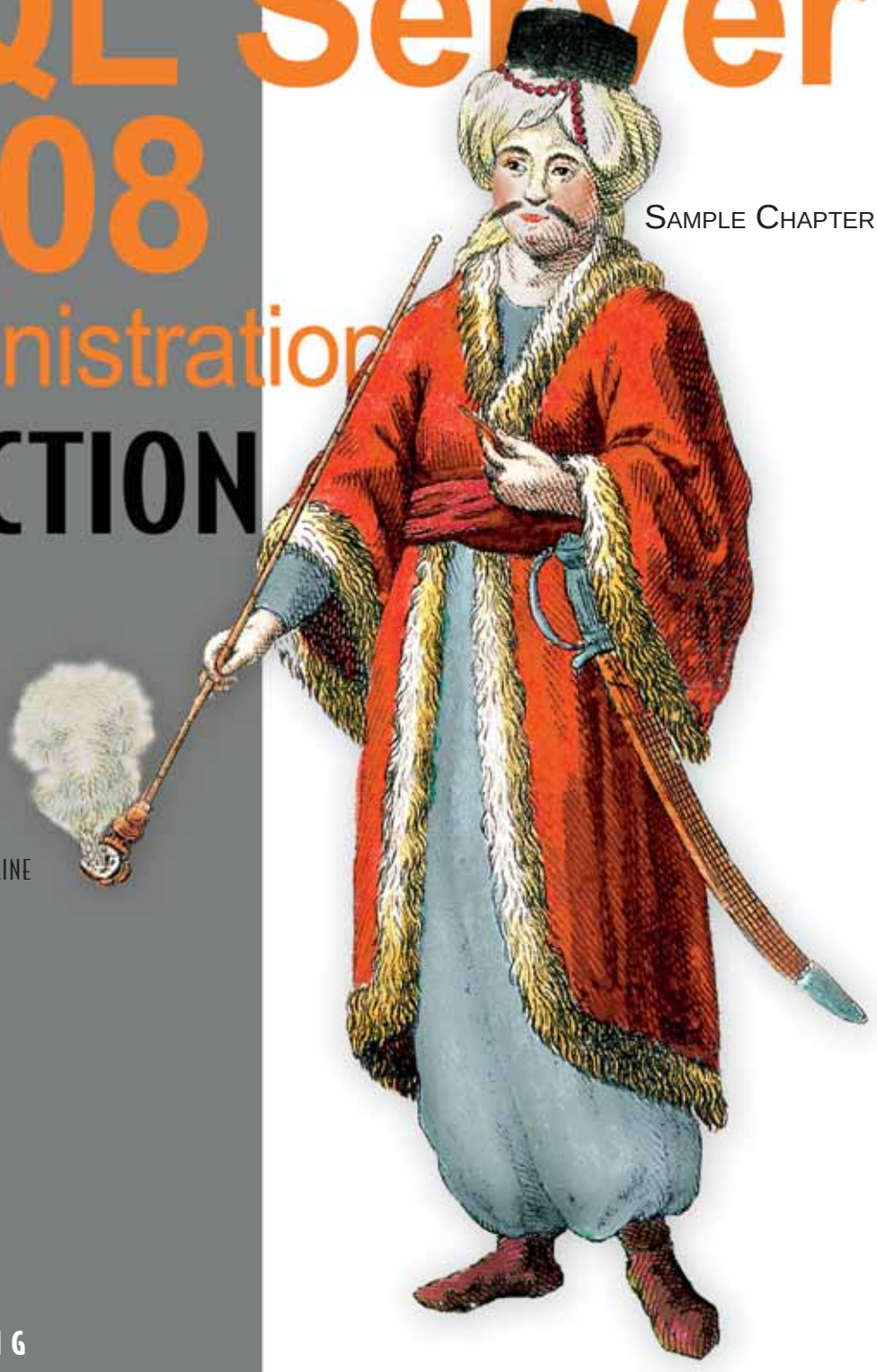
SQL Server 2008 Administration IN ACTION

SAMPLE CHAPTER



Rod Colledge

FOREWORD BY KEVIN KLINE

MANNING

SQL Server 2008
Administration in Action
by Rod Colledge

Chapter 4

brief contents

4 Installing and upgrading SQL Server 2008

In this chapter, we'll cover

- Preparing for installation
- Installing SQL Server 2008
- Upgrading to SQL Server 2008
- Developing a service pack upgrade strategy

With SQL Server 2008's new and enhanced features, you can install and configure SQL Server instances that comply with best practices much easier than with earlier versions. Starting with the installation wizard, there are several new options, such as the ability to specify separate directories for backups, transaction logs, and the tempdb database. On the configuration side, policy-based management (which we'll discuss in detail in chapter 8) lets you store instance configuration settings in XML configuration files that can be applied to a server instance after installation.

The next chapter will focus on clustered installations of SQL Server. This chapter covers the installation and upgrade of nonclustered SQL Server instances. We'll start with some important preinstallation tasks, run through the installation process, and finish with upgrade techniques.

4.1 Preparing for installation

Adequate preinstallation planning is a crucial element in ensuring a successful SQL Server installation. A large percentage of problems with SQL Server environments can be traced back to poor installation choices, often performed by those with minimal SQL Server skills.

In this section, we'll cover the importance of a preinstallation checklist before looking at some additional preinstallation tasks, such as the creation of service accounts and directories.

4.1.1 Preinstallation checklist

Creating a preinstallation checklist is a great way to make sure appropriate attention is paid to the important elements of an installation. A checklist is particularly useful in environments where DBAs aren't involved in the installation of SQL Server. By creating and providing thorough checklists, you ensure that the chances of a successful deployment are significantly improved.

Figure 4.1 shows an example of a preinstallation checklist. The important point here isn't necessarily the contents of the checklist, but the fact that you create one and tailor it to the needs of your environment.

A lot of the items covered in the checklist shown in figure 4.1 are taken from the previous two chapters, and others will be covered in subsequent chapters. Let's move on now to look at some of these, beginning with the creation of service accounts.

