.

Microsoft Access is found on a large number of PCs. The fact that it is packaged with some editions of Office and has been around for a number of years in different versions of Office has helped make this database ubiquitous; in fact, a great number of people actually do use the software. Unfortunately, it does have its limitations when it comes to scalability, speed, and flexibility, but for many small, in-house systems, these areas of concern are not an issue, as such systems do not require major database functionality.

Now we come to the serious competition: Oracle and Sybase. Oracle is seen as perhaps the market leader in the database community, and it has an extremely large user base. There is no denying it is a great product to work with, if somewhat more complex to install and administer than SQL Server; it fits well with large companies that require large solutions. There are many parts to Oracle, which make it a powerful tool, including scalability and performance. It also provides flexibility in that you can add on tools as you need them, making Oracle more accommodating in that area than SQL Server. For example, SQL Server 2008 forces you to install the .NET Framework on your server whether you use the new .NET functionality or not. However, Oracle isn't as user friendly from a developer's point of view in areas like its ad hoc SQL Query tool and its XML and web technology tools, as well as in how you build up a complete database solution; other drawbacks include its cost and the complexity involved in installing and running it effectively. However, you will find that it is used extensively by web search engines, although SQL Server could work just as effectively. With the new functionality in SQL Server 2008, Oracle will be under pressure to expand its existing functionality to meet this challenge. SQL Server has always been a one-purchase solution, such that (providing you buy the correct version) tools that allow you to analyze your data or copy data from one data source such as Excel into SQL Server will all be "in the box." With Oracle, on the other hand, for every additional feature you want, you have to purchase more options.

Then there is Sybase. Yes, it is very much like SQL Server with one major exception: it has no GUI front end. Sybase iAnywhere, which is mainly used for small installations, does have a front end, but the top-of-the-range Sybase does not. To purists, there is no need for one, as GUI front ends are for those who don't know how to code in the first place—well, that's their argument, of course, but why use 60+ keystrokes when a point, click, and drag is all that is required?

Sybase is also mainly found on Unix, although there is a Windows version around. You can get to Sybase on a Unix machine via a Windows machine using tools to connect to it, but you still need to use code purely to build your database solution. It is very fast and very robust, and it is only rebooted about once, maybe twice, a year. Another thing about Sybase is that it isn't as command- and feature-rich as SQL Server. SQL Server has a more powerful programming language and functionality that is more powerful than Sybase.

Each database has its own SQL syntax, although they all will have the same basic SQL syntax, known as the ANSI-92 standard. This means that the syntax for retrieving data, and so on, is the same from one database to another. However, each database has its own special syntax to maintain it, and trying to use a feature from this SQL syntax in one database may not work, or may work differently, in another.

So SQL Server seems to be the best choice in the database marketplace, and in many scenarios it is. It can be small enough for a handful of users, or large enough for the largest corporations. It doesn't need to cost as much as Oracle or Sybase, but it does have the ability to scale up and deal with terabytes of data without many concerns. 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.

Now that you know the reasons behind choosing SQL Server, you need to know which versions of SQL Server are out there to purchase, what market each version is aimed at, and which version will be best for you, including which version can run on your machine.

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.
1994		Microsoft splits from Sybase.
1995	SQL Server 6.05, a small business database	Major rewrite of the core database engine. First "significant" release. 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.
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%.[a]

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

Year	Version	Description
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 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 had with its inclusion of Java.
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 and the use of Language Integrated Query (LINQ). It also deals with data, such as XML, compact devices, and massive database installations, that reside in many different places. Also, it offers the ability to set rules within a framework to ensure databases and objects meet defined criteria, and it offers the ability to report when these objects do not meet this criteria.

[a] *Gartner Report, May 21, 2003*

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 my chosen SQL Server edition on? Will this help me refine my decision?"

Judging by today's standards of minimum-specification hardware that can be bought—even the low-cost solutions—the answer will in most cases be "Yes" to most editions. However, you may have older hardware (things move so fast that even hardware bought a couple of months ago can quickly 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 1GHz processor for the 32-bit edition and a 1.6GHz for the 64-bit version, or a compatible processor, or similar processing power; however, 2GHz is recommended. However, as with most minimums listed here, Microsoft wholly recommends a faster processor. 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 2GHz or above, but the faster the processer the better. You will find your development time reduced for it.

However, it is not processor alone that speeds up SQL Server. A large part is 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. SQL Server requires a minimum of 512MB of RAM onboard your computer, although you shouldn't have too many more applications open and running, as they could easily not leave enough memory for SQL Server to run fast enough. Microsoft recommends 1GB or above, and really double that at least for when you start using your SQL Server in earnest.

If you wanted to run the Enterprise Edition, then a minimum, and I mean a bare minimum, of 1GB really should be installed, especially if you want to use any of the more advanced features.

The more memory the better: I really would recommend a minimum of 1GB on any computer that a developer is using, with 2GB ideal and sufficient to give good all-around performance. If a process can be held in memory, rather than swapped out to hard drive or another area while you are running another process, then you are not waiting on SQL Server being loaded back into memory to start off where it left off. This is called **swapping**, and the more memory, the less swapping could, and should, take place.

Taking CPU speed and memory together as a whole, it is these two items that are crucial to the speed that the computer will run, and having sufficient speed will let you develop as fast as possible.

When it comes to installing SQL Server, insufficient memory won't stop the install, but you will be warned that you need more.

Hard Disk Space

You will need lots! But name a major application these days that doesn't need lots! For SQL Server alone, ignoring any data files that you are then going to add on top, you will need over 1GB of space. Certainly, the installation options that will be used later in the chapter will mean you need this amount of space. You can reduce this by opting not to install certain options—for example, Books Online; however, even most notebooks these days come with a minimum 40GB, and 80GB is not uncommon either. 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 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 fully, together we will develop a system for a financial company that will have features such as banking, share purchase, 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, as I don't want to detract from SQL Server. 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 SQL Server.

Installation

The remainder of this chapter will guide you through the installation process of the Developer Edition, although virtually all that you see will be in every edition. Some of the differences will be due to the functionality of each edition. Microsoft offers a 120-day trial version at `http://www.microsoft.com/sql/evaluation/trial/`, which you can use to follow along with the examples in this book if you don't already have SQL Server 2008.

This book will cover many of the options and combinations of features that can be included within an installation. A number of different tools are supplied with SQL Server to be included with the installation. We will look at these tools so that a basic understanding of what they are will allow us to decide which to install.

Installation covers a great many different areas:

- Security issues
- Different types of installation—whether this is the first installation and instance of SQL Server 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.

This book uses the Developer Edition because it is most suited to our needs, as developers, for it doesn't have all the operating system requirements of the Enterprise Edition. Insert the CD for the Microsoft SQL Server 2008 edition of your choice in your CD-ROM drive and start the installation. What the upcoming text covers is a standard installation.

Beginning the 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.

If you are using a CD-ROM and the installation process does not automatically start, open up Windows Explorer and double-click autorun.exe, found at the root level of the CD-ROM. If you are not using a CD-ROM, double-click the installer executable that you downloaded.

