2

Installing SQL Server 2008

Installing SQL Server 2008 is deceptively simple. I say *deceptively* because although SQL Server includes several wizards and tools that make the installation process itself go smoothly, a good database administrator will have devised a thorough plan for installing SQL Server and its requisite components. This chapter will introduce you to the process of installing SQL Server, beginning with an overview of the planning process. Although it would be impossible to document every possible design decision for every possible scenario, the goal of this chapter is to help you understand the installation process, some key design considerations, and the various components and options available prior to and during installation.

SQL Server Installation Planning

''There is never enough time to do it right, but always enough time to do it twice.'' ''Measure twice, cut once.'' How many times have you heard these sayings? There are a number of these clichés that point out that doing something right the first time means not having to do it over and over again. To avoid having to do it twice (or more!), you need to create a thorough plan. Too often installations are rushed and then must be uninstalled when technical issues arise. The questions that must be asked range from collation settings and named instances to the separation of log and data files. Will SQL Server be installed in a cluster? How about Storage Area Networks (SAN) or Network Attached Storage (NAS)? And virtualization, won't someone *please* think of the virtualization! Although the Installation Wizards will ask you to provide answers to several questions about *how* you want SQL Server installed, before you launch the Wizard you should know the *why* behind your answers.

In addition to the ''how'' and ''why,'' there are the ''who,'' ''what,'' and ''when'' questions that must be answered to create an adequate plan.

- ❑ The ''who'' is most likely going to be the database administrator (DBA), but other individuals will need to be included in the deployment plan as well. In addition to getting members of the IT department because there are network and storage considerations to account for, other departments or individuals that are considered key stakeholders may need to be involved in the process. Remember that SQL Server 2008 is an enterprise data platform, and users who own or interact with the data that will be managed by SQL Server will need to have their interests represented.

- The "what" question can be a bit more complex. The first "what" is "What features will be installed?" However, more "what" questions could include "What constitutes a successful installation?" or "What resources are required?"
- The "when" question is also imperative. "When will the installation be started and when will it be complete?"

It would be impossible to cover all the possible variations that could arise during a SQL Server installation, so this chapter covers only the essentials. Remember, when it comes to technology, the answer to almost every question is, "It depends." There are almost always "best practices," but sometimes the best practices are based on various "ivory tower" assumptions. We don't all have 50 billion dollars, 20,000 IT professionals, and unlimited access to hardware and software. Sometimes the "best practices" have to be left behind in favor of practicality and budget.

For example, as a best practice, transaction logs should be placed on a RAID 1 array as opposed to any striped array configuration because of how the transaction log is accessed by SQL Server. However, if the only available fault-tolerant storage is a RAID 5 striped array, then by all means it should be used to store and protect the log data. In many cases, the only storage available because of budget and hardware constraints is a single RAID 5 array where both the transaction log and data files are hosted. In a large enterprise solution, this would be completely unacceptable; but for a small-to-medium business implementation, it may be the only choice. The key point is that it is very important to know what the "best" solution is, but also keep in mind that compromises are often necessary to meet deadlines and budgetary constraints.

Hardware Considerations

Minimum requirements are exactly that: *minimum*. SQL Server will run on a system with minimum hardware, but the performance is not going to be stellar. Even the "recommended" hardware is to be exceeded whenever practical. I tend to think of these as "minimum to install and start the services" and "minimum to run a production system," respectively.

Upgrading almost any hardware object on a server hosting SQL Server 2008 will result in improved performance, but all things being equal, increasing RAM often has the best impact on performance. An underpowered processor or slow disk system will cause just as many performance problems as insufficient RAM, but RAM limitations will often cause processor and disk issues to be exacerbated.

A common scenario for certification exams often presents a series of questions that involve allocating different limited resources across different types of servers such as a file server, domain controller, and database server. Often, you're tasked with determining where to place the faster CPU, the better disk array, and the new RAM. I've been an IT generalist for many years, so I know what the test designers are after, but when I wear my DBA hat, I want to put everything into SQL Server.

This seems kind of self-serving, but based on my experience, SQL Server tends to be the core or underlying technology for a lot of the business applications. A company that I worked at for a number of years relies on a single SQL Server for all financial data, logistics and materials tracking, SharePoint, and several other line-of-business applications. Without exception, these applications used SQL Server as a data store. Optimizing the server running SQL Server would have an immediate positive impact on a majority of the applications used for the key business activities, as well as many support applications.

There are four main subsystems that you need to optimize for SQL Server 2008 to perform optimally. These include the Processor, Memory, Storage, and Network subsystems. Performance of these subsystems will affect SQL Server in a variety of ways, and as part of the pre-installation process, you should have an understanding of what your hardware needs are. One quick note about the network subsystem is that it is often the one the DBA has the least control over, and yet sometimes has the most impact, depending on the number of applications and users that are being supported. You should work with your network administrators and engineers to plan a strategy for concurrent database access by your users.

Processor Considerations

Microsoft sets the minimum processor requirements at 1 GHz Pentium III or a compatible processor for 32-bit installations of SQL Server, and 1.4 GHz for 64-bit systems. However, 2.0 GHz is considered the recommended speed for both platforms. SQL Server uses the processor extensively during the compilation and execution of query plans. Your server can have an extraordinarily fast disk array and plenty of RAM, but if it has an underpowered processor, it is all for naught. As the workload of the server increases and more and more transactions are executed against it, the processor will have to schedule and handle the multitude of query execution plans and programmatic manipulation of data.

Chapter 10 discusses the ways to monitor SQL Server to ensure that the CPU is not a bottleneck, but from the outset, SQL Server should be given plenty of processor power. In addition, SQL Server is very adept at using multiple processors to execute parallel operations, so adding a second processor will often pay larger dividends than upgrading a single processor. However, if your license is per processor, the cost may be prohibitive to add additional processors.

As of this writing, Microsoft considers multiple logical processors to be covered under a single processor license. This would allow you to buy a quad-core CPU, essentially supplying SQL Server with up to four CPUs for the cost of a single processor license. For example, if you wanted to buy a new server that has two quad-core processors, you would be able to leverage all eight cores, but only have to buy two processor licenses.

Memory Considerations

The minimum amount of RAM, according to Microsoft, is 512 MB. I personally find this minimum requirement a bit on the ridiculous side. I wouldn't set up a Windows server running any multi-user application with only 512 MB of RAM, let alone a RAM-hungry application like SQL Server. Would 512 MB be sufficient for a desktop machine running SQL Server 2008 Developer Edition? Maybe, as long as no serious load was put on the server.

That's not to say that SQL Server wastes memory or that it consumes a bloated footprint. The simple fact is that SQL Server likes memory — *a lot*. It attempts to place as much data as possible in RAM so that the data is readily available for processing. It also tries to keep the data in RAM as long as possible.

SQL Server creates and maintains different memory pools for various database operations. For example, there is a *buffer cache* that is used to store data pages retrieved from the disk; a *procedure cache* that is used to store compiled stored procedures, triggers, functions, views, and query plans; and even a *log cache* for transaction log operations.

Having sufficient RAM on hand allows SQL Server to minimize the amount of page swapping required and enables the data to be pre-fetched for fast processing. If you want to keep SQL Server happy, feed it RAM. What you will get in return is a hard-working database server that efficiently and effectively utilizes that RAM to service your requests as fast as possible. Lack of sufficient RAM can also cause degradation in performance of the storage subsystem, as more data gets paged to disk.

Microsoft recommends just over 2 GB of RAM for both the 32-bit and 64-bit editions. Although Microsoft considers operating system overhead when publishing their recommended values, given the relatively low cost of RAM, I typically recommend this *above* the operating system requirements. For example, if Windows Server 2008 recommends 2 GB of RAM for the OS, I would recommend a total of 4 GB to help optimize performance.

Storage Considerations

An often overlooked hardware aspect of many SQL Server installations is the disk subsystem. I have personally witnessed deployments in which undertrained personnel installed the OS, SQL Server, and all the database files on the system partition. Although this will work, it is less than ideal. The question of how to best place the application and database files is answered with a definite ''It depends.''

If you're not familiar with the different levels of RAID technology, let me offer a quick primer. *RAID*, first of all, stands for ''Redundant Array of Inexpensive Disks'' (*inexpensive* being a relative term here). When working with SQL Server, there are four types of RAID implementations that are commonly used:

- **RAID 0** — RAID 0 offers no redundancy or fault tolerance, but instead helps improve performance by striping across multiple disks. RAID 0 also allows you to use the combined storage capacity of both disks. RAID 1, also known as *mirroring*, provides fault tolerance by making a bit-for-bit copy of your data on two disks. While this provides basic redundancy and can improve Read performance (by having two separate disks available to read from), you might suffer minor loss of Write performance, since the data will have to be written across both disks. RAID 1 has 50 percent storage overhead.
- **RAID 5** — RAID 5 is one of the more common implementation types of RAID, utilizing three or more disks. RAID 5 is also called *striping with parity*, because as it stripes across multiple disks, it writes a parity block on each stripe that allows the data to be rebuilt in case of a disk failure. RAID 5 is considered a good option for most scenarios because it provides fault tolerance and improved Read and Write performance and has a relatively low storage overhead. Because the available capacity on a RAID 5 array is $n - 1$ (n being the total number of disks in the array), the storage overhead decreases as the number of disks in the array increases.
- **RAID 10** — RAID 10 (also sometimes known as *RAID 1+0*) is the cat's pajamas of RAID, and is considered the optimal design solution for SQL Server database files. RAID 10 requires a minimum of four disks and essentially stripes data across two mirrored sets. So let's say, for example, that you have four disks: *a*, *b*, *c*, and *d*. Disks *a* and *b* will be used to make one mirrored set, which we'll call *ab*, and disks *c* and *d* will be used to make the *cd* mirrored set. The two mirrored sets are then part of a new striped set, so when data is written to the array, it is striped across *ab* and *cd*.