SQL Server Preinstallation Checklist

Storage	RAID configuration	**Misc**	BIOS & firmware versions
	Battery backed write optimized cache		Physical security
	Partition offset		Antivirus configuration
	64K allocation unit size		WMI
	Multipathing		Pending reboots
	SQLIO/SIM checks		Windows service packs and hotfixes
	LUN configuration & zoning		No domain controller role
	Backup, tempdb, T-log, DB : volumes & directories		
CPU/Memory	PAE/3GB settings	**Service Accounts**	Separate, non-privileged accounts for each service
	NUMA configuration		Password expiration policy
	Page file configuration		Lock pages in memory
			Perform volume maintenance tasks
Clustering	IP addresses	**Network**	Manual configuration
	MSDTC in dedicated resource group		Switched gigabit connections
	Network priority & bindings		ISCSI NIC teaming
	Private LAN: connectivity & ping time		Windows & perimeter firewall configuration
	FCCP certification		Disable NETBIOS & SMB

Figure 4.1 Along with other strategies such as policy-based management, a preinstallation checklist enables a SQL Server installation to have the best chance of meeting best practice.

4.1.2 Service accounts

A SQL Server installation will create several new Windows services, each of which requires an account under which it will run. As we'll see shortly, these accounts are specified during installation, so they need to be created in advance.

Depending on which features are installed, SQL Server setup creates the following services for each installed instance:

- SQL Server
- SQL Server Agent
- SQL Server Analysis Services
- SQL Server Reporting Services
- SQL Server Integration Services

Prior to installation, you should create service accounts for each of these services with the following attributes:

- *Domain accounts*—While you can use local server accounts, domain accounts are a better choice as they enable the SQL instance to access other SQL Server instances and domain resources, as long as you grant the necessary privileges.
- *Nonprivileged accounts*—The service accounts do not, and should not, be members of the domain administrators or local administrator groups. The installation process will grant the service accounts the necessary permissions to the file system and registry as part of the installation. Additional permissions beyond those required to run SQL Server, such as access to a directory for data import/export purposes, should be manually granted for maximum security.
- *Additional account permissions*—Two recommended account permissions that SQL Server doesn't grant to the SQL Server service account are Perform Volume Maintenance Tasks, required for Instant Initialization (covered in chapter 9), and Lock Pages in Memory, required for 32-bit AWE-enabled systems and recommended for 64-bit systems (this setting is covered in more detail in chapter 7).
- *Password expiration and complexity*—Like any service account, the service accounts for SQL Server shouldn't have any password expiration policies in place, and the passwords should be of adequate complexity and known only to those responsible for service account administration.
- *Separate accounts for each service*—Each SQL Server service for each installed instance should be configured with a separate service account. This allows for the most granular security, a topic we'll examine further in chapter 6.

4.1.3 Additional checks and considerations

Before we launch into an installation, let's discuss a number of other important preinstallation checks and considerations:

- *Collation*—Like Windows, SQL Server uses collations to determine how characters are sorted and compared. As we'll see shortly, a collation is chosen during installation, and by default, SQL Server setup will select a collation to match the server's Windows collation. An inconsistent collation selection is a common cause of various administration problems, and a well-considered selection is therefore a crucial installation step. In almost all cases, you should accept the default collation during installation; if you choose a custom collation, take into

account the potential collation conflicts when dealing with data from another instance with a different collation. SQL Server Books Online (BOL) covers this important topic in detail.

- *Storage configuration*—In previous chapters, we covered the importance of storage configuration. Prior to installation, you must ensure partitions are offset[1] and formatted with a 64K allocation unit size. Further, run SQLIO and SQLIOSIM to validate storage performance/validity and driver/firmware versions are up to date. Additional checks include ensuring multipathing software is installed and working, and consider NIC teaming for maximum performance and redundancy for iSCSI installations.
- *Directory creation*—One of the installation steps is to specify locations for the database data and log files, backup files, and the tempdb data and log files. For maximum performance, create each of these directories on partitions that are physically separate from each other—that is, they don't share the same underlying disks. Directories for these objects should be created before installation. Chapter 9 discusses the importance of physical disk separation in more detail.
- *Network security*—SQL Server should be secured behind a firewall, and unnecessary network protocols such as NetBIOS and SMB should be disabled. Chapter 6 provides detailed coverage on this process.
- *Windows version*—SQL Server 2008 requires at least Windows Server 2003 Service Pack 2 as a prerequisite for installation; Windows Server 2008 is recommended for the best performance and security. Further, SQL Server shouldn't be installed on a primary or backup domain controller; the server should be dedicated to SQL Server.
- *Server reboot*—SQL Server won't install if there are any pending reboots;[2] therefore, reboot the server prior to installation if appropriate.
- *WMI*—The Windows Management Instrumentation (WMI) service must be installed and working properly before SQL Server can be installed. This service is installed and running by default on both Windows Server 2003 and 2008.

Windows Server 2008

The ideal underlying operating system for SQL Server is Windows Server 2008. Why? For starters, the networking stack in Windows 2008 is substantially faster, so there's an immediate boost in network transfer times. Second, the Enterprise and Data Center editions of Windows Server 2008 include Hyper-V, which provides virtualization opportunities for SQL Server instances. Other improvements over Windows Server 2003 include more clustering options, NUMA optimizations, and leaner installations that translate to less maintenance and smaller attack surfaces for better security.

---

[1] Windows Server 2008 does this automatically.

[2] Open the registry editor and navigate to HKLM\System\CurrentControlSet\Control\Session Manager. The existence of PendingFileRenameOperations is an indication of a pending reboot.

Now that we've covered the important preinstallation checks and planning, let's walk through an actual installation of SQL Server 2008.

4.2 Installing SQL Server

In this section, we'll walk through the installation of a SQL Server instance using a number of different techniques. Before we do that, let's cover an important installation selection: the choice between a default and a named instance.

4.2.1 Default and named instances

Since SQL Server 2000, multiple instances (copies) of SQL Server can be installed on one server, thus providing various benefits, such as the ability to control the amount of memory and CPU resources granted to each instance, and the option to maintain different collation and service pack levels per instance. Such benefits are crucial for server consolidation projects, and we'll spend more time on some of these benefits in chapter 7 when we cover the process of configuring memory usage on multi-instance servers.

As we'll see shortly, one of the choices during installation of SQL Server is the selection between a named instance and a default instance. While there can only be a single default instance per server, the Enterprise edition of SQL Server 2008 supports the installation of up to 50 named instances.[3]

When connecting to SQL Server, the instance name is specified in the connection string; for example, *BNE-SQL-PR-01\SALES* will connect to the SALES instance on the BNE-SQL-PR-01 server. In contrast, connecting to the default instance requires the server name only—that is, *BNE-SQL-PR-01*.

In addition to the instance name, SQL Server 2008 uses an *instance ID*, which by default has the same value as the instance name. The instance ID is used to identify registry keys and installation directories, particularly important on servers with multiple installed instances.

With this background in mind, let's install an instance of SQL Server using the GUI installation wizard.