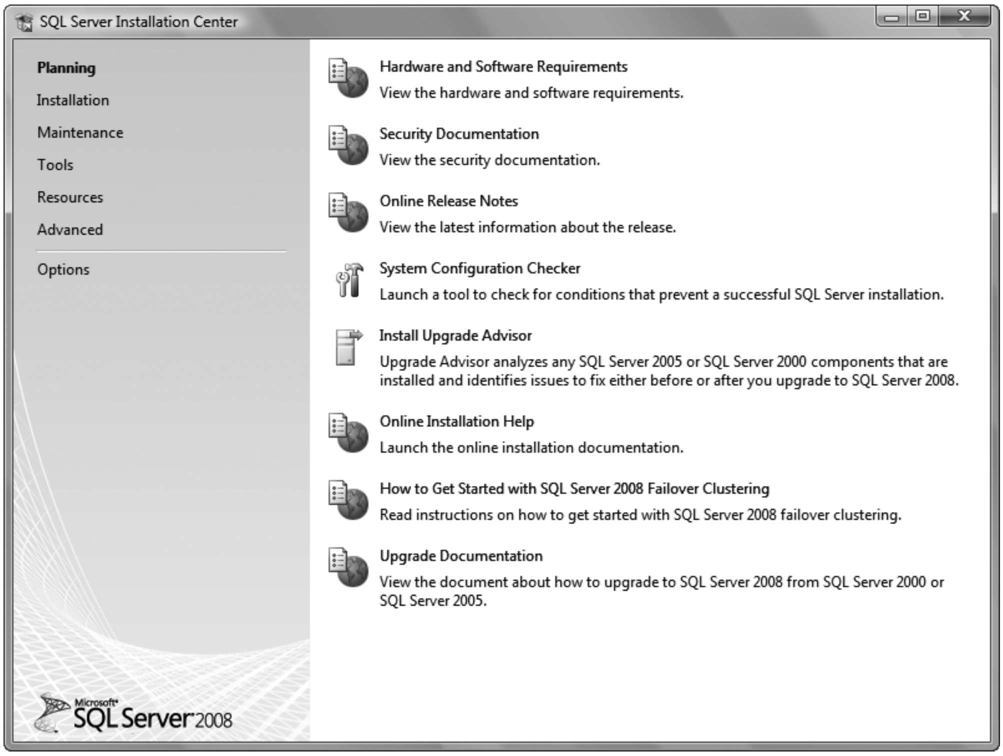
You may now be presented with the installation screen for Microsoft .NET 3.5 Framework if it is not already installed. .NET is a framework that Microsoft created that allows programs written in VB .NET, C#, and other programming languages to have a common compile set for computers. SQL Server 2008 uses .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. With SQL Server 2008, there is also the ability to query the database using .NET and LINQ rather than T-SQL.

■ **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).

Once this is installed, you are presented with the SQL Server Installation Center. This screen, shown in Figure 1-1,deals with planning an installation, setup processes, including new installations, upgrades from previous versions of SQL Server, and many other options for maintaining SQL Server installations.

When you click on the Installation item on the left of the Installation Center, you can then select from the first item of the list of Installation options, New SQL Server stand-alone installation or add features to an existing installation, and the SQL Server 2008 installation starts.

A quick system check is performed before you enter your product key and accept the license terms of SQL Server. There are a number of support files that SQL Server uses as part of the installation process, as well as to ensure that a clean and valid installation occurs. In Figure 1-2, you will see that there is one warning in the process, but it is still possible to proceed. Provided no errors are listed in your process, click Next.



Hardware and Software Requirements
View the hardware and software requirements.

Security Documentation
View the security documentation.

Online Release Notes
View the latest information about the release.

System Configuration Checker
Launch a tool to check for conditions that prevent a successful SQL Server installation.

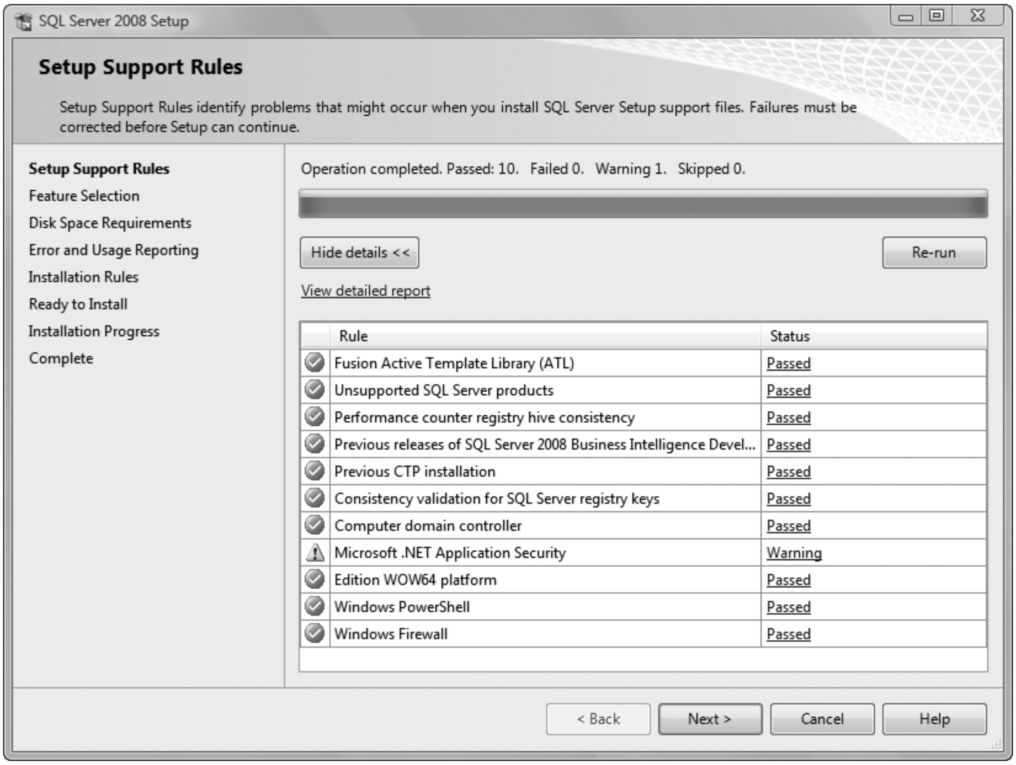
Install Upgrade Advisor
Upgrade Advisor analyzes any SQL Server 2005 or SQL Server 2000 components that are installed and identifies issues to fix either before or after you upgrade to SQL Server 2008.

Online Installation Help
Launch the online installation documentation.

How to Get Started with SQL Server 2008 Failover Clustering
Read instructions on how to get started with SQL Server 2008 failover clustering.

Upgrade Documentation
View the document about how to upgrade to SQL Server 2008 from SQL Server 2000 or SQL Server 2005.

Figure 1-1. *Beginning the install with the Installation Center*



Setup Support Rules

Setup Support Rules identify problems that might occur when you install SQL Server Setup support files. Failures must be corrected before Setup can continue.