Now that you know a little bit more about the various RAID levels, it's important to understand that there are several factors that can have an impact on the decision regarding where to install everything. How important is fault tolerance? How much money is the organization willing to spend on the database solution? How much disk space will be needed? How busy is the existing disk system? An optimal installation of SQL Server could look something like Figure 2-1.

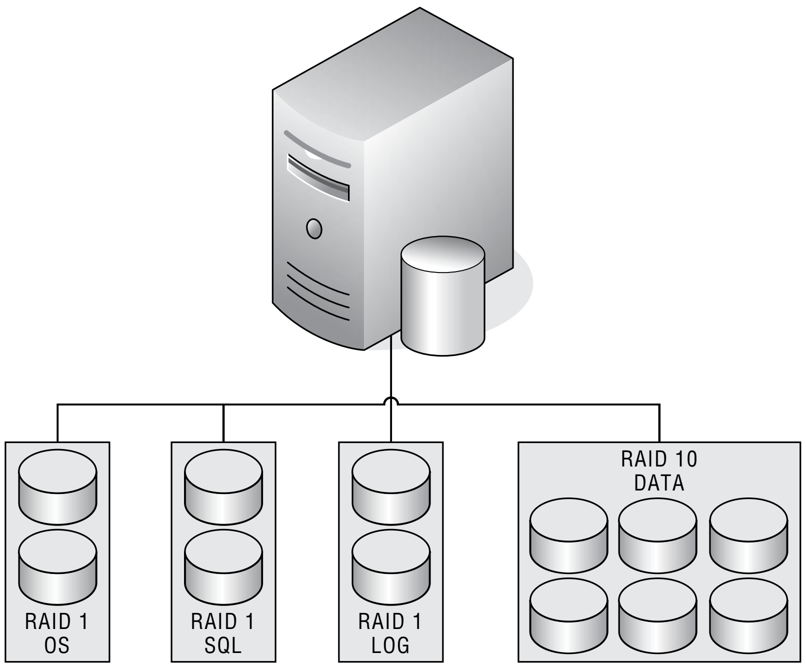


Figure 2-1: Optimal installation.

Notice that the application is installed on a separate set of spindles from the operating system. This reduces contention for disk resources and makes the application more efficient. Notice use of the term *spindle*. This is preferred to *drive* or *disk* because it leaves little room for interpretation. Physical disk drives have one spindle, which is loosely analogous with the center of a spinning top. Granted, the increase in capacity and general availability (as well as decreasing costs) of Solid State Drives, which have no spinning platter, may eventually make the term *spindle* obsolete. For now, let's agree to continue to use that term. In the case of Figure 2-1, the two spindles that host the log file on a RAID 1 array will actually look like a single drive to the operating system, when, in reality, there are two physical *disks*, or *spindles*.

In addition to the application existing on a separate set of spindles, the data files and the log files are on yet another set. The idea here is not to keep the hard disk industry in business, but to maximize efficiency, fault tolerance, and recoverability. Placing the operating system, application, and database all on the same spindle is basically putting all of your eggs in one basket. If the basket is dropped, you will lose all of your eggs. Likewise, if the spindle fails, you will lose your operating system, application, and databases. Your recovery time in this instance is tripled. Even if your server weren't to suffer a catastrophic failure, the amount of contention for resources on the disk subsystem could cause a severe degradation in performance.

Separating the database files from the transaction logs files can also help improve recovery efforts. If the database file is corrupted or damaged, the most recent backup can be used to recover it, and then the existing transaction log can be used to recover all the transactions since the last backup. Likewise, if the transaction log is lost, it can be re-created with minimal data loss from the database. If both data files and the log file are on the same spindle, a catastrophic failure of the spindle will result in all data since the last backup being lost.

Backup and recovery strategies are covered in more detail in Chapter 9.

The separation of the different components of SQL Server is just part of the equation. When choosing a disk system, it is also important to know what type of disk is best for each part of the database. Notice in Figure 2-1 that the operating system is installed on a RAID 1 array. The same goes for the SQL Server application and the database log file, while the data files are placed on a striped array. It is possible to place all the SQL resources on one or more RAID 10 or RAID 5 arrays, and many organizations do just that. However, when it comes to the transaction log, a RAID 1 configuration is more appropriate than a RAID 5 one. A transaction log placed on a striped array will actually decrease the performance of SQL Server. This is because of the inherent hit in Write performance on a RAID 5 array, and also because of the way SQL Server writes serialized data to the log. Log files, by their nature, are mostly written to, which means that often RAID 1 (or RAID 10, if you have the budget) is the best choice for performance. RAID 1 or RAID 10 is also better because of the sequential and serial nature of the transaction log as compared to the parallel friendly nature of data files.

Each transaction is written to the transaction log before it is written to memory. This puts the transaction log in the position to become a possible bottleneck. A fast array will help prevent the log from becoming a performance liability.

SAN and NAS versus Local Disk Storage

Another decision to be made during a SQL Server installation is that of storage architecture. There are many vendors in the marketplace with hundreds of possible configurations for sale. Many larger organizations have placed much of their corporate data on local and remote SANs. At the same time, other organizations have chosen NAS, and still others (mostly smaller organizations) have chosen to place all their data on local attached disk arrays. Although a complete discussion of these different technologies is beyond the scope of this book, a brief explanation is useful in describing the utilization of these technologies in database implementations.

Storage Area Network (SAN)

SANs typically transmit Small Computer Systems Interface (SCSI) block commands over a network (usually either Fibre Channel or iSCSI) in lieu of a SCSI connection for a direct-attached storage array. This option is well-suited to SQL Server because the database application expects block access to data, which is not easily supplied using NAS. Utilizing SAN software, multiple volumes can be created and ''presented'' to the servers, using the storage space on the SAN, as shown in Figure 2-2.

Network Attached Storage (NAS)

The NAS network interface is usually Gigabit Ethernet or Fast Ethernet, but the storage type is file-based via traditional file sharing protocols. Volumes are not presented to the servers that utilize a NAS; instead, files are accessed through Universal Naming Convention (UNC) shares, as shown in Figure 2-3. File-based access degrades SQL Server performance considerably, which is why NAS storage should be avoided. By default, databases cannot be created with a UNC location, but this behavior can be changed. However, if the database is going to be used for any serious I/O scenarios, you will find that NAS will not be able to provide an adequate response.

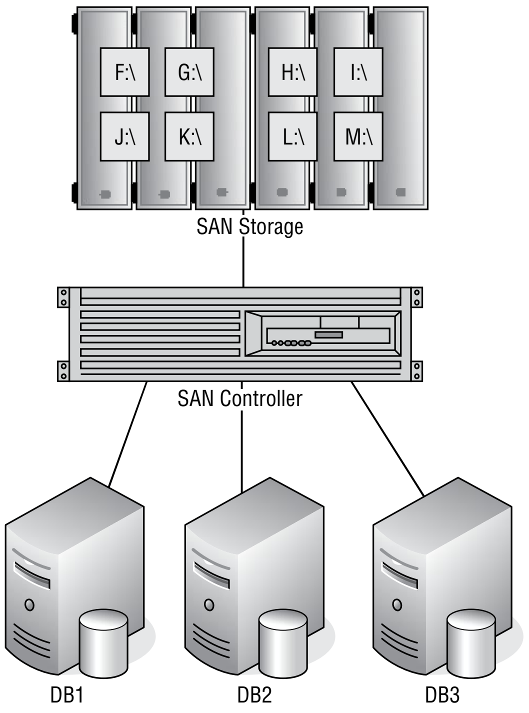


Figure 2-2: Storage Area Network.

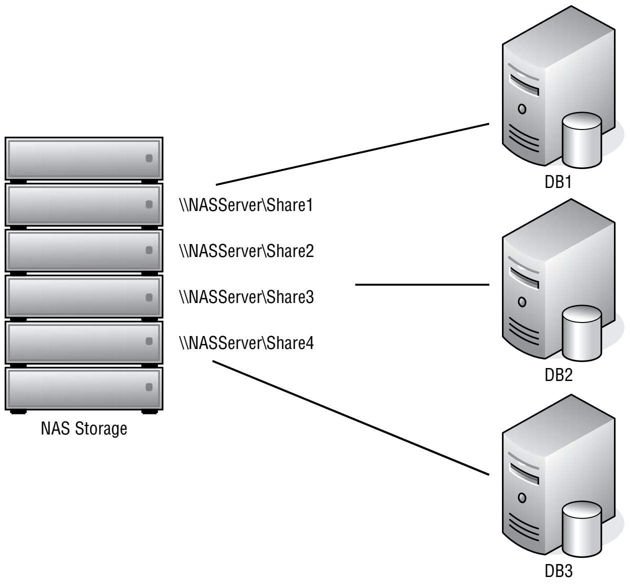


Figure 2-3: Network Attached Storage.

Local Attached Disk Array

There is a lot to be said for sharing storage resources among multiple servers on a high network, but some organizations (for a variety of reasons) have chosen to dedicate local attached storage to their database implementations (see Figure 2-4). In reality, the only differences between local attached disk arrays and SANs are that the volumes created on the local array are only accessible to the server to which the array is attached and that SAN controllers can optimize data transfer. Local arrays are typically connected via a high-speed SCSI cable or Fiber Channel.

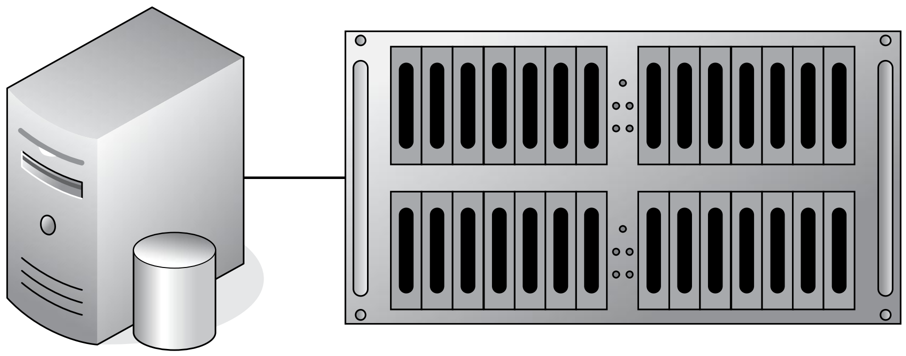


Figure 2-4: Local attached disk array.