4.2.2 GUI installation

Rather than bore you with every installation step, most of which are self-explanatory, I'll summarize the installation and include screen shots for the most important steps. Start the installation by running setup.exe from the SQL Server DVD. The setup process begins with a check on the installed versions of the Windows Installer and the .NET Framework. If the required versions are missing, the setup process offers the choice to install them. After these components are verified (or installed), setup begins with the SQL Server Installation Center, as shown in figure 4.2.

[3] Other editions support up to 16 instances.

Figure 4.2 The Installation Center allows you to perform various installation-related tasks.

1. The Installation Center is the starting point for a wide variety of tasks. For our example, let's start by clicking the Installation tab and then selecting the "New SQL Server stand-alone installation or add features to an existing installation" option. Setup begins with a check for potential problems that may prevent an installation from completing successfully. You can view details of the checks by clicking Show Details. Address any problems preventing installation, or click OK to continue.
2. Click Install to install the required setup support files.
3. In the Setup Support Rules screen, additional checks are processed before installation continues; for example, the installer warns of the presence of Windows Firewall with a warning to unblock appropriate ports. Review the warnings/failures (if any) and click Next.
4. The Installation Type screen lets you choose between installing a new instance or adding features to an existing instance. For our example, let's choose the default (Perform a New Installation) and click Next.
5. The Product Key screen asks you to select between a free edition (Enterprise Evaluation or Express) or the option to enter a product key (supplied with the purchase of SQL Server). Make the appropriate choice and click Next.
6. At the license terms screen, review the terms, check the "I accept the license terms" box, and click Next.
7. On the Feature Selection screen shown in figure 4.3, select the appropriate features and choose an installation directory (or accept the default). You can display additional information on each feature by clicking on the feature name. Click Next.
8. In the Instance Configuration screen, shown in figure 4.4, choose between a default or a named instance, enter the instance ID and root directory (or accept the default settings), and click Next.
9. The Disk Space Requirements screen confirms the existence (or absence) of the necessary disk space for installation to proceed. Review the summary of required and available space and click Next to continue.

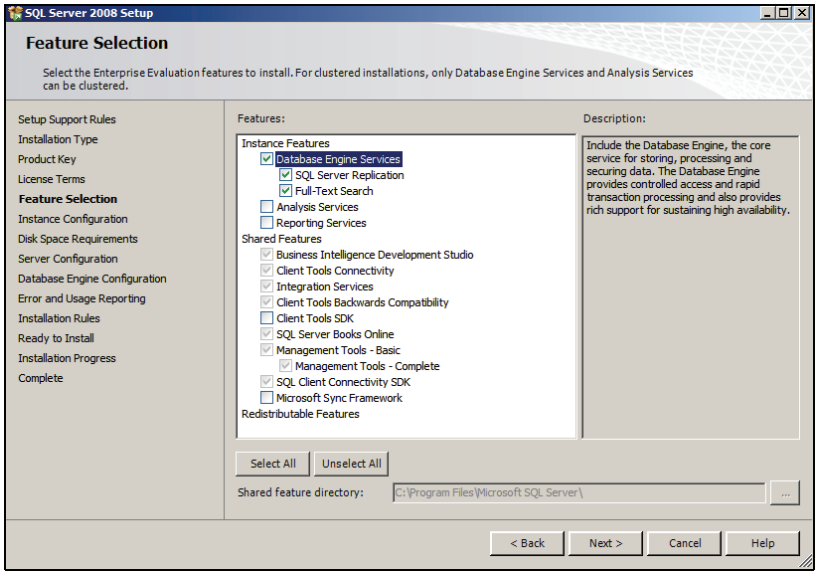


Figure 4.3 The Features screen enables you to select various features for installation.

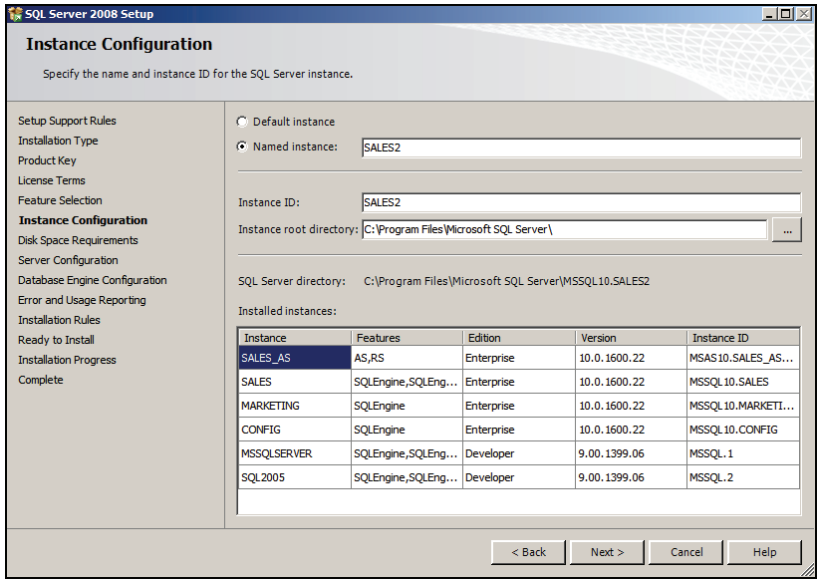


Figure 4.4 This screen lets you select between a default and a named instance.

10 In the Server Configuration screen, shown in figure 4.5, enter the account names and passwords for the SQL services, and optionally change the startup type. As we discussed earlier in the chapter, these accounts should be created as standard privilege accounts prior to installation. Before clicking Next to continue, click the Collation tab to review (and optionally modify) the default collation. As we covered earlier, use caution when selecting a custom collation.

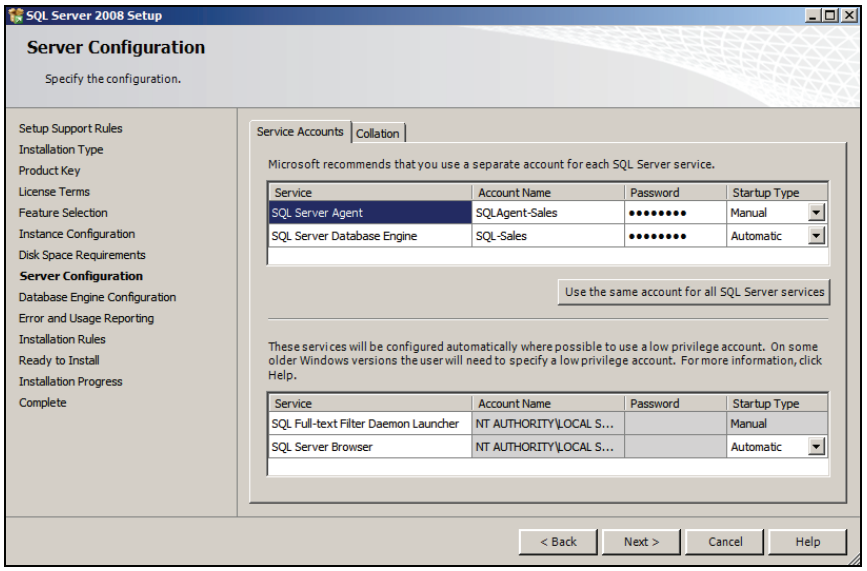


Figure 4.5 On this screen, you select a service account and startup type for each SQL service.