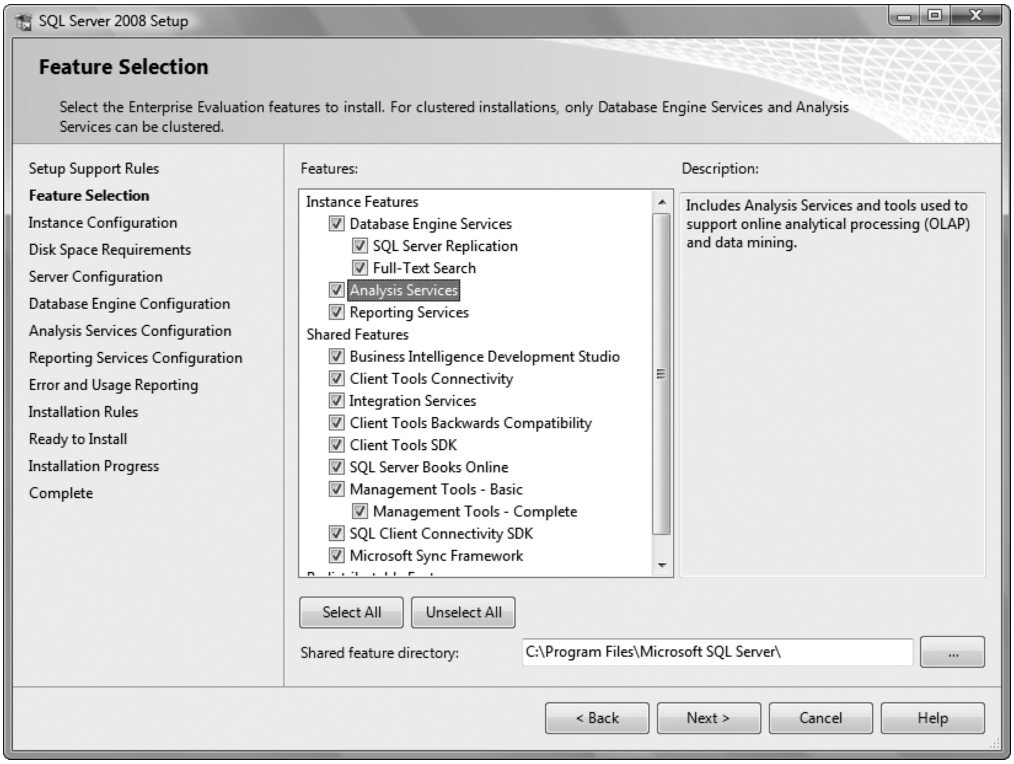
Operation completed. Passed: 10. Failed 0. Warning 1. Skipped 0.

	Rule	Status
✓	Fusion Active Template Library (ATL)	Passed
✓	Unsupported SQL Server products	Passed
✓	Performance counter registry hive consistency	Passed
✓	Previous releases of SQL Server 2008 Business Intelligence Devel...	Passed
✓	Previous CTP installation	Passed
✓	Consistency validation for SQL Server registry keys	Passed
✓	Computer domain controller	Passed
⚠	Microsoft .NET Application Security	Warning
✓	Edition WOW64 platform	Passed
✓	Windows PowerShell	Passed
✓	Windows Firewall	Passed

Figure 1-2. *System configuration checks*

Choosing the Features to Install

We now come to the Feature Selection screen, where we have to make some decisions. As you see in Figure 1-3, this installation will have everything installed, because this will be your development instance where you'll 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. However, you can be selective regarding what parts of the components you want to install. For this book, you will need Database Engine Services, Reporting Services, Client Tools, and Business Intelligence Development Studio for building reports, so ensure at least that these are checked.



Feature Selection

Select the Enterprise Evaluation features to install. For clustered installations, only Database Engine Services and Analysis Services can be clustered.

Features:

Instance Features
- [✓] Database Engine Services
  - [✓] SQL Server Replication
  - [✓] Full-Text Search
- [✓] Analysis Services
- [✓] Reporting Services

Shared Features
- [✓] Business Intelligence Development Studio
- [✓] Client Tools Connectivity
- [✓] Integration Services
- [✓] Client Tools Backwards Compatibility
- [✓] Client Tools SDK
- [✓] SQL Server Books Online
- [✓] Management Tools - Basic
  - [✓] Management Tools - Complete
- [✓] SQL Client Connectivity SDK
- [✓] Microsoft Sync Framework

Description:

Includes Analysis Services and tools used to support online analytical processing (OLAP) and data mining.

Shared feature directory: C:\Program Files\Microsoft SQL Server\

Figure 1-3. *Selecting every component to install*

Let's briefly take a look at most of these components:

- *Database Engine Services*: This is the main core for SQL Server 2008 and installs the main engine, data files, etc., to make SQL Server run.
  - *SQL Server Replication:* When you want to send data changes not only on the database it is being executed on, but also on a similar database that has been built to duplicate those changes, then you can use this option to replicate the changes to that database.
  - *Full text search:* This option lets you allow searching of the text within your database.
- *Analysis Services*: Using this tool, you can take a set of data and dice and slice and analyze the information contained.

- *Reporting Services*: This allows reports to be produced from SQL Server instead of using third-party tools such as Crystal Reports. We look at this component in more detail in Chapter 14.
- *Client Tools:* Some of these tools sit on the client machine and provide the graphical interface to SQL Server, while others sit on the client and work with SQL Server. This is the option you would package up for rollout to developers to sit on their machines.
- *Microsoft Sync Framework:* When working with offline applications such as those on mobile devices, there has to be some sort of synchronization mechanism in place. This option allows these interactions to occur.
- *SQL Server Books online:* This is the help system. If you need more information, clarification, or extra detail on any area of SQL Server, then this is the area to turn to.
- *Business Intelligence Development Studio:* When you want to analyze data using analysis-based services, then you can use this GUI to interact with your database. This book doesn't cover this option.
- *Integration Services:* This final option allows you to build processes to complete actions, such as importing data from other data sources and manipulating the data. You will see Integration Services in action in Chapter 7 when we look at building a backup maintenance plan.

Of these components, Analysis Services and Business Intelligence Development Studio fall outside the scope of this book, and we only touch upon Integration Services as mentioned.

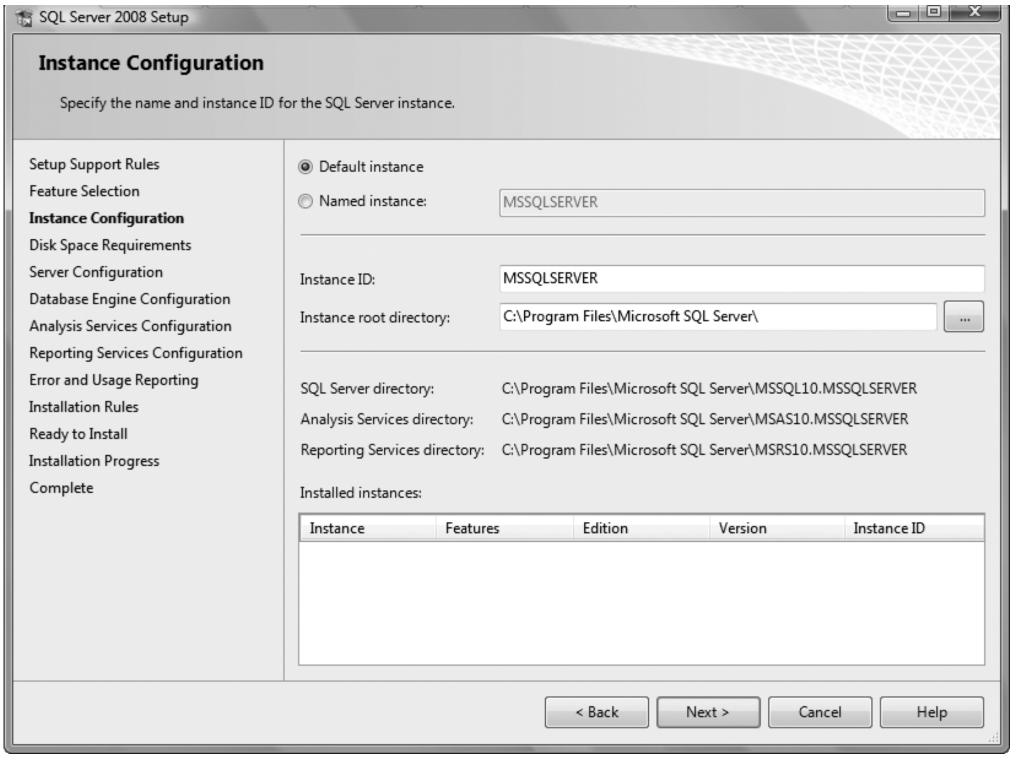
■**Note** At this point, SQL Server no longer has the option to install the sample databases. Microsoft is also altering the way sample databases and samples are delivered, so you may find newer versions on the SQL Server web site, `http://www.microsoft.com/sql`, or at `http://www.codeplex.com/SqlServerSamples`.

