Exam
Manual

LearnSmart

Microsoft (70-450) SQL Server 2008 Design, Optimize and Maintain

If you're looking to boost your job-market potential, this LearnSmart exam manual will help you prepare for the SQL Server 2008 Design, Optimize and Maintain exam (70-450). By studying this manual, you will become familiar with a plethora of exam-related topics, including:

- Designing a SQL Server Instance and a Database Solution
- Designing a Database Server Security Solution
- Designing a Backup and Recovery Solution
- And more!

Give yourself the competitive edge necessary to further your career as an IT professional and purchase this exam manual today!

Designing, Optimizing and Maintaining a Database Administrative Solution Using Microsoft SQL Server 2008 (70-450) LearnSmart Exam Manual

Product ID: 12378
Production Date: July 8, 2011

Warning and Disclaimer

Every effort has been made to make this document as complete and as accurate as possible, but no warranty or fitness is implied. The publisher and authors assume no responsibility for errors or omissions. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this document.

Volume, Corporate, and Educational Sales

PrepLogic offers favorable discounts on all products when ordered in quantity. For more information, please contact PrepLogic directly:

1-800-418-6789
solutions@learnsmartsystems.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Abstract

This Exam Manual covers all of the knowledge, concepts and procedures you need to master in order to take and pass the Microsoft SQL Server 2008, Designing, Optimizing and Maintaining a Database Administrative Solution (70-450) exam. We will discuss automating and maintaining administrative tasks, the Transact SQL statements associated with administering a SQL Server 2008 solution and the all important design aspects of implementing SQL Server 2008 in your enterprise environment.

What to Know

This exam is designed to test the candidate's knowledge in all aspects of database design, optimization and maintenance at the administrative level in SQL Server 2008. Included are topics like database instance design, instance authentication and security, backup and recovery, security and monitoring. The official objectives for this exam are:

- Designing a SQL Server Instance and a Database Solution (14 percent)
- Designing a Database Server Security Solution (15 percent)
- Designing a Database Solution for High Availability (15 percent)
- Designing a Backup and Recovery Solution (20 percent)
- Designing a Monitoring Strategy (13 percent)
- Designing a Strategy to Maintain and Manage Databases (14 percent)
- Designing a Strategy for Data Distribution (9 percent)

Please note that the percentages attached to each objective, provided by Microsoft, are meant to serve as a general guide for how much test content goes with each objective. A successful candidate should know each objective intimately, both from study and personal experience.

Tips

It is important to ensure that you have hands-on experience with SQL Server 2008 and can attempt the syntax of queries demonstrated in this manual and the configuration examples. The availability of so many virtual server/PC type products today and Microsoft's 180-day free trial of SQL Server 2008 should allow plenty of chances to work with the product prior to taking the exam.

Further, it's important that the successful candidate fully understands T-SQL statements, best practices for design and implementation and the hardware requirements for running SQL Server 2008 in an enterprise environment.

Domain 1: Designing a SQL Server Instance and a Database Solution

Designing for CPU, Memory and Storage Capacity Requirements

When designing a SQL server instance consideration must be given to the resources that will be utilized in that instance. What we will discuss in this section will be how to allocate those resources, how to design and configure a server instance, and how to design the physical database. And in addition to the server and database, you will learn how to design a migration, consolidation, and upgrade strategy.

CPU Requirements

SQL Server 2008 requires a processor with a minimum speed of 1 GHz, but like most of Microsoft's requirements you really should use the recommended 2 GHz processor. This depends somewhat on the OS but more is always better in the case of CPU speed.

When selecting a CPU, both 32 bit and 64 bit CPUs are supported by SQL Server 2008. There are, however, some benefits to using the 64 bit architecture over the x86 architecture. We'll list those, below:

- Up to 8 TB of memory is supported
- Better bus architecture which equals higher speed and better throughput
- Larger on-chip cache (L2 cache)
- Better chip cache management

Another processor option is a *multicore processor*. A *multicore processor* contains two or more complete processors on the same chip. Each processor runs as an independent CPU with its own L2 cache. A main advantage of using multicore processors is *hyperthreading*. This allows multiple threads to run simultaneously on a single CPU and each processor keeps the state for each thread.

To identify CPU performance problems use the SQL Server plan cache counters. These counters are available using the performance monitor (we will discuss later in Domain 5 of this manual) - and include the following statistics:

- Cache hits ratio
- Number of batch requests
- Number of compilations per second
- Number of recompilations per second

Memory Considerations

32bit architecture has some limitations when it comes to memory usage. For instance, you can only access 4GB of memory directly and only 2GB is available for SQL Server with another 2GB for the OS. However you can up this by using the /3GB switch in boot.ini to allow 3GB of memory per process. This leaves 1GB available for the OS.