11 On the first tab of the Database Engine Configuration screen, shown in figure 4.6, you select the authentication mode for the instance: Windows or Mixed Mode. As we'll discuss in chapter 6, Windows authentication mode is the most secure option, and is therefore the default (and recommended) option. If you choose Mixed Mode, be sure to enter a strong system administration (SA) account password. Regardless of the selected authentication mode, click either

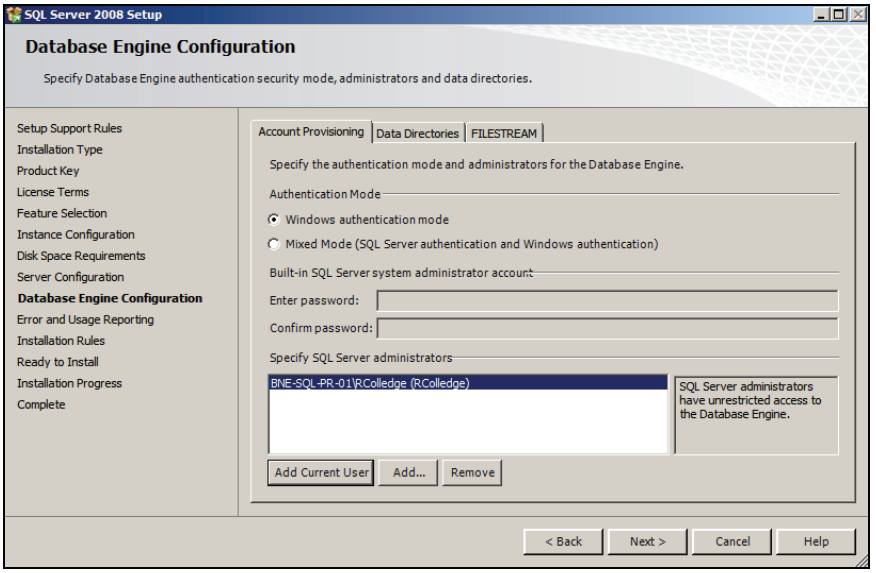


Figure 4.6 The Account Provisioning tab allows you to select the authentication mode and SQL Server administrators.

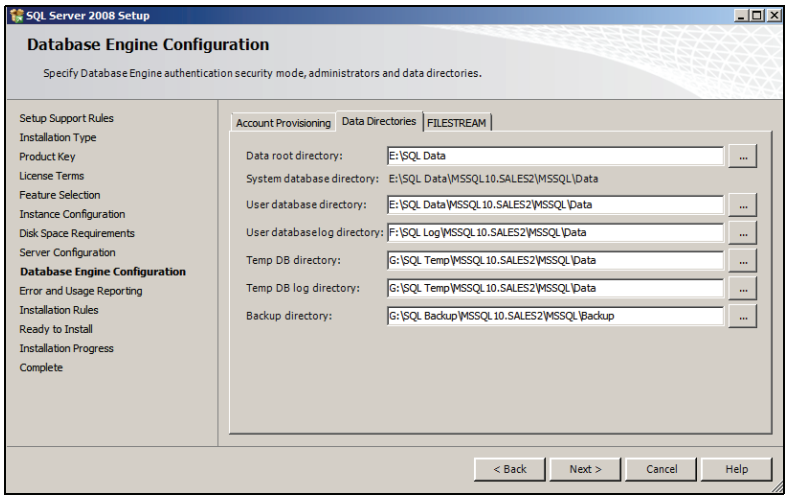


Figure 4.7 On the Data Directories tab, specify custom directories for data, logs, backup, and tempdb.

Add Current User or Add to select a user to add to the SQL Server administration group. Unlike earlier versions, SQL Server 2008 setup enforces this selection as a secure alternative to adding the BUILTIN\Administrators group to the SQL Server administration role. We'll explain this in more detail in chapter 6. To continue, click the Data Directories tab.

12 The Data Directories tab, as shown in figure 4.7, lets you specify default directories for data, log, tempdb, and backup directories. As covered earlier, physical disk separation of these directories is important, and we'll address this topic in greater detail in chapter 9. After entering the directory locations, click the FILESTREAM tab to continue.

13 Use the FILESTREAM tab to configure the instance for FileStream access. As you'll see in chapter 9, FileStream is a new option for binary large object (BLOB) management. Regardless of the selection at this point, FileStream can be configured as a postinstallation task. After reviewing the options on this tab, click Next.

14 In the remaining installation steps, you'll accomplish the following:

- Specify whether to send error reports and feature usage data to Microsoft
- Review final installation rules checks
- View the summary of installation choices, and click Install to execute the installation based on the previous selections
- View the installation progress
- On the Completion screen, view the installation log file

When installation is complete, SQL Server saves the choices you made during setup in ConfigurationFile.ini, which you'll find in the C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log*yyyymmdd_hhmmss* directory. You can use this file to confirm the installation proceeded with the required options, as well as use it as a base

for subsequent unattended installations via the command prompt. We'll cover these options shortly.

After installation, you must perform a number of important configuration activities, such as sizing the tempdb database, setting minimum and maximum memory values, and creating SQL Agent alerts. We'll cover these tasks in subsequent chapters.

4.2.3 Command prompt installations

In addition to using the GUI installation wizard that we've just covered, you can install SQL Server 2008 from the command prompt, as you can see in figure 4.8. You can find the syntax and options in SQL Server BOL in the "How to: Install SQL Server 2008 from the Command Prompt" topic.

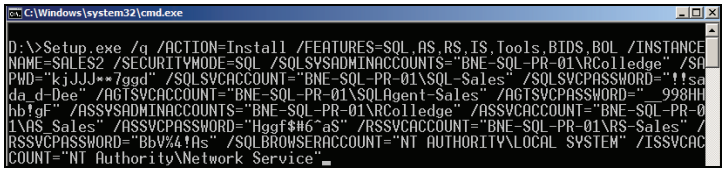


Figure 4.8 SQL Server 2008 command-line installation

As mentioned earlier, the ConfigurationFile.ini is created at the end[4] of a GUI-based installation. This file should be preserved in its original state for later analysis, but you can make a copy and use it for subsequent installations at the command prompt via the `/Configurationfile` parameter, as shown in figure 4.9.