Naming the Instance

As you know, SQL Server 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 and processor to cope with two or three different applications running. These different applications want to have their own SQL Server databases. 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 don't, and if you have a rogue development action that occurs and crashes the server, you will stop production from running. Although you would have to make this decision at the start of the install process you are currently going through, it is a useful second reminder here when naming the instance.

The default instance is available, which is what is selected when you are not giving the install a specific name; once you install SQL Server 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 1-4 and then click Next.Once you have instances installed on the server, they will be listed here. You can also see the path detailed for each of the directories for the three services selected in the previous step.


SQL Server 2008 Setup

Instance Configuration

Specify the name and instance ID for the SQL Server instance.

[✓ Default instance]

Named instance: MSSQLSERVER

Instance ID: MSSQLSERVER

Instance root directory: C:\Program Files\Microsoft SQL Server\

SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER

Analysis Services directory: C:\Program Files\Microsoft SQL Server\MSAS10.MSSQLSERVER

Reporting Services directory: C:\Program Files\Microsoft SQL Server\MSRS10.MSSQLSERVER

Installed instances:

Instance	Features	Edition	Version	Instance ID

Figure 1-4. *Naming the install instance*

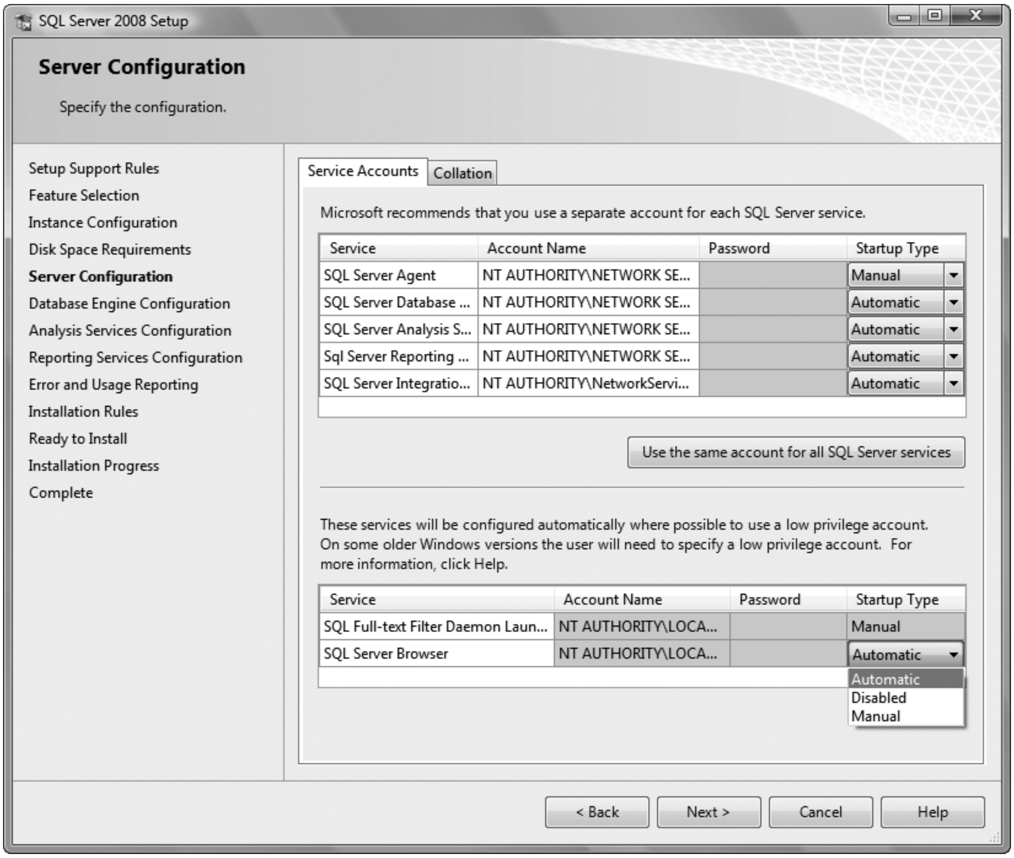
Choosing Service Accounts

SQL Server and other services, as defined in the Feature Selection screen (shown earlier in Figure 1-3), require you to log on to Windows before starting, just as you need to log on 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 that 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.

We will look at these options in more detail toward the end of the chapter. The options you see in Figure 1-5 will install SQL Server with a low-level set of privileges.

You can always change these later via the Services icon within the Control Panel. However, it would be much better to use the SQL Server Configuration Manager, found in Configuration Tools. By using the Configuration Manager, the account will be added to the correct group and be given the right permissions. Click Next.

If you look at the SQL Server Browser, which is another name for SQL Server Management Studio, it will be disabled by default. Many SQL Server installations will be on servers, quite often on remote servers; therefore, there should be no need to have the SQL Server Browser running. Normally, you will connect to SQL Server from a client machine. However, I am making the assumption that this installation is not on a server and is on a local computer, so change this option to start automatically.


SQL Server 2008 Setup

Server Configuration

Specify the configuration.

Setup Support Rules
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Analysis Services Configuration
Reporting Services Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Service Accounts | Collation

Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Agent	NT AUTHORITY\NETWORK SE...		Manual
SQL Server Database ...	NT AUTHORITY\NETWORK SE...		Automatic
SQL Server Analysis S...	NT AUTHORITY\NETWORK SE...		Automatic
Sql Server Reporting ...	NT AUTHORITY\NETWORK SE...		Automatic
SQL Server Integratio...	NT AUTHORITY\NetworkServi...		Automatic

Use the same account for all SQL Server services

These services will be configured automatically where possible to use a low privilege account. On some older Windows versions the user will need to specify a low privilege account. For more information, click Help.