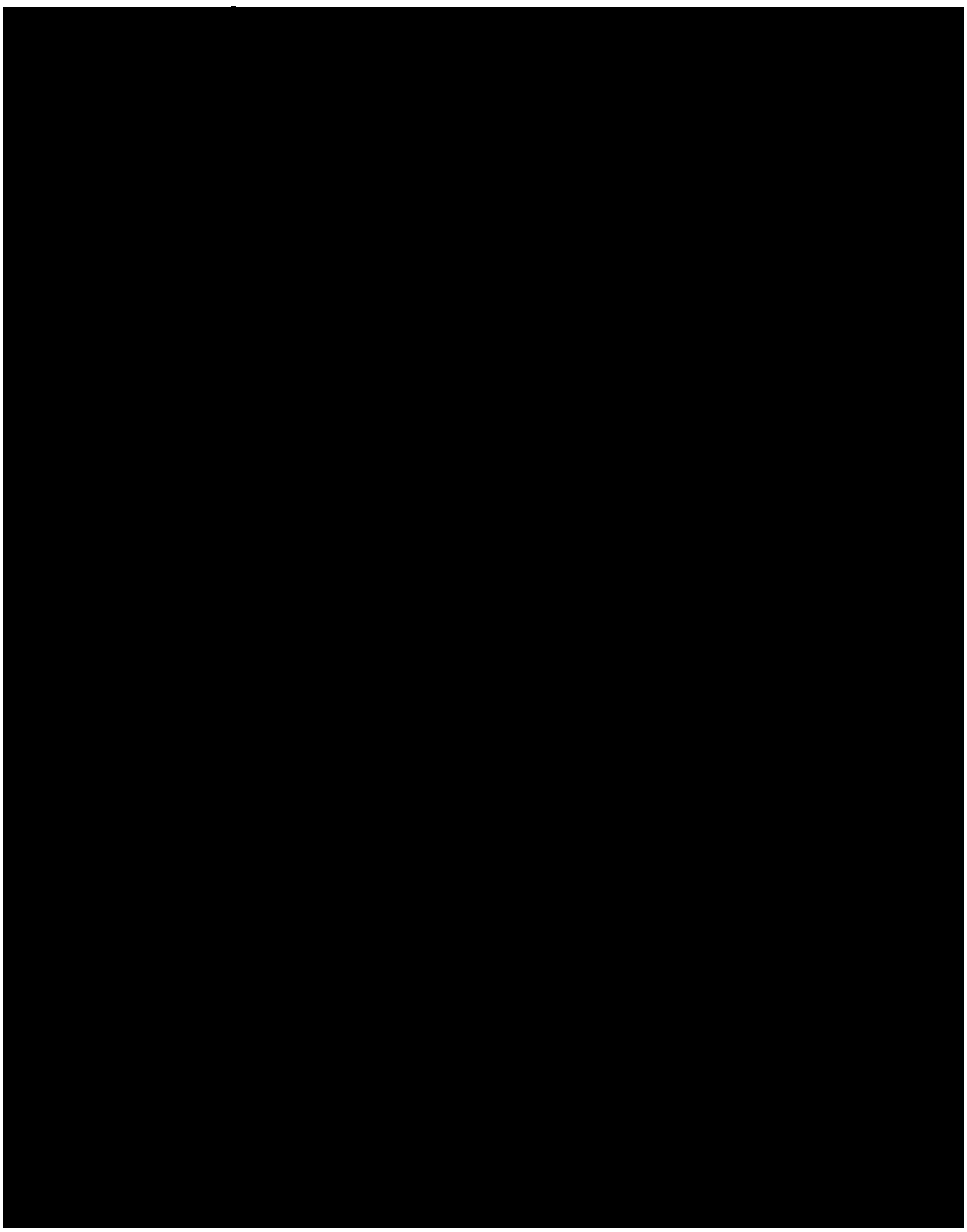
Virtualization Considerations

SQL Server 2008 is the first version of SQL Server that is supported in virtual environments; however, there are some limitations. Microsoft will officially only support installations of SQL Server in Hyper-V environments on Windows Server 2008, and clustering of virtual machines is not supported. Because of the continued improvement in virtualization technology, it is becoming a much more attractive option to companies that want to either consolidate hardware or take advantage of some of the recovery and portability options available. It's been my experience that the biggest bottleneck that occurs when running SQL Server in a virtual machine is I/O performance. For this, I strongly recommend using SAN storage for the database and transaction log files to avoid storing database information in a virtual hard drive file.

Software Prerequisites

In addition to the hardware dependencies mentioned above, there are a number of software dependencies that exist to support the various features of SQL Server 2008. The System Consistency Checker does a very thorough job of identifying all the requirements and dependencies, and informing you if anything is missing. For example, if a critical component is missing, the installer won't proceed until that component has been installed. If, however, you are running with less than recommended RAM, the SCC will give you a warning, but allow you to proceed with the installation. It is up to the DBA to evaluate the warning to ensure that it is acceptable to continue the installation.

Another critical dependency is the operating system. As you might expect, the IA86 and x64 editions of SQL Server 2008 can only be installed if the operating system is using the same platform. Note that 32-bit versions of SQL can be installed on 64-bit operating systems, but may actually suffer a performance loss because it will need to run within the WOW64. The following table describes the different operating systems required for each edition of SQL Server 2008. For a complete list of requirements, visit `http://technet.microsoft.com/en-us/library/ms143506.aspx`.



SQL Server Installation Center

The SQL Server 2008 setup process itself is pretty straightforward. If Autorun is enabled (I usually turn it off), the setup splash screen will launch as soon as you insert the media. If not, the installation can be launched from the `SETUP.EXE` file located in the root folder of the installation media.

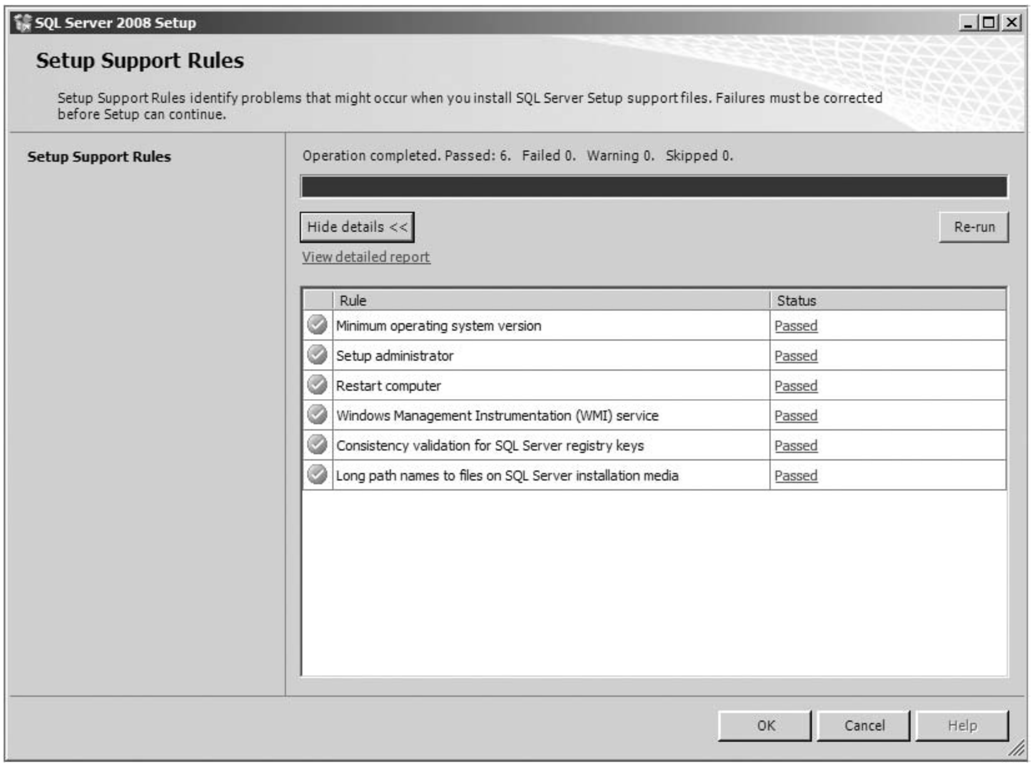
You may also note that there are three folders in the root folder of the SQL Server 2008 installation media. Each folder contains the platform-specific setup files for the x86, x64, and IA64 platforms. When you launch setup from the root folder, it runs a detection script to determine the platform of the current system and launches the installer for that platform. If you have a specific need to install, for example, the 32-bit version on a 64-bit platform, the preferred method is to select the Options page of the SQL Server Installation Center.

Before the setup application launches the SQL Server Installation Center, it checks for several dependencies that are critical to installing SQL Server. This includes an updated version of the Microsoft .NET Framework (version 3.5 SP1), and in some cases, an update to the Windows Installer service may be required as well. Be aware that if these components have not yet been installed, a reboot will be necessary before SQL Server setup can continue.

Once the dependent components can be installed, the SQL Server Installation Center menu pops up. From here, you can navigate through the different pages, to learn more about the planning and installation process. You can choose to run the System Configuration Checker manually, but all of the tests are run as part of the Installation Wizard for SQL Server 2008.

Setup Support Rules (for Setup Support Files)

Prior to installing the SQL Server setup support files, SQL Server checks a series of conditions to ensure that the support files can be installed before the actual setup process begins. The six items shown in Figure 2-5 and described in the following table are checked:



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: 6. Failed 0. Warning 0. Skipped 0.