You can also use the **AWE** (Address Windows Extension) feature to use high memory for data buffers. Using AWE however provides no benefit for stored procedure cache, some CLR functions or lock arrays. 64bit architecture provides 8GB of directly accessible memory with a full 7GB available to SQL Server - no switches activated. There is still 1GB available for the OS.

Note: Although SQL Server 2008 can access huge amounts of memory, the amount of actual memory available to SQL Server may be limited by the OS.

NUMA

When there are many processors in a single machine with one memory controller, performance can be limited because all of the processors are competing for access to memory through a single hardware channel. If you could divide the processors into groups that each have their own memory controllers -- to the point where each processor is able to access memory without competition -- you have maximum performance in regards to memory access.

That's exactly what **NUMA** (Non-uniform Memory Access) can do for you. As more and more processors became the norm in server design, the need for a hardware solution to the problem of overloaded memory controllers increased. NUMA was implemented to meet that need.

NUMA hardware divides multiple processors into small groups of processors (NUMA nodes) that each have their own memory and sometimes their own I/O controllers. This has increased performance by eliminating the memory access bottleneck in the older systems.

SQL Server is NUMA aware and works just fine on systems with NUMA designed into the hardware, without any changes. **NUMA** is a scalable solution to the limitations of **SMP** (Symmetric Multiprocessing) architecture.

The main advantage of NUMA is scalability. The task of memory access is broken down into groups of CPUs with shared memory rather than available memory being shared by all CPUs in the server. Bottlenecks are eliminated by limiting the number of CPUs on any one memory bus and providing access via a high speed interconnection on each node.

There is another consideration when using NUMA -- That is the difference in local and foreign memory. Local memory is the memory that is running on the same node as the CPU currently running the thread. Foreign memory is memory that is not running on the current thread. Since NUMA uses both local and foreign memory it will take longer to access some memory than others.

NUMA Ratio is the ratio of foreign memory access time to local memory access time. Ideally this ratio should be '1' – this indicates that it takes the same time to access foreign memory as it does local memory. A ratio of '2' indicates that it takes twice the time to access foreign memory as it does the local memory – and so forth.

Some Windows applications and SQL Server 2003 and earlier, are not NUMA aware and will sometimes perform poorly with NUMA hardware. The NUMA configuration is recorded in the server log as multimode configuration along with CPU mask when the server is started.

SQL Server 2008 supports both hardware NUMA and soft NUMA.

Hardware NUMA

Computers with hardware NUMA have more than one system bus. The hardware determines the number of CPUs that are served by a single bus and each group of CPUs has its own memory as well as its own bus. A hardware NUMA node is a collection of CPUs, memory bus, and I/O channels.

If you already have the hardware and you're not sure how many NUMA nodes you have, you can find out by executing the query:

SELECT DISTINCT memory_node_id FROM sys.dm_os_memory_clerks

If this query returns node 0 only, either you don't have hardware NUMA or your NUMA is configured for interleaved memory.

Software NUMA

With SQL Server you can group CPUs into nodes with software as well. This grouping is known as soft-NUMA. Soft-NUMA is used when you have many CPUs but you have no hardware NUMA or if you have a need to divide the hardware nodes into smaller groups (nodes).

You do this by creating a registry entry that describes your Soft NUMA setup.

Soft NUMA does not affect memory nodes, only CPU nodes. In other words, with soft NUMA you can only group processors, but you have no control over how the buses or memory are grouped, that's a function of hardware only.

SQL Server has only one Lazy Writer and one I/O handler. The I/O handler handles all of the I/O completes and the Lazy Writer keeps the disk updated -- during slow times in the system -- with changes made to memory.

The problem: If you have a system with many CPUs and many users the thread that handles the Lazy Writer and I/O could become a bottleneck.

The solution: Configuring many soft NUMA nodes provides one I/O thread and one Lazy Writer thread per node.

You can divide hardware nodes into even smaller groups (soft Nodes), but you must be careful not to cross hardware node boundaries when you configure SQL Server for soft NUMA and hardware NUMA is present in the system.

Let's say you have two hardware NUMA nodes in your system, the first node contains CPUs 0-3 and the second node contains CPUs 4-7. You could create a soft NUMA node to group processors 0 and 1 together because they are both in the same hardware NUMA node. And you could create another node to group processors 2 and 3 together. But you can't create a node that would group processors 0 and 4 together -- because you can't cross hardware boundaries in a soft NUMA node.

How to Configure SQL Server for Soft NUMA

As we mentioned earlier, you configure soft NUMA nodes by a registry entry -- here's how:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\100\NodeConfiguration

Type	Value Name	Value
\Node 0	DWORD	0x03
\Node 1	DWORD	0x0c
\Node 2	DWORD	0x30
\Node 3	DWORD	0xc0

Figure 1: NUMA Configuration Example

With the right most bit of the value representative of CPU 0, the mask value selects which CPUs will be associated with which node. In this case processors 0 and 1 will be associated with node 0, processors 2 and 3 will be associated with node 1, and so forth.