Service	Account Name	Password	Startup Type
SQL Full-text Filter Daemon Laun...	NT AUTHORITY\LOCA...		Manual
SQL Server Browser	NT AUTHORITY\LOCA...		Automatic

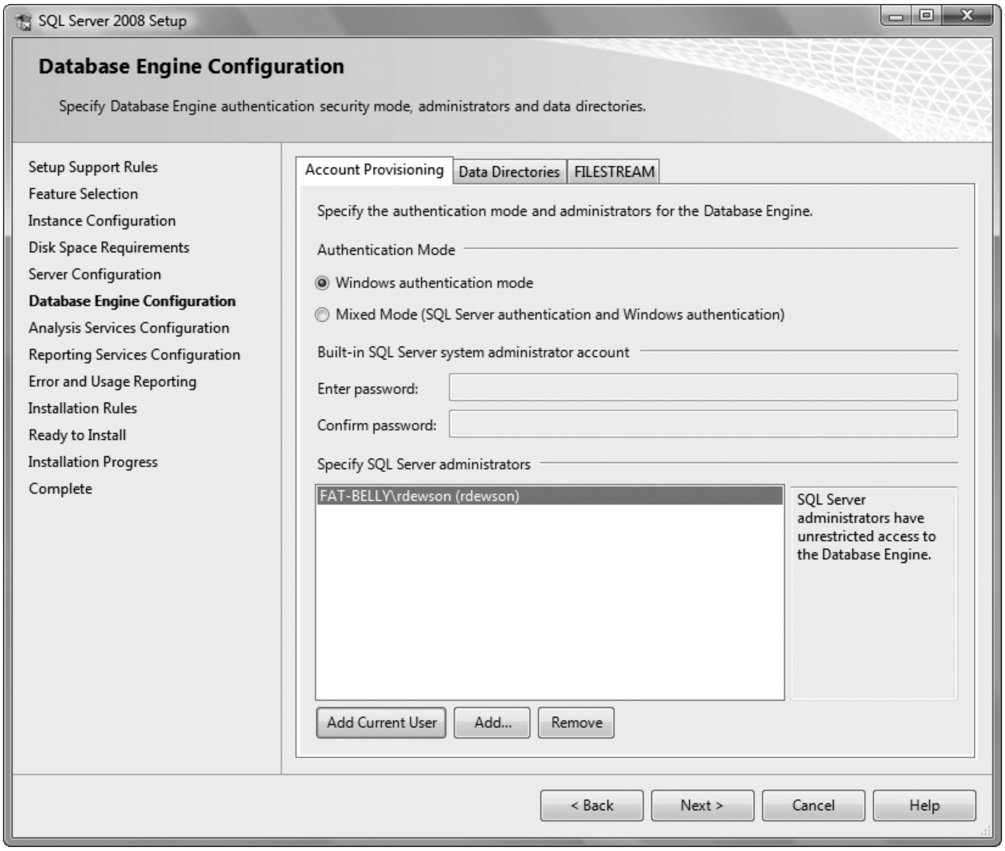
Figure 1-5. *Service account selection*

Selecting an Authentication Mode

We now come to how we want to enforce security on your SQL Server installation. As Figure 1-6 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 a SQL Server defined login ID and password. We also need to define a password for a special login ID, called sa, if you are working with mixed mode. 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, but not one that you will forget.

It is also necessary to define a SQL Server Administrator account. This is a special account that you can use to log on in 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 would be a server account ID, but for now, use the account you have used to log on to your computer.

You will also see a similar screen for Analysis Services, and the settings are the same.



Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

[✓ Windows authentication mode]

Mixed Mode (SQL Server authentication and Windows authentication)

Built-in SQL Server system administrator account

Enter password:

Confirm password:

Specify SQL Server administrators

FAT-BELLY\rdewson (rdewson)

SQL Server administrators have unrestricted access to the Database Engine.

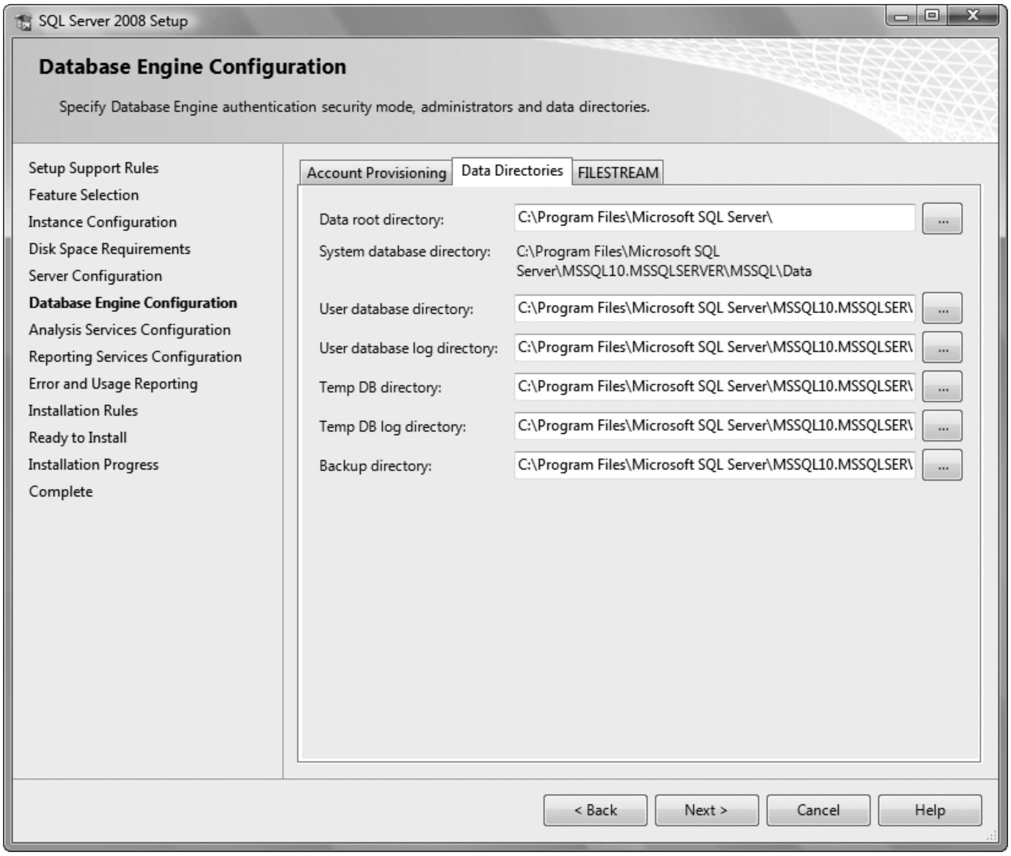
Figure 1-6. *Authentication choices including the SQL Server administrator account*

Defining the Data Directories

The Data Directories tab, as shown in Figure 1-7, 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 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, the installation is isolated. As we are only dealing with one instance within this book, leave these settings as they are.

The FILESTREAM tab is another type of data directory, but ignore this for now. We'll discuss this more later in the book. However, to give you a small insight, FILESTREAM is used when dealing with large amounts of unstructured data. In the past, this data was held totally separate from SQL Server, but now FILESTREAM allows the data to be managed by SQL Server, and this tab informs SQL Server of where it resides on the physical server.

You will see these same tabs in the next step if you selected Analysis Services.


SQL Server 2008 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators and data directories.