Rule	Status
Minimum operating system version	Passed
Setup administrator	Passed
Restart computer	Passed
Windows Management Instrumentation (WMI) service	Passed
Consistency validation for SQL Server registry keys	Passed
Long path names to files on SQL Server installation media	Passed

Figure 2-5: Setup Support Rules results for setup files.

Once the initial validation tests have been completed and there are no errors that would halt the installation, the Registration Information screen appears and asks for your 25-character product key. After entering the product key, you will be presented with the License Terms to review and accept.

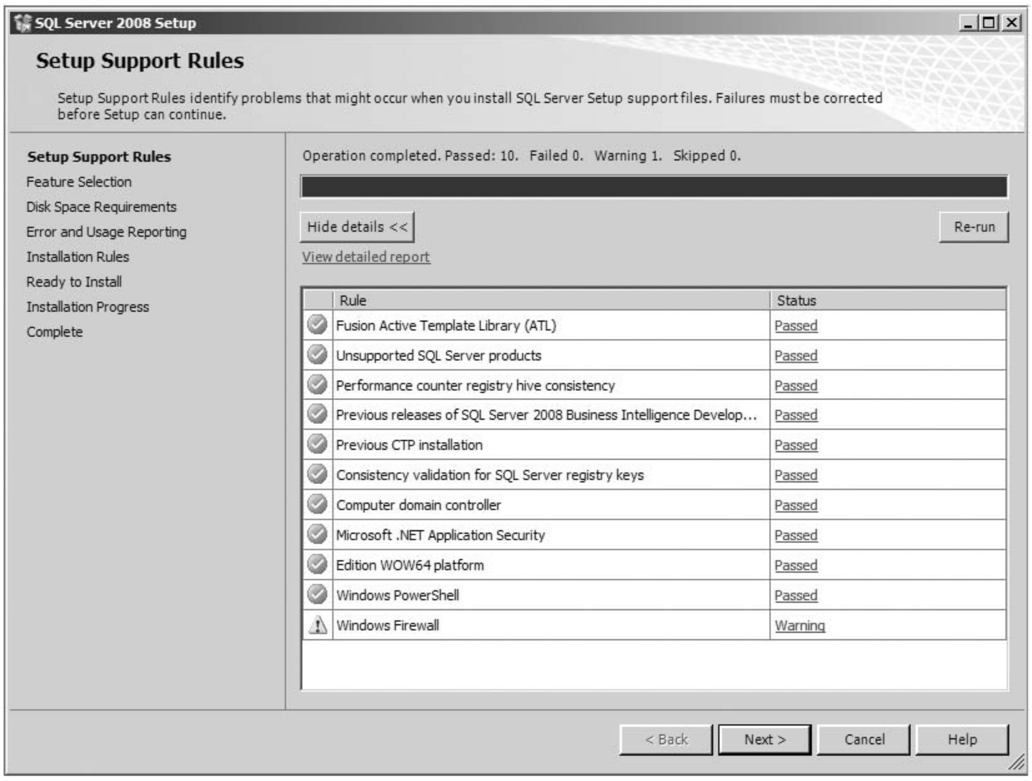
Before proceeding with installation, you should understand some of the licensing constraints around SQL Server 2008. Many organizations are not aware that the components of SQL Server are licensed as a bundle, and when you purchase a server or processor license for SQL Server, you can install some or all of those components on one machine, and one machine only. For example, if the Database Engine is installed on one server and the Reporting Services engine is installed on a different server, a separate license is required for each installation. This is a major area of confusion for many DBAs. Common sense would say that a purchase of a SQL Server license that included the Database Engine, Reporting Services, Integration Services, and Analysis Services would give an organization the right to spread these services across as many servers as necessary, as long as only one instance of each service was used. Common sense, in this instance, may get you into trouble with the licensing police. If you haven't read the licensing agreement, do so, or have your lawyer read it for you. The license agreement can also be found in the Resources section of the SQL Server Installation Center.

After accepting the terms of the license agreement, you will be prompted to install the SQL Server Setup files. These files are used during the installation process and are usually removed as part of the post-install cleanup.

Setup Support Rules (for Installation)

Another set of validation tests must be performed to verify that the system meets the conditions for installing SQL Server. The tested components are listed in the following table and shown in Figure 2-6:

[redacted]


SQL Server 2008 Setup

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 Develop...	Passed
✓	Previous CTP installation	Passed
✓	Consistency validation for SQL Server registry keys	Passed
✓	Computer domain controller	Passed
✓	Microsoft .NET Application Security	Passed
✓	Edition WOW64 platform	Passed
✓	Windows PowerShell	Passed
⚠	Windows Firewall	Warning

Figure 2-6: Setup Support Rules results for installation.

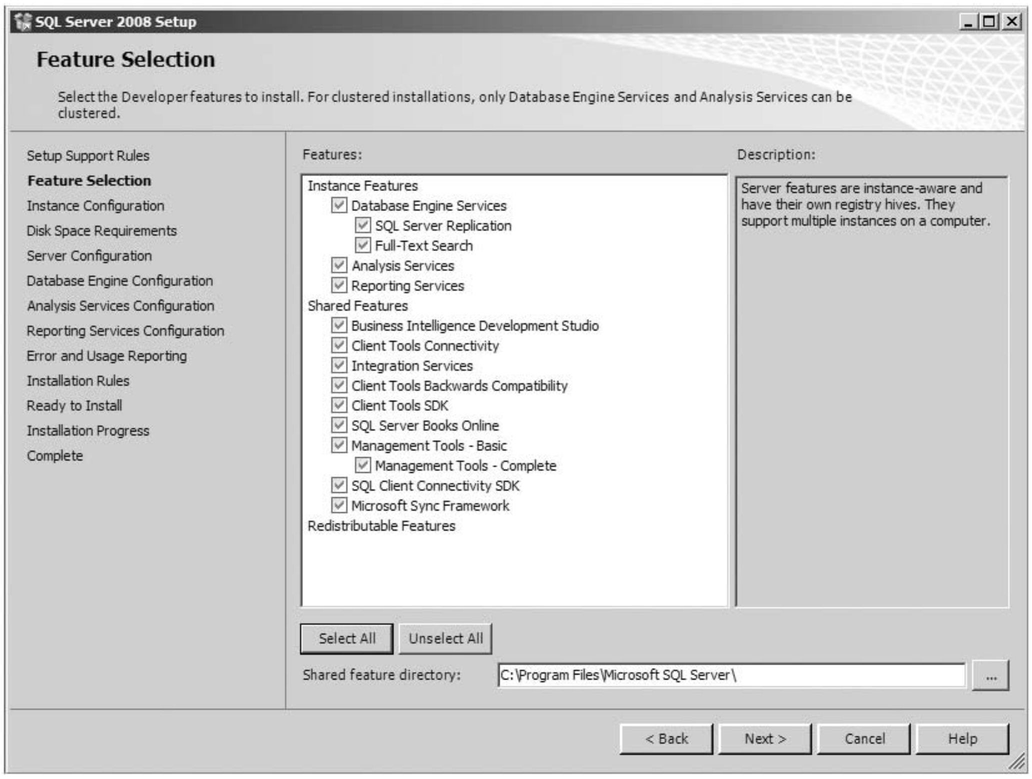
Feature Selection

The next step in the Installation Wizard is the Feature Selection screen (see Figure 2-7). This is where you will choose what aspects of SQL Server you want to install. If you intend to follow along with the examples in this book, it's recommended that you install all features in your test environment. In a production environment, you should install the features you intend to use, and no more. You can always go back and install additional services and features, but for the sake of efficiency, if you're not going to be using Analysis Services, there's no reason to install it. If you've installed an earlier version of SQL Server, sample databases were often included with the installation media. In the case of SQL Server 2005, installation of the sample databases was disabled, but it was still a feature that you could enable through the advanced installation options. With SQL Server 2008, the sample databases are not included with the media and are available online at `www.codeplex.com`. More information on installing the sample databases is covered later in this chapter.

Instance Configuration

After choosing what features of SQL Server are to be installed, the setup utility asks for instance information. You can install either a named instance or a default instance. The *default instance* takes on the name of the machine where SQL Server is being installed. There can be only one default instance; however, SQL Server 2008 Enterprise Edition supports installing up to 50 instances of SQL Server on a single machine.

If there is a default instance, a maximum of 49 named instances can be configured. If no default instance is installed, 50 named instances can be configured.



Feature Selection

Select the Developer 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
- Redistributable Features

Description:

Server features are instance-aware and have their own registry hives. They support multiple instances on a computer.

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

Figure 2-7: Feature Selection screen.

Named instances are referenced by the server name followed by the instance name. For example, the server name used in the examples for this book is *AughtEight*. The default name of the SQL Server 2008 installation is the same as the server name. However, you could install a named instance on AughtEight called *Dagobah*. To connect to the Dagobah instance of SQL Server, it must be referenced as *AughtEight\Dagobah*. In addition to the name, any client accessing a named instance must use the SQL connection objects from SQL Server 2000 or later. Legacy ODBC and old OLEDB drivers will be unable to enumerate a named instance of SQL Server 2008.

The Instance Configuration screen also provides you with the opportunity to change the default location for the SQL Server files. This sets the file location for the SQL binaries as well as the system databases. Best practices recommend that you separate the instance folders from the OS drive.