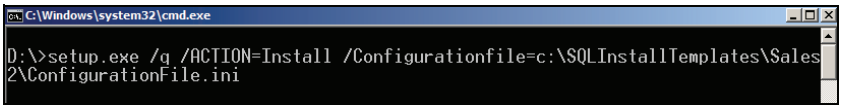


Figure 4.9 Use the `/Configurationfile` option at the command line to direct SQL Server to install based on the contents of a ConfigurationFile.ini file.

Command prompt installations with configuration files are ideal in standardizing and streamlining installation, particularly when installations are performed by those without the appropriate SQL Server knowledge.

SQL Server 2008 can be installed alongside earlier versions of SQL Server. Doing so is a common technique in *migrating* databases, an alternative to an *in-place upgrade*, both of which we'll cover next.

4.3 Upgrading to SQL Server 2008

Depending on the environment, the upgrade to SQL Server 2008 can be complex, particularly when technologies such as replication and clustering are involved. In this section, rather than attempt to cover all of the possible upgrade issues, I've aimed for the more modest task of providing you with an insight into the various upgrade techniques.

---

[4] The INI file can also be created by proceeding through the GUI installation, but cancel it at the very last step, on the Ready to Install page.

SQL Server 2008 Upgrade Technical Reference Guide

As with installation, upgrading to SQL Server 2008 requires considerable planning and preparation. Microsoft recently released the *SQL Server 2008 Upgrade Technical Reference Guide*. Weighing in at 490 pages and available for free download from the Microsoft website, this guide is essential reading as part of any upgrade project and contains important information on best practices and specific advice for various upgrade scenarios.

Depending on the availability of spare hardware and the allowed downtime, you can choose one of two upgrade techniques. The first is known as an *in-place* upgrade, in which an entire instance of SQL Server 2000 or 2005 along with all of its databases are upgraded in one action. Alternatively, you can use the *side-by-side* technique, in which individual databases can be migrated one at a time for more control. Both of these techniques have their advantages and disadvantages, as you'll see shortly.

Before we look at the details of in-place versus side-by-side, let's discuss the importance of analyzing the target database/instance before upgrading by using SQL Server's Upgrade Advisor tool.

4.3.1 Upgrade Advisor

Each new version of SQL Server contains behavioral changes, some major, some minor. In any case, even small, subtle changes can significantly impact application behavior. Regardless of the upgrade technique, a crucial step in preparing for an upgrade to SQL Server 2008 is to analyze the upgrade target and determine whether any issues require attention. Together with the appropriate upgrade technique, such analysis is essential in minimizing unexpected issues, making the upgrade process as smooth as possible.

SQL Server 2008, like 2005, includes an Upgrade Advisor tool, which you can use to examine existing 2000 and/or 2005 instances to determine whether any issues will prevent an upgrade from completing (blocking issues) or backward-compatibility issues that may lead to application failure after the upgrade. The Upgrade Advisor tool (which you access from the Planning menu of the SQL Server Installation Center as shown in figure 4.2) can be used to examine all SQL Server components, including Analysis Services, Reporting Services, Integration Services, and the core database engine itself.

The Upgrade Advisor has two main components: an analysis wizard and a report viewer. Use the wizard to select an upgrade target (SQL Server components, instances, and databases) and begin the analysis. Once complete, the report viewer can be used to view the details of the analysis, as shown in figure 4.10.

Like the 2005 version, the Upgrade Advisor tool can analyze trace files and Transact-SQL (T-SQL) scripts. You perform this analysis as a proactive measure to identify possible issues with application-generated data access code or T-SQL scripts, such as backup scripts used in scheduled maintenance plans.

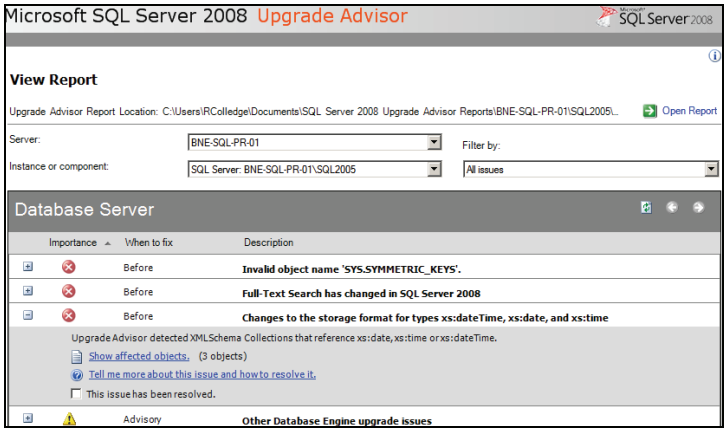


Figure 4.10 The Upgrade Advisor wizard lets you inspect a SQL Server 2000 or 2005 instance before upgrading to detect issues requiring attention.

Running the Upgrade Advisor produces a list of tasks that fall into two categories: those that must be completed before an upgrade can be performed, and those for attention once the upgrade is complete. Rather than list all of the possible post-upgrade tasks, here are the recommended ones for the core database engine:

- *Compatibility level check*—Post-upgrade, the *compatibility level* of the database is left at the pre-upgrade version. Although this is advantageous in minimizing the possibility of applications breaking due to the use of older language syntax that's no longer compatible with SQL Server 2008, some new functionality and performance optimizations won't be available until the new compatibility level is applied. Leaving an upgraded database at a previous compatibility level should be seen as an interim migration aid, with application code updated as soon as possible after installation. You can change the compatibility level of an upgraded database using the `sp_dbcmptlevel` stored procedure, or via the database properties in Management Studio, as shown in figure 4.11.
- *Max Worker Threads*—If you're upgrading from SQL Server 2000, the Max Worker Threads setting, covered in chapter 7, is kept at the default value of 255. After the upgrade, change this value to 0, which allows SQL Server to determine the appropriate value based on the number and type of CPUs available to the instance.
- *Statistics update*—Although SQL Server 2008 can work with statistics generated from earlier versions, I recommend you perform a full statistics update to take advantage of the query optimizer improvements in SQL Server 2008. Chapter 13 will discuss statistics in more detail.