Data root directory: C:\Program Files\Microsoft SQL Server\

System database directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Data

User database directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSER\

User database log directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSER\

Temp DB directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSER\

Temp DB log directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSER\

Backup directory: C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSER\

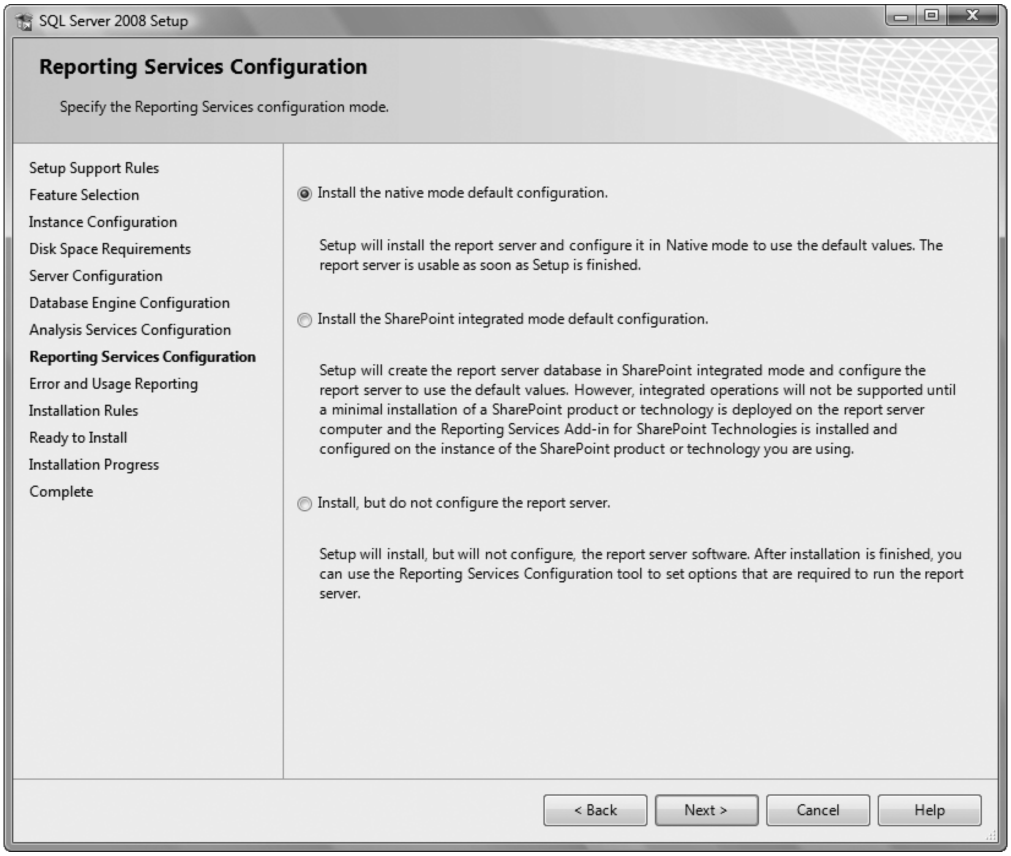
Figure 1-7. *Defining the locations of SQL Server data directories*

Creating the Reporting Services Database

As we selected Reporting Services to be installed, we 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 installed, you would then have to create a reporting database.

The Native mode configuration, as shown in Figure 1-8, is the simplest and the one we will be using. It installs Reporting Services and also creates the necessary databases within our SQL Server. It will only be available 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.

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.



[✓ Install the native mode default configuration.]

Setup will install the report server and configure it in Native mode to use the default values. The report server is usable as soon as Setup is finished.

Install the SharePoint integrated mode default configuration.

Setup will create the report server database in SharePoint integrated mode and configure the report server to use the default values. However, integrated operations will not be supported until a minimal installation of a SharePoint product or technology is deployed on the report server computer and the Reporting Services Add-in for SharePoint Technologies is installed and configured on the instance of the SharePoint product or technology you are using.

Install, but do not configure the report server.

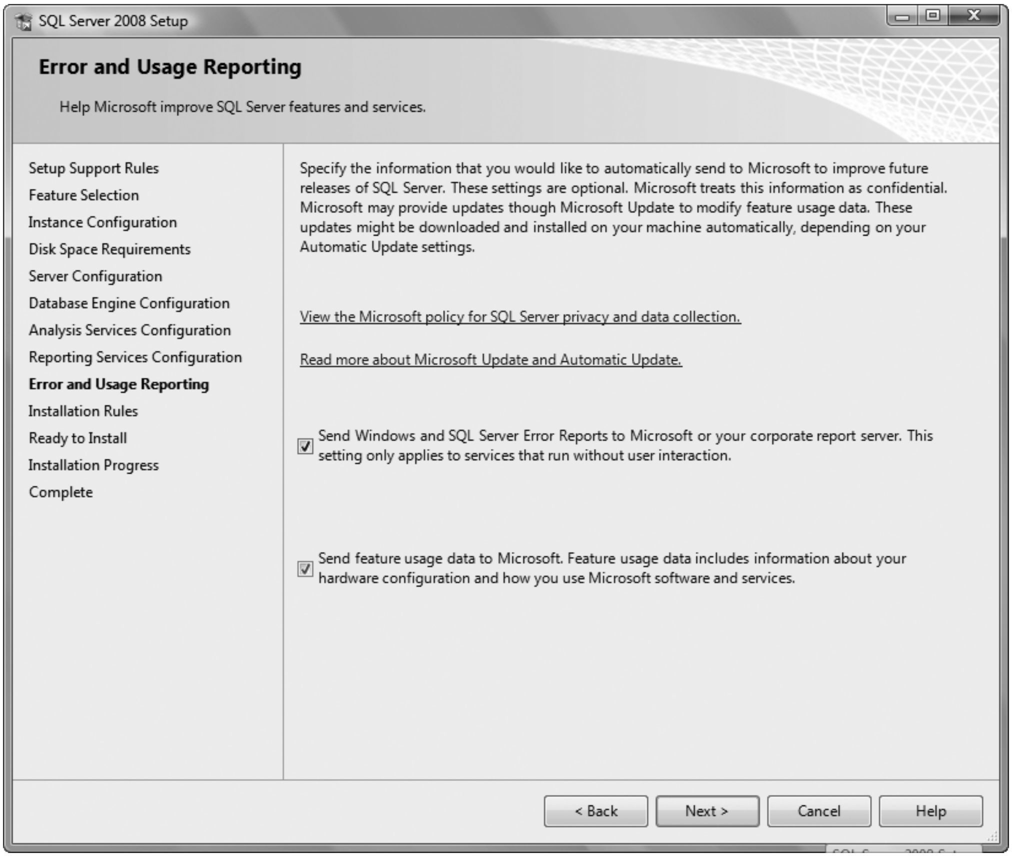
Setup will install, but will not configure, the report server software. After installation is finished, you can use the Reporting Services Configuration tool to set options that are required to run the report server.

Figure 1-8. *Installing native mode configuration for Reporting Services*

Configuring Error and Usage Reports

Within SQL Server, it is possible for any errors to be automatically reported and sent to Microsoft. These include fatal errors where SQL Server shuts down unexpectedly. It is recommended that you keep the error settings shown in Figure 1-9 enabled. No organizational information will be sent, so your data will still be secure. This is similar to sending reports when Excel crashes, for example. It is better to have this switched to active. Sending the errors to Microsoft will hopefully produce faster patch fixes and better releases in the future. It is also possible for SQL Server to take information about how you are using SQL Server. This is also a useful setting to have switched on, so that Microsoft can receive information that might help it improve the product. However, it would be useful to have this switched on in a production environment where it is more relevant.