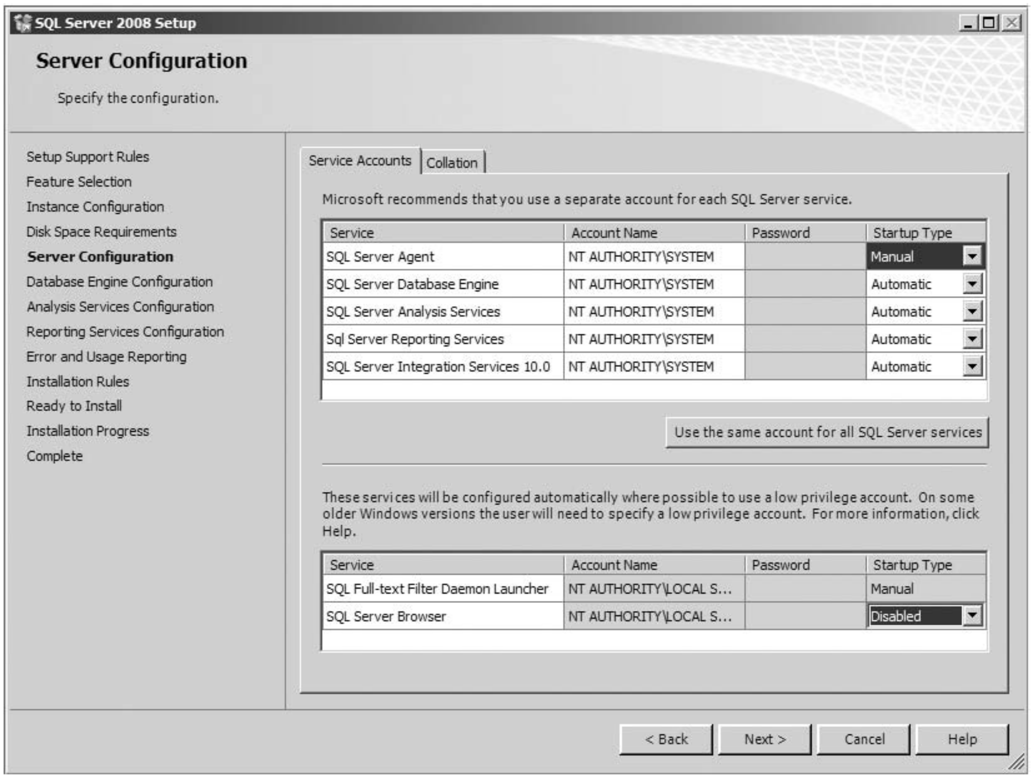
Server Configuration

After the instance configuration is completed and the Disk Space Requirements have been verified, the service accounts that SQL Server will use must be specified. Chapter 1 describes the various services that SQL Server may need to run depending on what features were installed. When configuring the security credentials for these services, you have a choice to make. Does the service require the ability to authenticate and connect to external resources? If so, the local system account will not be appropriate.

Best-practice security guidelines recommend that the local system account not be used because it grants full administrative access to the computer on which SQL Server is installed. This expands the attack surface of the system by allowing a compromised SQL Server to be used to attack other components on the system. Also, services running as the local system account will have no authenticated access to resources that exist on other servers in your environment.

A very useful feature of SQL Server is the ability to use the SQL Server Agent's scheduling options to run unattended jobs. If the ability to schedule SQL Server jobs that require access to external resources is desired, then at a minimum, the SQL Agent account will need to be configured to use a domain account so that the respective account can be granted permissions to the remote resource.

The ability to configure each installed SQL Server service individually is provided (see Figure 2-8), which is also a security best practice, but it does increase the administrative complexity of the system.



Server Configuration

Specify the configuration.

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

Service	Account Name	Password	Startup Type
SQL Server Agent	NT AUTHORITY\SYSTEM		Manual
SQL Server Database Engine	NT AUTHORITY\SYSTEM		Automatic
SQL Server Analysis Services	NT AUTHORITY\SYSTEM		Automatic
Sql Server Reporting Services	NT AUTHORITY\SYSTEM		Automatic
SQL Server Integration Services 10.0	NT AUTHORITY\SYSTEM		Automatic

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 Launcher	NT AUTHORITY\LOCAL S...		Manual
SQL Server Browser	NT AUTHORITY\LOCAL S...		Disabled

Figure 2-8: Service account screen.

In addition to the security information for each individual service, each service can be configured for automatic or manual startup during installation. By default, the SQL Agent Service is configured to start manually, while other installed components will start automatically. In order for scheduled tasks and jobs to execute, the SQL Agent Service must be running. It is usually a good practice to configure this service to run automatically.

Additionally, the SQL Server Browser Service is disabled by default. The Browser Service is only needed if you have multiple instances of SQL Server installed on the same machine. Although you cannot change the account being used by the Browser Service or the Full-Text Daemon Filter Launcher, these can be changed manually after SQL Server is installed.

Collation Settings

After setting the Authentication mode of SQL Server, you can configure the collation settings of SQL Server 2008 by selecting the Collation tab in the Server Configuration window. The first question many people have is, ''What is collation?'' The dictionary definition of *collation* is ''assembling in proper numerical or logical sequence.'' Collation settings have two significant effects on your database: the sorting of your character-based data and the searching of your character-based data.

A different collation can be set for both the SQL Server and Analysis Services, but Analysis Services only supports Windows collation, whereas SQL Server can support both Windows and SQL collation. SQL collation support is included for backward compatibility, and it is recommended to configure the server collation with Windows collation (despite the fact that SQL collation is configured as the default).

Choosing Windows collation by selecting the Collation Designator provides a greater level of control and more choices when it comes to customizing the collation settings for the server. The collation setting affects what data will be returned when searching on character data and in what order the data will be returned. It also determines what characters will be supported.

The default collation for an installation is determined by the locale with which Windows was configured. For example, the default collation the SQL Server installation application chooses when being installed on a Windows server configured for the United States is SQL_Latin1_General_CP1_CI_AS. A brief definition of this underscore-delimited name is definitely in order:

- `SQL_Latin1_General_CP1` indicates that characters from the Latin Code Page One (CP1), which is equivalent to the 1,252-character set, are supported. These characters provide support for the storing, sorting, and searching of character data in any Latin-derived language. These languages include Western European, English, and Latin American languages. However, it is important to note that sort orders can be different among Latin-derived languages. For example, in German the *ö* character comes before *z*, but in Swedish, the opposite is true (*z* comes before *ö*). Therefore, small discrepancies can occur from language to language.

    The number 1,252 represents the character set identifier as assigned by the International Organizations for Standardization (ISO).

- `CI` (Case Insensitive) indicates that the character data is to be sorted and searched in dictionary order without regard to capitalization. As this setting infers, there is also a `CS` (Case Sensitive) setting as well.
- `AS` (Accent Sensitive) indicates that the character data is to be sorted and searched in dictionary order with preference to accent marks. As a result, a search for a German ''spatlese'' wine will not return the correct spelling of this sweet late-harvest wine, which is *spätlese* if it is stored with the umlauts. Accent sensitivity can be turned off by specifying `AI` (Accent Insensitive).

These are not the only character settings that can be set. Character data can be set to be stored with sensitivity to width with the designation of `WS` (Width Sensitive) or `WI` (Width Insensitive). Width sensitivity applies to Unicode character data and differentiates between UTF-8 (8-Bit Unicode Text Format) and

UTF-16 (16-Bit Unicode Text Format). There is also a setting for Kana sensitivity: `KS` (Kana Sensitive) and `KI` (Kana Insensitive). *Kana sensitivity* essentially controls the sorting and searching of Asian Unicode characters (Japanese, Chinese, etc.) that can represent the same words using different script. For example, when Japanese kana characters Hiragana and Katakana are treated differently, it is called *Kana sensitive*; when they are treated the same, it is *Kana insensitive*.

Character data can also be sorted by their binary value. Binary sorting and searching is actually faster than dictionary sorting and searching, but is not as user-friendly. For example, the following script creates a table with two columns. The first column is assigned a character data type with case-sensitive dictionary collation. The second column is assigned a character data type with binary collation:

```
USE TempDB
CREATE TABLE MySortTable
(DictionarySort varchar(10) COLLATE Latin1_General_CS_AS NULL,
 BinarySort varchar(10) COLLATE Latin1_General_BIN)
GO
```

Once the tables are created, you can populate both of them with the same six rows: `Alpha`, `Bravo`, `Charlie` and `alpha`, `bravo`, `charlie` by executing the following command:

```
USE TempDB
INSERT MySortTable
 VALUES ('Alpha','Alpha')
INSERT MySortTable
 VALUES ('Bravo','Bravo')
INSERT MySortTable
 VALUES ('Charlie','Charlie')
INSERT MySortTable
 VALUES ('alpha','alpha')
INSERT MySortTable
 VALUES ('bravo','bravo')
INSERT MySortTable
 VALUES ('charlie','charlie')
GO
```

Now that the tables are created and populated, you can query them. Notice the different order of results using an identical query:

```
SELECT DictionarySort
FROM MySortTable
ORDER BY DictionarySort ASC

DictionarySort
--------------
alpha
Alpha
bravo
Bravo
charlie
Charlie

(6 row(s) affected)


SELECT BinarySort
```

```
FROM MySortTable
ORDER BY BinarySort ASC

BinarySort
----------
Alpha
Bravo
Charlie
alpha
bravo
charlie

(6 row(s) affected)
```

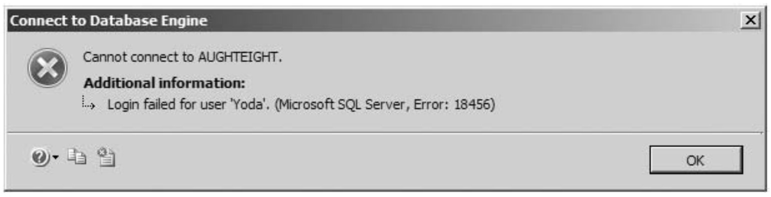
As you can see, server collation can have a profound effect on how your data is stored and retrieved, so careful planning is essential when deciding on a server collation. Fortunately, collation can also be set at the database and column level, so multiple collations are supportable.

[redacted]

Database Engine Configuration

After you have configured the server options, the next stage in the installation process requires you to set additional configuration properties on the Database Engine. It begins with the Account Provisioning screen, which allows you to set the Authentication mode and define administrative users. Authentication and security are covered in great detail in Chapter 6. However, a brief explanation is appropriate at this point.