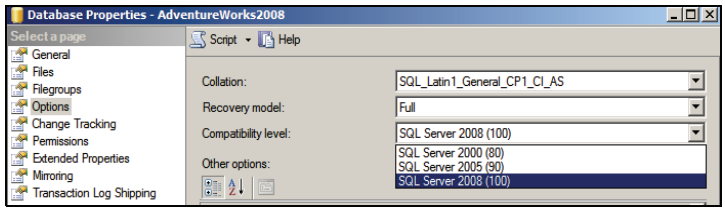


Figure 4.11 After you upgrade, the compatibility level of a database should be set to SQL Server 2008, once testing has confirmed the absence of any unexpected problems.

- *Configuration check*—Like 2005, SQL Server 2008 complies with the *secure by default* installation mode, whereby certain features are disabled upon installation. We'll cover these issues in chapter 6. Depending on the installation, certain required features may need to be manually enabled postinstallation.
- *DBCC UPDATEUSAGE*—Databases upgraded from SQL Server 2000 may report incorrect results when using the `sp_spaceused` procedure. Running the `UPDATEUSAGE` command will update the catalog views used by this procedure, thus fixing this inaccuracy.

Before looking at the upgrade methods, note that regardless of the method you choose, performing a test upgrade is crucial in identifying possible issues. Test upgrades provide an opportunity to

- Determine the length of the upgrade process, crucial in planning downtime for the *real* upgrade.
- Determine application behavior after the upgrade. This will allow you to set the database compatibility level to SQL Server 2008 and ensure applications work as expected.
- Performance-test applications on the upgraded database. If performance is poor, a test upgrade provides a chance of investigating the reasons prior to the production upgrade.
- Last but not least, complex upgrades involving components such as replication are certainly not something you want to be doing the first time in production!

So, with these points in mind, let's take a look at the two upgrade methods, starting with the in-place method.

4.3.2 In-place upgrade

The *in-place* upgrade method upgrades an instance, and all of its databases, in a single irreversible action. For simple, small database instances, it provides the easiest and quickest upgrade method and has the following additional benefits:

- Applications connecting to any of the databases don't need modification. The instance name remains the same (if upgrading from a named instance), so no connection strings need to be changed.
- No additional hardware is required. The instance executables and data are changed in place.

Despite these benefits, there are some significant downsides to this method, making it unsuitable for a range of scenarios:

- All of the instance's databases have to be upgraded at once. There's no option to upgrade only some. If one of the databases (or its applications) needs modification before upgrading, then all of the other instance databases will have to wait before the upgrade can proceed.
- A failed upgrade, or one that produces unexpected results, can't be rolled back, short of running up new hardware and restoring old backup files, or using a virtualization snapshot rollback process.

Because the rollback options are limited with this method, it's critical that you complete a full backup and DBCC check on all databases before beginning the upgrade. If database activity occurs after the full backup, make transaction log backups immediately prior to the upgrade.

To begin the in-place upgrade, select Upgrade from SQL Server 2000 or SQL Server 2005 from the Installation menu of the SQL Server Installation Center.

For greater control of the upgrade process, you can choose one of several side-by-side upgrade methods.

4.3.3 Side-by-side upgrade

In contrast to an in-place upgrade, which upgrades all databases for a given instance, a side-by-side upgrade is a *per database* method:

1. A SQL Server 2008 instance is installed as a *new* installation (compared to an upgrade). The new instance can be installed on either the same server as the instance to be upgraded (legacy instance), or on a new server.
2. Each database to be upgraded is migrated from the legacy instance to the new 2008 instance, using one of several methods I'll describe shortly. Databases are migrated individually on their own schedule, and possibly to different destination servers.
3. Once the new 2008 instance is up and running, the legacy instance is decommissioned, or retained in an offline state for rollback purposes if required.

The side-by-side upgrade method offers several advantages over the in-place method. The major advantage is that if something goes wrong with the upgrade, or unexpected results render the upgrade a failure, the legacy instance is still available and unchanged for rollback purposes. Further, the upgrade is granular; individual databases can be migrated with others remaining on the original server, migrated to a different server, or migrated at a later point.

Disadvantages and complexities of this method when compared to the in-place method are as follows:

- Application connection strings will need to be modified to point to the new instance name.
- Security settings, maintenance plans, and SQL Server Agent jobs will need to be re-created on the new 2008 instance, either manually or from a script.
- If the new 2008 instance is on the same server as the legacy instance, the capacity of the server to run both instances in parallel must be taken into account. A possible workaround for this issue is to limit the resources of one of the instances, such as the maximum memory and CPU affinity, for the period of time that both are actively running.
- Downtime is typically longer than for an in-place method; there are several techniques used to limit this, as you'll learn in a moment.

Side-by-side upgrades are often scheduled for the same time as server replacements—that is, the new server is purchased, installed, configured, and loaded with a

new instance of SQL Server 2008, and databases are migrated from the legacy instance. At the completion of this process, the legacy server is decommissioned, cycled back to lower environments, or used for other purposes.

The method used to migrate databases as part of a side-by-side upgrade is an important consideration in determining the rollback and downtime implications for the upgrade, particularly for larger databases. Let's walk through the major methods used, beginning with backup/restore.

BACKUP/RESTORE

The backup/restore method is straightforward and keeps the original database in place for rollback purposes. A full database backup is made on the legacy database and restored to the new SQL Server 2008 instance. As part of the restore process, SQL Server will upgrade the internal structures of the database as necessary.

A variation on this approach involves filegroup backups and piecemeal restores. These topics will be discussed in more detail in chapters 9 and 10, but essentially this involves backup and restore of the legacy database's primary filegroup to the new 2008 instance. After this, the database is online and available on the new 2008 instance, after which individual filegroups can be backed up and restored using a piecemeal approach in priority order.

ATTACH/DETACH

The attach/detach method involves detaching the legacy database and attaching to the new 2008 instance. Similar to a restore, SQL Server will upgrade the internal structure as part of the attach process. To keep the original database available for rollback, you can copy the database files to the new server before attaching them to the new 2008 instance. After the copy is complete, the database can be reattached to the legacy instance for rollback purposes if required.

TRANSACTION LOG BACKUP/RESTORE

Depending on the size of the database to be migrated, the time involved in copying either the data files or backup files in the previous two methods may exceed the downtime targets. For example, if the backup file was hundreds of gigabytes and had to be copied over a slow network link, the copy could take many hours to complete. To reduce downtime, a third method can be used involving transaction log backups. This method is similar to setting up log shipping (covered in chapter 11) and involves these steps:

1. A full database backup of the legacy database is taken and copied to the new SQL Server 2008 instance. The legacy database remains online and in use throughout the copy process.
2. The legacy database is restored on the new 2008 instance *WITH NORECOVERY* (full details provided in chapter 10).
3. Finally, at the moment of migration, users are disconnected from the legacy database, and a transaction log backup is made and copied to the 2008 instance.
4. The transaction log backup is restored *WITH RECOVERY.*
5. At this point, application connection strings are redirected to the 2008 instance and users are reconnected.