When you click Next, you will be met with a screen detailing the installation rules. There is nothing to do; just click Next, where the final screen (see Figure 1-10) is displayed. The setup collection is complete, and you are ready to install.

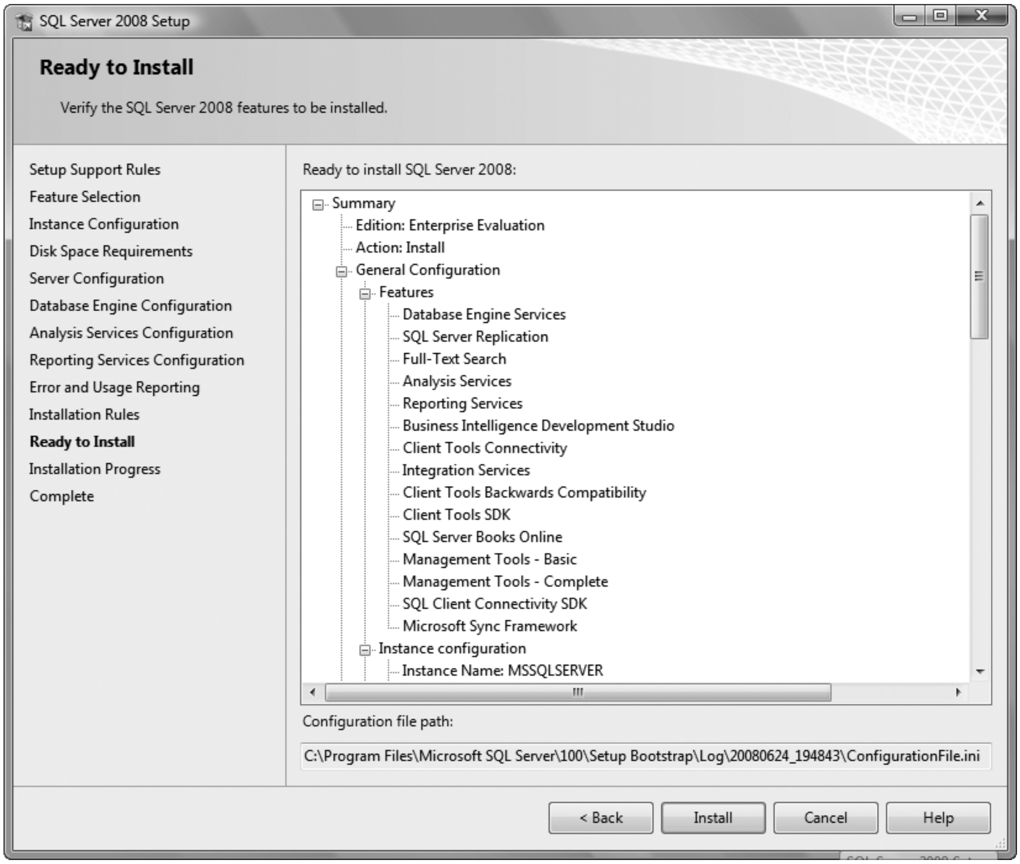

SQL Server 2008 Setup
Error and Usage Reporting
Help Microsoft improve SQL Server features and services.
Setup Support Rules
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Analysis Services Configuration
Reporting Services Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete
View the Microsoft policy for SQL Server privacy and data collection.
Read more about Microsoft Update and Automatic Update.
< Back
Next >
Cancel
Help

Specify the information that you would like to automatically send to Microsoft to improve future releases of SQL Server. These settings are optional. Microsoft treats this information as confidential. Microsoft may provide updates though Microsoft Update to modify feature usage data. These updates might be downloaded and installed on your machine automatically, depending on your Automatic Update settings.

[✓] Send Windows and SQL Server Error Reports to Microsoft or your corporate report server. This setting only applies to services that run without user interaction.

[✓] Send feature usage data to Microsoft. Feature usage data includes information about your hardware configuration and how you use Microsoft software and services.

Figure 1-9. *Error and Usage Reporting settings*


SQL Server 2008 Setup

Ready to Install

Verify the SQL Server 2008 features to be installed.

Setup Support Rules
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Analysis Services Configuration
Reporting Services Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Ready to install SQL Server 2008:

```
Summary
    Edition: Enterprise Evaluation
    Action: Install
    General Configuration
        Features
            Database Engine Services
            SQL Server Replication
            Full-Text Search
            Analysis Services
            Reporting Services
            Business Intelligence Development Studio
            Client Tools Connectivity
            Integration Services
            Client Tools Backwards Compatibility
            Client Tools SDK
            SQL Server Books Online
            Management Tools - Basic
            Management Tools - Complete
            SQL Client Connectivity SDK
            Microsoft Sync Framework
        Instance configuration
            Instance Name: MSSQLSERVER
```

Configuration file path:

C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\20080624_194843\ConfigurationFile.ini

Figure 1-10. *Complete setup details*

Security

To discuss the Service Account dialog box that we came across in the installation properly, 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

SQL Server runs as a Windows service. So what is a service? A good example of a service is any anti-virus 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 the services 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 in to the service but is a totally separate unit of work (for example, SQL Server Management Studio), 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, SQL Server 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. SQL Server will run quite happily and, with any luck, never give an error. But what if one day there is an error? If SQL Server is running as a program, you'll have to make some sort of decision. Even if SQL Server crashes, 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, SQL Server is under Windows control. If a problem occurs, whether with SQL Server, 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, because SQL Server will be running on a home or local system, then you can use this Windows user ID for SQL Server to also 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 SQL Server. This could exist for several reasons. For example, your Windows account should be set up so that the password expires after 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 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 successful 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 1-5, 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. This is quite a crucial option when moving to a corporate environment due to the security implications that you must deal with.

SQL Server has several different processes that exist for different work. SQL Server is used to run SQL Server itself, and SQL Server Agent runs processes such as batch jobs. SQL Server should only really need to access itself. Therefore, it should only require a domain login with very restricted privileges.

SQL Server Agent, which runs batch processes and complex tasks including working with other servers, needs a more powerful domain account. 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 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 you have to make, 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 2003/XP/Vista machine, a username must be supplied. There is no way around this (unlike in Windows 9x/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 Windows/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.