If the default ''Windows Only'' configuration is chosen, only connections that have been authenticated by the local Windows security subsystem (or by the domain security subsystem) are allowed to be made to SQL Server. In this scenario, SQL Server validates that the login exists and has been authenticated, but no password verification takes place because SQL Server ''trusts'' that the login has been validated. A frequent connection error that occurs on servers configured for ''Windows Authentication mode'' is one that says simply that the login failed (see Figure 2-9).


Connect to Database Engine

Cannot connect to AUGHTEIGHT.

Additional information:

Login failed for user 'Yoda'. (Microsoft SQL Server, Error: 18456)

Figure 2-9: Bad login or password error message.

This is admittedly a vague response to a login request and is not the most intuitive message in the world. ''Login Failed because it is not a valid Windows account and the server is configured for Windows authentication mode'' or something a bit more informative would have been more useful. The message can be even more cryptic, given that the respective SQL login may, in fact, exist. Being in ''Windows Authentication mode'' does not prevent the database administrator from creating SQL Server login accounts. However, any attempt to connect with a valid SQL Server login when the server is in ''Windows Authentication mode'' will result in the vague ''trusted SQL Server connection'' error.

With ''Mixed Mode,'' SQL Server can authenticate Windows logins as well as logins that have been created locally on the SQL Server. Local SQL Server logins are validated by username and password verification. The username and an encrypted version of the password are stored in the `master` database. When a SQL login connection is requested, the SQL Server security subsystem encrypts the provided password, compares it to the stored password, and allows or refuses the connection based on the credentials provided.

If you choose the ''Mixed Mode'' option, you will need to specify a password for the `sa` account. Not setting one, or setting a weak one, would expose your SQL Server to any number of potentially disastrous results.

Invalid credentials (either bad login name or bad password) result in the same ''Login failed'' message (see Figure 2-9) when SQL Server is configured for ''Mixed Mode'' security.

Also on this screen you will need to provide at least one Administrator account. Often, you will want to choose the ''Add Current User'' option (to give yourself rights to the SQL Server), but you may need to include additional users or groups as well. Most common production environments will require you to add a group that identifies all the SQL Server administrators, which can be added through this tool.

The Data Directories tab allows you to change the default locations for data files (which will also change the default location for the system databases, user databases, the `tempdb` database, and the backup directory).

The last tab in the Database Engine Configuration screen allows you to enable FILESTREAM options. FILESTREAM is turned off by default, and if you don't enable it from here, you can enable and configure it using SQL Server Configuration Manager and SQL Server Management Studio. More information about using FILESTREAM is in Chapter 5.

Analysis Services Configuration

As with the Database Engine, the Analysis Services Engine will require you to specify which users or groups will have administrative control over the Analysis Services instance, as well as the data directories for data, log, temp, and backup files. Note that Analysis Services does not use SQL-based logins. All authentications are done through the Windows Authentication provider.

Reporting Services Configuration

If you are using SQL Server Reporting Services, you may want to specify how reports will be published. As mentioned in Chapter 1, SQL Server Reporting Services no longer uses Internet Information Services (IIS) for hosting access to the Reporting Services Web Service and the Reports virtual directory (both of which are covered in more detail in Chapter 18). In your production environment, you should have already decided whether reports will be published natively from SQL Server, or if they are going to be published on a SharePoint Server. You have the option during installation to configure Reporting Services to use the default ''Native Mode'' configuration or to use ''SharePoint Integrated Mode.'' There is also a third option that allows you to install the files needed for the SSRS, but configuration will be done using the Reporting Services Configuration tool after the installation has completed.

Error and Usage Reporting

Microsoft also provides an opt-in screen to allow you to send Windows and SQL error reports, as well as feature usage data, to Microsoft. Personally, I see some value in this, as it helps Microsoft identify problems or bugs. That being said, I enable this only on my test and development systems and never in my production systems, unless there is a corporate policy that dictates otherwise. Microsoft Enterprise licensing customers can send error reports to a Corporate Error Reporting server that allows your administrators to selectively choose which events get sent to Microsoft.

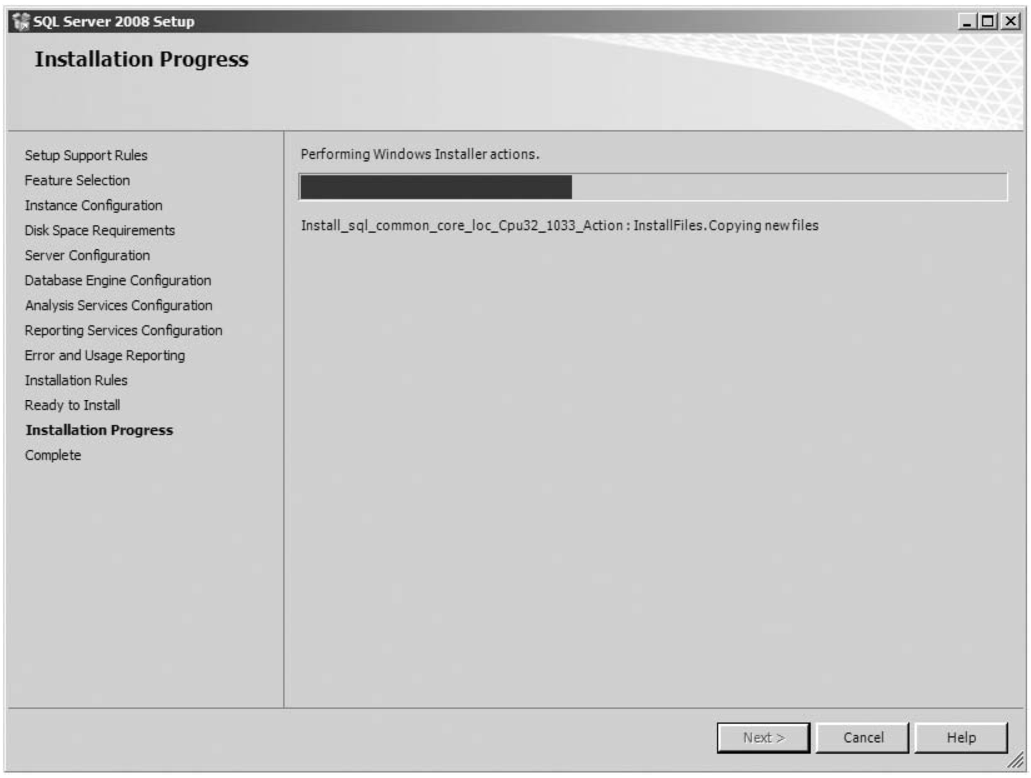
Installation Rules

Prior to finally installing the files for SQL Server, one last set of rules is checked. These rules validate your setup configuration options to identify potential problems that would cause undesired behavior or prevent SQL Server from installing. The following table lists the components that are checked before installation begins:

[redacted]

Final Steps

After the Install Rules have been validated, a final summary screen appears that provides you with a list of the services and features that will be installed. Clicking ''Install'' launches the SQL Server installation, and an Installation Progress screen appears (see Figure 2-10). The Installation Progress screen gives summary information about all the different features required by SQL Server and shows when each individual feature is finished installing.



Performing Windows Installer actions.

Install_sql_common_core_loc_Cpu32_1033_Action : InstallFiles.Copying new files

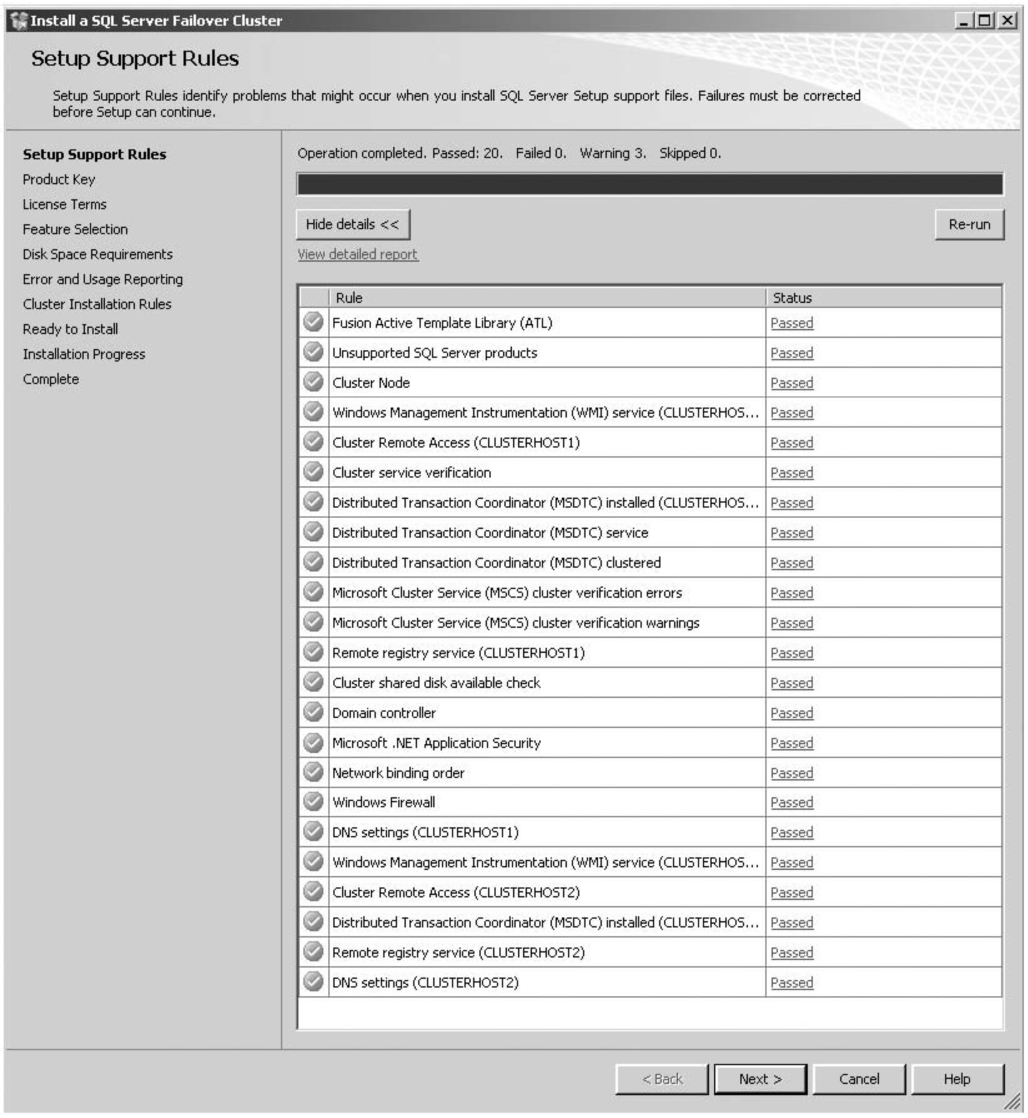
Figure 2-10: Installation Progress screen.