There are several variations of this method. If the transaction rate is very high, the size of the transaction log backup in step 3 may be very large; if so, regular transaction log backups can be made leading up to this step, reducing the size (and therefore copy time) of the final transaction log backup. If using this method, restore all but the last of the transaction log backups *WITH NORECOVERY*.

TRANSACTIONAL REPLICATION

This method is similar to the transaction log backup/restore but involves replication:

1. Transactional replication is set up from the legacy instance to the new 2008 instance.
2. At the moment of migration, replication is stopped and applications are redirected to the new 2008 instance.
3. Optionally, replication can then be set up in the reverse direction to support a rollback scenario—that is, data entered on the new 2008 instance post-migration is copied to the legacy database instance to prevent data loss in the event of a rollback.

OTHER TECHNIQUES

Other migration techniques include using the *Copy Database wizard* (in Management Studio, right-click a database and choose Tasks > Copy Database) and manually creating the database on the new 2008 instance from script and performing bulk copy operations.

Table 4.1 compares the attributes of the various upgrade techniques.

Table 4.1 Upgrade options compared

Upgrade technique	Complexity	Rollback options	App reconfig	Downtime
In-place	Lowest	No	No	Lowest
Side-by-side				
—Backup/restore	Medium	Yes	Yes	Highest
—Detach/copy/attach	Medium	Yes	Yes	Highest
—Filegroup restore	Medium	Yes	Yes	Moderate
—T-Log backup/restore	Medium	Yes	Yes	Lowest
—Transaction replication	Highest	Yes	Yes	Lowest

The side-by-side upgrade method offers much more flexibility and granularity than the all-or-nothing in-place approach. In all cases, regardless of the upgrade method, planning is crucial for a successful upgrade. The same is true for the installation of service packs, our next topic.

4.4 Developing a service pack upgrade strategy

If you were to develop a list of the top ten issues that a group of DBAs will argue about, one that's sure to appear is the approach to installing service packs. Let's take a look at the various considerations involved before looking at a recommended approach.

4.4.1 Installation considerations

You must consider a number of important factors before making the decision to install a service pack:

- *Third-party application vendor support*—Most application vendors include supported SQL Server versions and service pack levels in their support agreements. In such cases, a service pack upgrade is typically delayed until (at least) it becomes a supported platform.
- *Test environments*—The ability to measure the performance and functional impacts of a service pack in a test environment is crucial, and doing so counters a common argument against their installation—the fear that they'll break more things than they fix.
- *Support timeframes*—Microsoft publishes their support lifecycle at http://support.microsoft.com/lifecycle. While an application may continue to work perfectly well on SQL Server 6.5, it's no longer officially supported, and this risk needs to be considered in the same manner as the risk of an upgrade. It's not uncommon to hear of situations in which an emergency upgrade is performed as a result of a bug in a platform that's no longer supported. Clearly, a better option is to perform an upgrade in a calm and prepared manner.

Complicating the decision to apply a service pack is the inevitable discussion of the need for application outage.

4.4.2 Application outage

A common planning mistake is to fail to consider the need for scheduled maintenance; therefore, a request to apply a service pack is often met with a negative response in terms of the impact on users.

Very few organizations are prepared to invest in the infrastructure required for a zero outage environment, which is fine as long as they realize the importance of planning for scheduled outages on a monthly or quarterly basis. Such planning allows for the installation of service packs and other maintenance actions while enabling the management of downtime and user expectations.

Incremental servicing model

Microsoft has recently moved to an incremental servicing model (http://support.microsoft.com/default.aspx/kb/935897/en-us) whereby cumulative updates, consisting of all hotfixes since the last service pack, are released every 2 months. In addition to the bi-monthly release, critical on-demand hotfixes will be delivered as soon as possible, as agreed between Microsoft and the customer experiencing the critical issue.

So with these issues in mind, let's take a look at a recommended approach for installing SQL Server service packs.

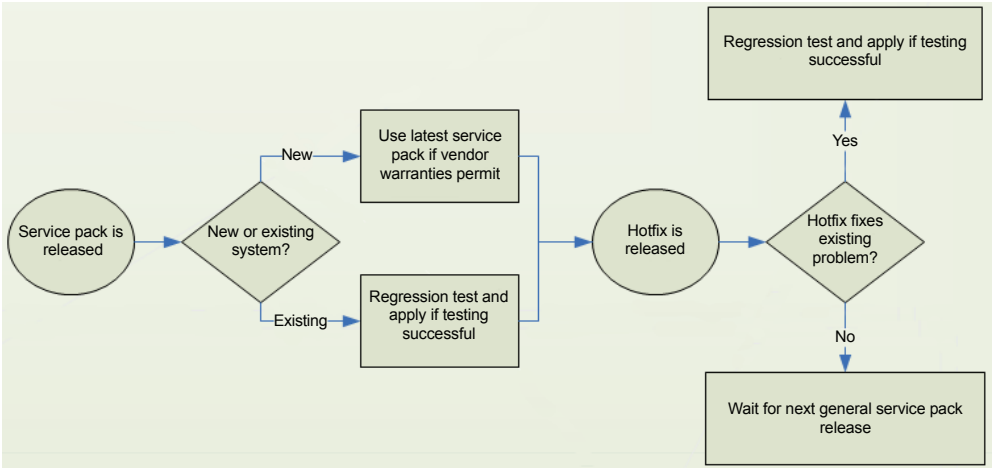


Figure 4.12 A recommended approach for implementing service packs and hotfixes

4.4.3 Recommended approach

Although each environment will have its own special circumstances, the following approach is generally accepted by most DBAs (see figure 4.12 for a summary):

- For a new deployment of SQL Server, use the latest service pack available, subject to application vendor support policies.
- For existing production systems, aim to apply service packs as soon as possible after their release—for example, at the next available/advertised maintenance window. This will require preparation and budget to ensure the availability of test environments for regression testing.
- Only apply hotfixes or cumulative updates if there's a particular need—that is, you're suffering from a bug or security vulnerability that's fixed in the release. If you're not in this category, wait for the next general service pack.
- If you're denied the chance to apply a service pack, for example, an objection to the required downtime or fear of the unknown consequences, ensure management is kept informed of the Microsoft support lifecycle for previous versions of SQL Server.