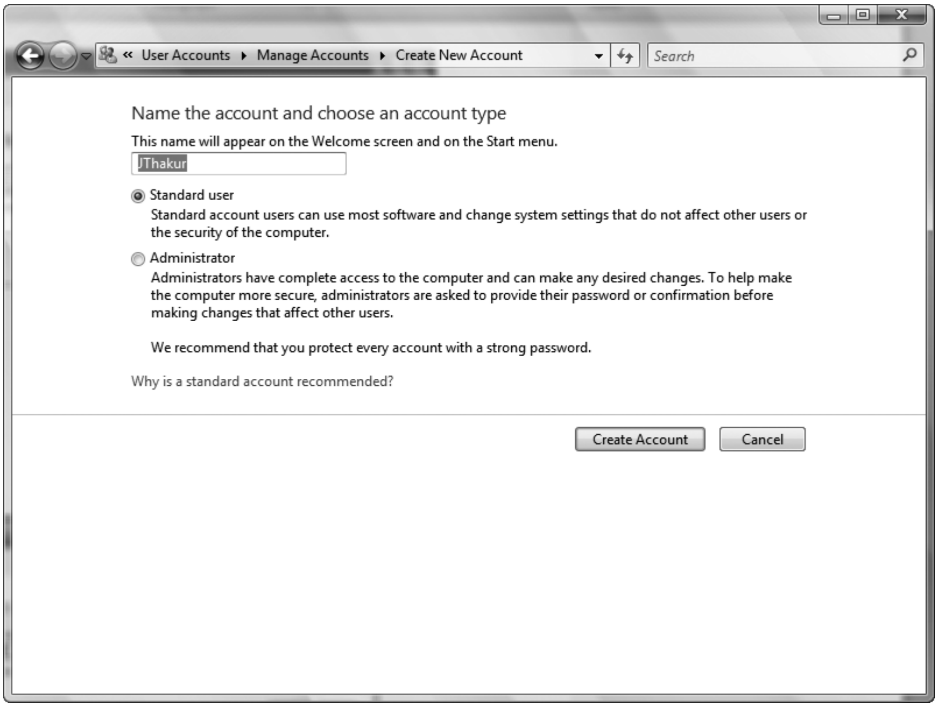
Once we have logged in to Windows, SQL Server uses a trusted connection when working with Windows Authentication mode. This means that SQL Server is trusting that the username and password have been validated as we just mentioned. If, however, the username does not exist, then based on the user ID alone, 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 if 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 with his or her user ID and password, providing they have access to it. 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.

We are 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, which the next chapter covers. We will keep that area simple, 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 Windows Vista, you may need to change your user control access to avoid many dialog boxes confirming that you wish to continue with each step.
2. From Start /Control Panel, select User Accounts.
3. When the Users and Passwords dialog box comes up, click Create a New Account on XP or Manage Another Account on Vista, followed by Create New Account.
4. Once the Name the Account and Choose an Account Type dialog box comes up, enter the username JThakur, as shown in Figure 1-11.
5. Ensure that the account type specified is Limited on XP or Standard on Vista. This means that it will not have administrator privileges. Once ready, click Create Account.



Name the account and choose an account type

This name will appear on the Welcome screen and on the Start menu.

JThakur

[✓ Standard user]
Standard account users can use most software and change system settings that do not affect other users or the security of the computer.

Administrator
Administrators have complete access to the computer and can make any desired changes. To help make the computer more secure, administrators are asked to provide their password or confirmation before making changes that affect other users.

We recommend that you protect every account with a strong password.

Why is a standard account recommended?

Figure 1-11. *Creating a new user account*

6. Stay in the Name the Account and Choose an Account Type dialog box, as you want to add a second username. Repeat the preceding process using the following details:

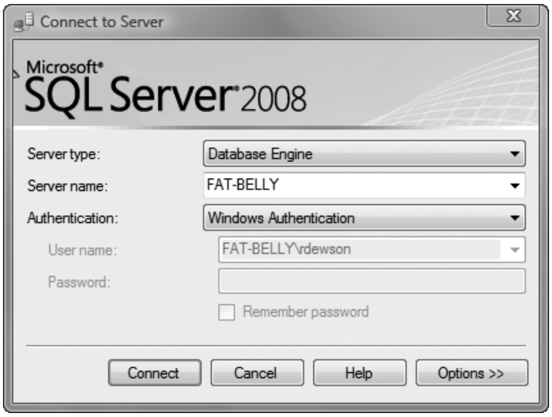
   Username: VMcGlynn

   Account type: (Computer) Administrator

7. Log off from Windows and then log on using the first ID that you created: JThakur.
8. Once logged in, start up SQL Server Management Studio by selecting Start ➤ All Programs ➤ Microsoft SQL Server 2008 ➤ SQL Server Management Studio. You will need to populate the dialog with the server name of the install. Click on Browse For More, then select Database Engine and select the install. We go through this in more detail in Chapter 2. The dialog should look like Figure 1-12.
9. Examine the error message that appears, which should resemble what you see in Figure 1-13. JThakur 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 JThakur is a Limited user.
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 we created—VMcGlynn. Once logged in, start up SQL Server Management Studio and connect to your server. This time the login will work.

We have created two usernames: one that has restricted access (JThakur) and one that has administration rights (VMcGlynn). However, neither of these specific usernames exists 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 would be 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 1-14, neither JThakur nor VMcGlynn exist.

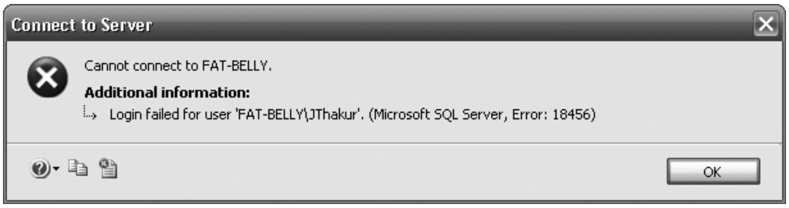


Connect to Server

Microsoft® SQL Server® 2008

Server type: Database Engine
Server name: FAT-BELLY
Authentication: Windows Authentication
User name: FAT-BELLY\rdewson
Password:
Remember password

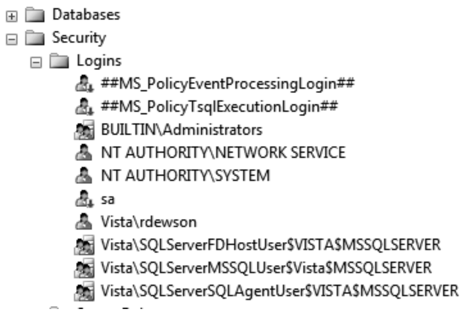
Figure 1-12. *Attempting to connect to SQL Server*



Connect to Server

Cannot connect to FAT-BELLY.
Additional information:
Login failed for user 'FAT-BELLY\JThakur'. (Microsoft SQL Server, Error: 18456)

Figure 1-13. *Failed login to server*



```
Databases
Security
  Logins
    ##MS_PolicyEventProcessingLogin##
    ##MS_PolicyTsqlExecutionLogin##
    BUILTIN\Administrators
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    sa
    Vista\rdewson
    Vista\SQLServerFDHostUser$VISTA$MSSQLSERVER
    Vista\SQLServerMSSQLUser$Vista$MSSQLSERVER
    Vista\SQLServerSQLAgentUser$VISTA$MSSQLSERVER
```

Figure 1-14. *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.

Mixed Mode

If we installed SQL Server with mixed mode, this means we could 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 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 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 left 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 had selected mixed mode authentication during the installation process, you would have seen that 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 up the server as implementing 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, it would be possible for that user to 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 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 include letters than can be made into numbers and numbers into letters. 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 will 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 logged on with an identifiable name, he or she would have an ID in SQL Server, which we could 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 when we'll 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 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 if it is suitable for a SQL Server installation.

By following the steps given earlier, you should have a successful installation of SQL Server 2008 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 2008. One of the best ways of managing SQL Server is by using SQL Server Management Studio, which will be discussed next.