Installing to a Windows Cluster

The most difficult part about installing SQL Server to a cluster is configuring the Windows cluster, which is beyond the scope of this book. It is important to note that planning and configuration of the cluster must be done prior to running SQL Server Setup. There are several dependencies that must be in place, such as clustering the Microsoft Distributed Transaction Coordinator (MS DTC). Once the Windows cluster is installed and configured, the installation of SQL Server to the cluster has some very significant differences from installing to a single server. One of the first things you will most likely notice is that when the pre-installation rule validation process runs, it detects all nodes in the cluster and ensures that they meet the requirements for a SQL Server install (see Figure 2-11).

Because you are installing a SQL Server failover cluster, the installation will be slightly different from that previously described for a single server installation. After choosing to install the failover cluster, the

Instance Configuration screen appears. SQL Server 2008 supports multiple instances in a cluster, as well as in stand-alone scenarios.



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

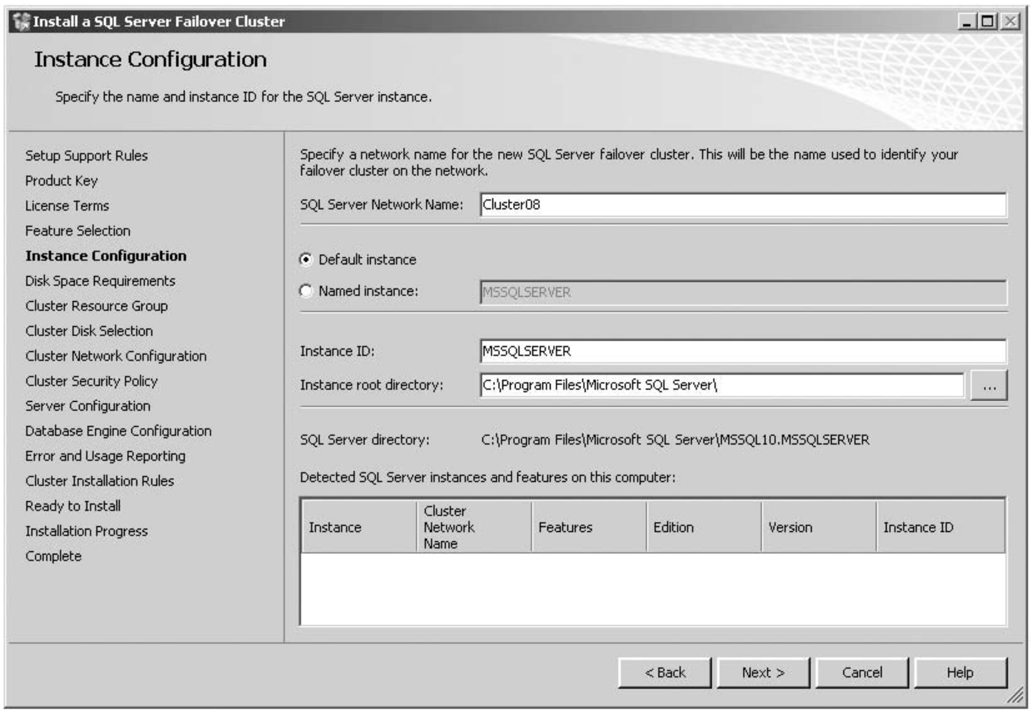
Operation completed. Passed: 20. Failed 0. Warning 3. Skipped 0.

Rule	Status
Fusion Active Template Library (ATL)	Passed
Unsupported SQL Server products	Passed
Cluster Node	Passed
Windows Management Instrumentation (WMI) service (CLUSTERHOS...	Passed
Cluster Remote Access (CLUSTERHOST1)	Passed
Cluster service verification	Passed
Distributed Transaction Coordinator (MSDTC) installed (CLUSTERHOS...	Passed
Distributed Transaction Coordinator (MSDTC) service	Passed
Distributed Transaction Coordinator (MSDTC) clustered	Passed
Microsoft Cluster Service (MSCS) cluster verification errors	Passed
Microsoft Cluster Service (MSCS) cluster verification warnings	Passed
Remote registry service (CLUSTERHOST1)	Passed
Cluster shared disk available check	Passed
Domain controller	Passed
Microsoft .NET Application Security	Passed
Network binding order	Passed
Windows Firewall	Passed
DNS settings (CLUSTERHOST1)	Passed
Windows Management Instrumentation (WMI) service (CLUSTERHOS...	Passed
Cluster Remote Access (CLUSTERHOST2)	Passed
Distributed Transaction Coordinator (MSDTC) installed (CLUSTERHOS...	Passed
Remote registry service (CLUSTERHOST2)	Passed
DNS settings (CLUSTERHOST2)	Passed

Figure 2-11: Ensuring that the requirements are met for the install.

Configuring the Virtual Server Name

The least-intuitive part of installing a SQL Server failover cluster is the naming configuration. When the Windows cluster was originally installed, a virtual name was designated for the cluster. However, a virtual name must also be specified for the SQL Server installation, and it cannot be the same as the virtual name used for the cluster. For my test cluster, I installed Windows Server 2008 Enterprise Edition on two Virtual PC images and configured the two servers as nodes in a Windows failover cluster. During the SQL Server installation, the setup utility will prompt for the instance configuration information, at which time you can supply both the SQL Server Network Name, which is the name of the SQL Server cluster, and the instance name (Figure 2-12).



Specify a network name for the new SQL Server failover cluster. This will be the name used to identify your failover cluster on the network.

SQL Server Network Name: Cluster08

(•) 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

Detected SQL Server instances and features on this computer:

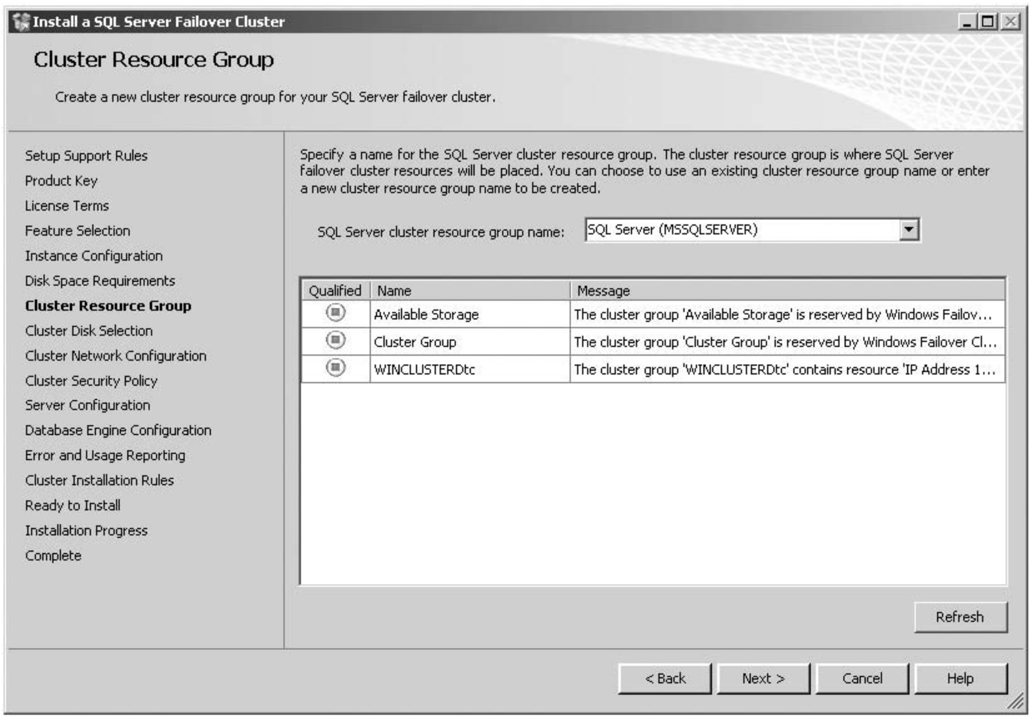
Instance	Cluster Network Name	Features	Edition	Version	Instance ID

Figure 2-12: Instance Configuration screen.

If you choose a default instance, the name of the SQL Server will be whatever you provide for the SQL Server Network Name. If you choose a named instance, the instance name will be NetworkName\InstanceName.

After specifying the Network Name, a cluster resource group must be created. The resource group is where SQL Server places all failover cluster resources. You will also need to specify the shared disk (or disks) that will be used to store shared SQL Server data (Figure 2-13). Additionally, you will need to designate an IP address that will be used as a listener for the SQL Server cluster.

The last cluster-specific option applies a cluster security policy for services that are installed as part of the cluster. Windows Server 2003 and Windows XP only support the use of domain groups for clustered services. This meant that the service accounts for each SQL Service installed as part of this cluster would be added to the identified domain groups. Windows Vista and Windows Server 2008 include a new feature that uses a Service SID rather than a group (Figure 2-14). This improves security by not requiring the service account to run with unnecessary elevated privileges.