4.5 Best practice considerations: installing and upgrading SQL Server

Despite the ease with which SQL Server can be installed and upgraded, adequate preparation is essential in ensuring a stable and secure platform for later use:

- Review the best practices from the previous chapters to ensure hardware components are designed and configured appropriately.
- Prior to installation, create nonadministrator domain accounts for each instance/service combination, and ensure the accounts don't have any password expiration policies in place.

- Grant the SQL Server service account the Perform Volume Maintenance Tasks right; and for 32-bit AWE and 64-bit systems, also grant the Lock Pages in Memory right.
- Prior to installation of each SQL Server instance, prepare directories on separate physical disk volumes for the following database components:
  - Data files
  - Log files
  - Backup files
  - tempdb data and log
- Prior to installation, use the resources available in the Planning tab of the installation wizard. Included here (among others) are hardware and software requirements, security documentation, and online release notes.
- Only install the minimum set of SQL Server features required. This will increase security by reducing the attack surface area, and ensure unnecessary services aren't consuming system resources.
- Don't install SQL Server on a primary or secondary domain controller.
- Ensure consistency in the selection of collations across SQL Server instances, and unless compatibility is required with earlier SQL Server versions, select a Windows collation.
- Unless it's being used for applications that can't work with Windows authentication, don't choose the Mixed Mode option. If you do, make sure you choose a strong SA password and enforce strong password policies for all SQL Server logins.
- If you need to change any of the SQL Server service accounts after the installation, make changes using the SQL Server Configuration Manager tool. Changing in this manner ensures the appropriate permissions are granted to the new account.
- When installing SQL Server using the GUI wizard, take care to click and review each of the tabs before clicking Next. For example, the Database Engine Configuration screen lets you select the Authentication mode. Once you do this, clicking Next will skip the Data Directories and FILESTREAM tabs, which will enforce the default values. One of the ramifications of this is that data and log files will be created in the same directory, unless you manually change them after the installation.
- For a smoother installation process, particularly when a non-DBA is responsible for installation, use a tailored checklist. Alternatively (or as well as), copy and modify a ConfigurationFile.ini created from a successful installation and use the command-line method with the `/Configurationfile` option.
- Before upgrading, download and read the *SQL Server 2008 Upgrade Technical Reference Guide*. It contains important information on best practices and specific advice for various upgrade scenarios.

- Always run through a trial upgrade in a test environment using a recent copy of the production database (if possible). Doing so offers many benefits as well as circumvents potential problems. A trial upgrade will enable you to gather accurate timings for the actual production upgrade, determine the effect of the new compatibility level on application behavior, allow performance testing against a known baseline to determine the performance impact of the upgrade, and finally, develop a checklist to ensure the real upgrade runs as smoothly as possible.
- Use the Upgrade Advisor to analyze upgrade targets for issues that will prevent an upgrade or to gather a list of issues that will need to be addressed after the upgrade. If possible, feed SQL Server Profiler trace files into the Upgrade Advisor to examine any application code that may need to change before the upgrade.
- After the upgrade, attend to any issues identified by the Upgrade Advisor, including the following: setting the database compatibility level, updating statistics, checking configuration for features that need to be enabled, and if upgrading from SQL 2000, setting the Max Worker Threads option to 0 and running `DBCC UPDATEUSAGE`.
- The in-place upgrade method may be simple, but it exposes the possibility of having no rollback position if required. The various side-by-side upgrade methods offer more choices for rollbacks while also minimizing downtime.
- If using the in-place upgrade method, perform a full backup and DBCC check of each database prior to the upgrade.
- Using the Microsoft Assessment and Planning Toolkit Solution Accelerator tool is an effective means of discovering SQL Server instances and can be used as the starting point for consolidation and/or upgrade projects.
- Prior to any installation, upgrade, or service pack/hotfix install, always read the release notes for any late-breaking news and details on how certain components and features may be affected.
- Prepare for service packs. They're released for a good reason. Have both the time and environments ready for regression testing before applying in production, and consider any software vendor warranties before application.
- Only apply hotfixes and cumulative updates if there's a specific reason for doing so; otherwise, wait for the next service pack release.
- *Always* read the release notes that accompany service packs, hotfixes, and cumulative updates. Such notes often contain crucial information that may impact certain configurations.

Additional information on the best practices covered in this chapter can be found online at http://www.sqlCrunch.com/install.

In the next chapter, we'll discuss installing SQL Server on a failover cluster.

SQL SERVER/DATABASE

SQL Server 2008 Administration IN ACTION

Rod Colledge • Foreword by Kevin Kline

Ensuring databases are secure, available, reliable, and recoverable are core DBA responsibilities. This is a uniquely practical book that drills down into techniques, procedures, and practices that will keep your databases running like clockwork.

Open **SQL Server 2008 Administration in Action** and find sharply focused and practical coverage of

- Selecting and configuring server components
  - Configuring RAID arrays
  - Working with SANs and NUMA hardware
- New features in SQL Server 2008
  - Policy-based management
  - Resource Governor
  - Management Data Warehouse
- And much more!
  - Performance tuning techniques
  - Index design and maintenance
  - SQL Server clustering and database mirroring
  - Backup and restore

The techniques and practices covered in this book are drawn from years of experience in some of the world's most demanding SQL Server environments. It covers new features of SQL Server 2008 in depth. Its best practices apply to all SQL Server versions.

Rod Colledge is an SQL Server consultant based in Brisbane, Australia, and founder of sqlCrunch.com, a site specializing in SQL Server best practices. He's a frequent speaker at SQL Server user groups and conferences.

Free eBook
SEE INSERT

"Simply loaded with excellent and immediately useful information."
—From the Foreword by Kevin Kline, Technical Strategy Manager, Quest Software

"I thought I knew SQL Server until I read this book."
—Tariq Ahmed, coauthor of *Flex 4 in Action*

"A refreshing database administration book."
—Michael Redman, Principal Consultant (SQL Server), Microsoft

"Required for any MS DBA."
—Andrew Siemer, Architect, OTX Research

"It delivered way beyond my expectations... Packed with useful enterprise-level knowledge."
—Darren Neimke, Author of *ASP.NET 2.0 Web Parts in Action*

For online access to the author, code samples, and a free ebook for owners of this book, go to:

www.manning.com/SQLServer2008AdministrationinAction

ISBN 13: 978-1-933988-72-6
ISBN 10: 1-933988-72-X
54499
9 781933 988726

MANNING $44.99 / Can $56.99 [INCLUDING eBOOK]