Install a SQL Server Failover Cluster

Cluster Resource Group

Create a new cluster resource group for your SQL Server failover cluster.

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Cluster Resource Group
Cluster Disk Selection
Cluster Network Configuration
Cluster Security Policy
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Cluster Installation Rules
Ready to Install
Installation Progress
Complete

Specify a name for the SQL Server cluster resource group. The cluster resource group is where SQL Server failover cluster resources will be placed. You can choose to use an existing cluster resource group name or enter a new cluster resource group name to be created.

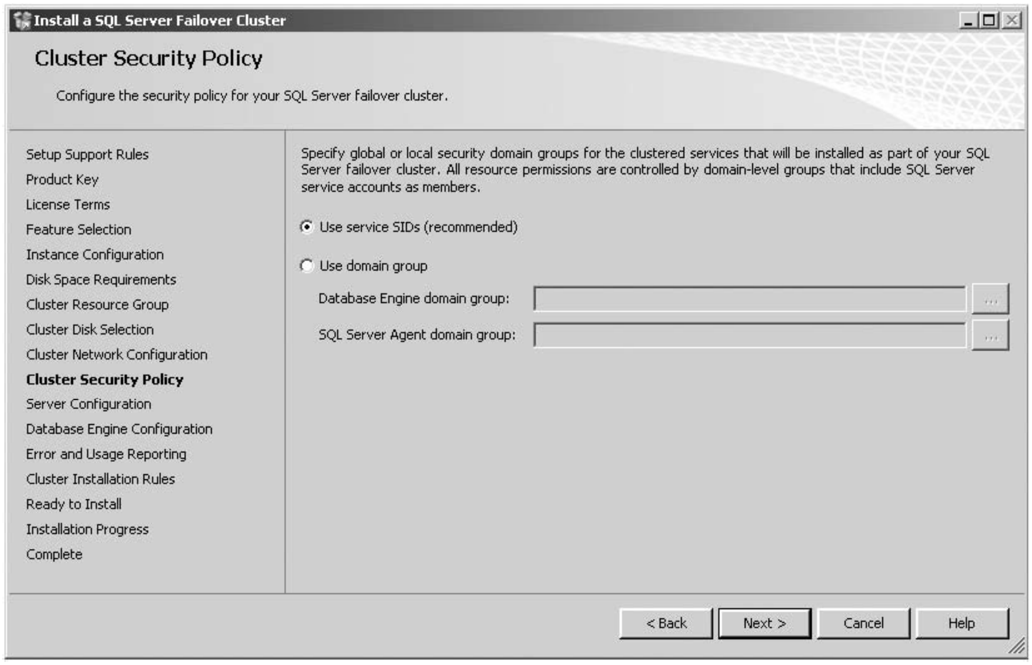
SQL Server cluster resource group name: SQL Server (MSSQLSERVER)

Qualified	Name	Message
⊜	Available Storage	The cluster group 'Available Storage' is reserved by Windows Failov...
⊜	Cluster Group	The cluster group 'Cluster Group' is reserved by Windows Failover Cl...
⊜	WINCLUSTERDtc	The cluster group 'WINCLUSTERDtc' contains resource 'IP Address 1...

Refresh

< Back Next > Cancel Help

Figure 2-13: Cluster Resource Group screen.


Install a SQL Server Failover Cluster

Cluster Security Policy

Configure the security policy for your SQL Server failover cluster.

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Cluster Resource Group
Cluster Disk Selection
Cluster Network Configuration
Cluster Security Policy
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Cluster Installation Rules
Ready to Install
Installation Progress
Complete

Specify global or local security domain groups for the clustered services that will be installed as part of your SQL Server failover cluster. All resource permissions are controlled by domain-level groups that include SQL Server service accounts as members.

[✓ Use service SIDs (recommended)]

Use domain group

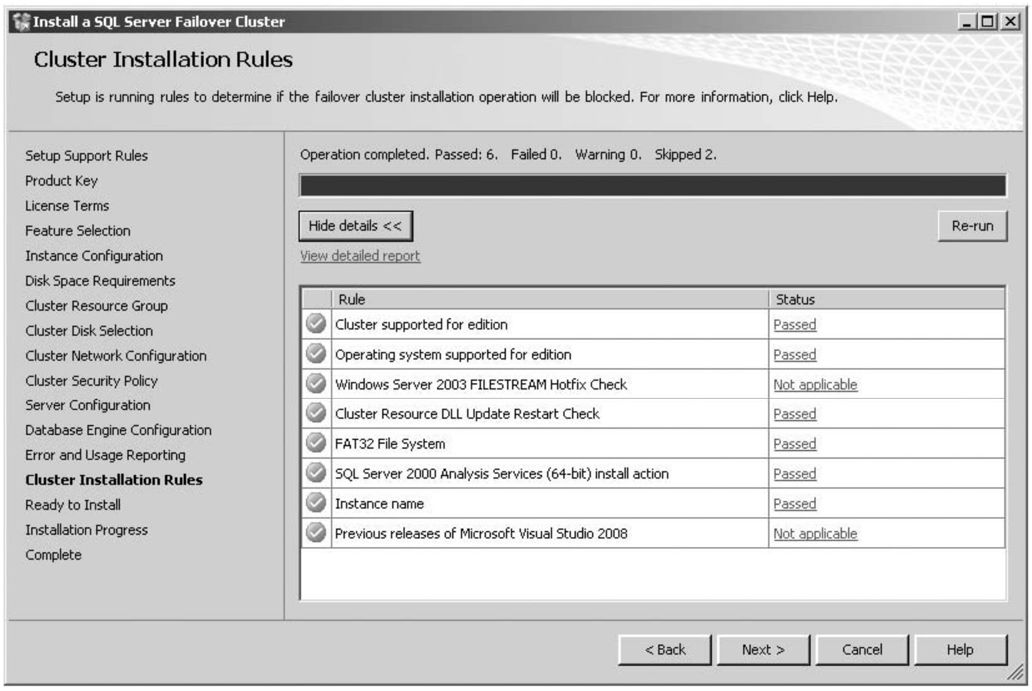
Database Engine domain group:

SQL Server Agent domain group:

< Back Next > Cancel Help

Figure 2-14: Cluster Security Policy screen.

The Cluster Security Policy configuration screen is the last dialog that is different from a stand-alone installation. The summary screen (Figure 2-15) is presented after the services screen, and then the installation begins.


Install a SQL Server Failover Cluster

Cluster Installation Rules

Setup is running rules to determine if the failover cluster installation operation will be blocked. For more information, click Help.

Operation completed. Passed: 6. Failed 0. Warning 0. Skipped 2.

Rule	Status
Cluster supported for edition	Passed
Operating system supported for edition	Passed
Windows Server 2003 FILESTREAM Hotfix Check	Not applicable
Cluster Resource DLL Update Restart Check	Passed
FAT32 File System	Passed
SQL Server 2000 Analysis Services (64-bit) install action	Passed
Instance name	Passed
Previous releases of Microsoft Visual Studio 2008	Not applicable

Figure 2-15: Cluster Installation Rules.

Once SQL Server is successfully installed, it can be controlled just like any other SQL Server instance. The only difference is the ability of SQL Server to fail over to the second node automatically in the case of fault tolerance, or manually for scheduled maintenance events.

Sample Databases

Because SQL Server no longer ships with sample databases, in order to follow along with many of the examples in the book, you will need to download and manually install the sample databases from Mirosoft's CodePlex site. There are three databases available that can be found at `www.codeplex.com/MSFTDBProdSamples`. These include the AdventureWorks2008, AdventureWorksLT2008, and AdventureWorksDW2008 databases. It is important to note that these are not the same databases that shipped with SQL Server 2005. Although they may look similar on the surface, they have a different schema, and they have been optimized for SQL Server 2008. Each of these comes in a platform-specific version (x86, x64, ia64) and gives you several options for download types (.zip and .msi). I prefer the MSI installer myself, as it makes it easy to download and deploy. There is also an installer for some sample scripts that are used in conjunction with the databases.

In most cases, you'll be using the AdventureWorks 2008 OLTP database, but the Analysis Services chapter uses the AdventureWorks DW database.

Installation Review

Not every installation goes flawlessly, and not everything may behave as expected. After installing SQL Server, it is important to review the installation to ''inspect,'' or ensure that what you expected to happen, actually happened. Did the services get installed with the proper credentials? Are they configured to auto-start? Are the program files and database files where they were expected to be? This may seem to be a little overkill, but ''An ounce of prevention is better than a pound of cure.''

Summary

Careful planning prior to the installation process prevents the need to uninstall and start over. It also prevents a continuous struggle with an installation that is ''not quite right.'' In later chapters, the optimization process, disk and memory access, as well as disaster recovery are discussed in great detail, and the connection to a well-designed infrastructure for installation will become increasingly evident. One of the most important aspects of installing SQL Server is having an understanding of the effects of the options you select. Too many times I've seen production environments where a junior administrator who was given the install media and no direction installed every feature under the sun. This was wasteful and unnecessary.

This chapter described physical storage options, which are a big part of any database configuration. By placing SQL data files and log files on separate physical disks, you decrease the chances of a major disaster and increase the speed of recovery. By placing SQL Server's assets on separate controllers and arrays, you also increase the performance of SQL Server by reducing resource conflicts and maximizing database throughput. It's always a balancing act to try to get the most out of your system performance while staying within a reasonable budget.

When it comes to availability, understand that Microsoft worked very hard to make SQL Server ''cluster-friendly.'' It is fairly easy to configure a failover cluster, but remember that a full discussion of the Windows cluster was not provided. Many resources are available in the Windows Help files, online resources such as TechNet, and in print that cover the topic of clustering in great detail. It is strongly recommended that you research them thoroughly prior to any SQL Server cluster installation.

Chapter 3 will introduce you to the tools used to administer, manage, monitor, and maintain SQL Server 2008. You will learn about a number of improvements that have been made to facilitate the administration of SQL Server 2008.