Storage Considerations

Types of Storage

- SQL Server works best with disk drives that have high rotation speeds and low seek times.
- A more expensive option is to use **SAN**s (Storage Area Network) -- especially for high-end, high configuration systems.
- Always err on the side of more disk space.
- You must consider I/O throughput as well.

Even with plenty of disk space you could still have throughput issues. For example, if you use a two terabyte single drive and you have multiple users, the time to access the single drive could slow the entire system.

You should consider the type and amount of storage used as well. Here are some hardware considerations for disk storage:

SQL Server works best with disk drives that have high rotation speeds and low seek times. You may even consider using the more expensive **SAN**s (Storage Area Network) -- especially for high-end, high configuration systems. You want to always err on the side of more disk space - more is always better when it comes to storage capacity. And you must consider I/O throughput as well.

Managing Disk Storage

Rotation speed and seek times are important, but so is availability. Performance of a drive is equal to the speed of the reads and writes. But the performance in a system is equal to the speed of the reads and writes of the individual drives and the availability.

In other words, you may have high performance drives, but limited availability will create bottlenecks and slow down the entire system. Fault tolerance is the ability of the system to continue functioning through a system failure without a loss of data.

You should consider **NTFS** (New Technology File System) formatted drives with a 64KB allocation table. Why a 64KB allocation table? 64KB is the size of one extent. One extent is equal to eight pages (one page equals 8K of data) side by side. Some large I/Os can read an entire extent at one time so the minimum allocation unit should be the same size as one extent.

Another way to increase performance is to spread data across many different drives. This allows parallel scans of a table and therefore increases throughput.

RAID and SQL Server

RAID (Redundant Array of Independent Disks) combines two or more hard disks in a single logical unit. The data is divided among several disks in an array of disks that are seen by the system as one unit, therefore the risk of data loss due to a disk failure is reduced.

Having all data in a server stored on single disks that are treated independently by the system is akin to having all of your financial investments in one company. If the company fails, you risk losing all of your investment.

Spreading data among several disks with mirroring or parity error correction is a way to protect the integrity of the data and spreads the risk around so that one failure cannot cost you all your data investment. This is the basis of RAID - although some configurations have neither redundancy nor parity but do increase performance.

A mirroring scheme, as the name implies, writes the same data to two or more disks at once making an exact copy of all of the data written to the original. Parity provides a way to correct errors in the data on the fly.

Note: Hardware RAID is better and faster than OS RAID.

There are several RAID levels available that allow differing combinations of performance, redundancy, and error checking. Levels 0, 1, and 5 are the ones typically implemented by the SQL Server so for test preparation we will focus on these three configurations.

RAID 0 - Striping

In RAID level 0 disk striping is used to separate the data into blocks and store data in a fixed order across the disk array.

- Minimum of two disks
- Data is written to one disk, then the other
- No fault tolerance
- Better read and write performance
- Max disks can be quite large

Disk striping is used at this level to improve the performance of read and write operations. With disk striping several disks are arranged in an array so that independent read and write operations can be performed at the same time to more than one disk, thus improving performance. Data is divided into blocks and in a fixed order across the disk array. Data can be written and read faster than on a single disk. With no mirroring or parity, a failure of a single disk can cause loss of data.

RAID Level 0 is similar to level 5 except there is no fault tolerance at level 0.

Advantage: Better read and write performance.
Disadvantage: No fault tolerance.

RAID 1 - Mirroring

This level uses disk mirroring to replicate data and provide an identical copy of the data on the disk. Disk mirroring, as the name implies, creates a mirror image of a disk by performing write operations on two disks at the same time.

This level provides fault tolerance through disk mirroring.

- Minimum of two disks
- One disk is an exact copy of the other
- Great for reading but not writing
- Fault tolerant

If only one disk controller is used the possibility of failure in the controller can cause loss of all disks functionality. Using one controller for each drive will eliminate this problem, this is called Disk Duplexing.

Advantages: Fault tolerant and generally improves read performance.
Disadvantage: May degrade write performance.

RAID 5 – Striping with Parity

Level 5 is similar to level 1 using disk striping with the addition of parity spread across the array for data error correction. Spreading parity across the array allows for the failure of any one disk in the array without loss of data. If one disk fails performance will suffer until the disk is replaced and rewritten, but data will not be lost.

- Minimum of three disks
- Data is added to one disk and parity to another
- Great for reading but there is a 50% performance penalty for writing
- Fault tolerant
- One drive can fail without loss of data

Advantages: Fault tolerant and improves read/write performance.
Disadvantage: If a striping member is missing, read performance is decreased.

Test Tip: Since levels 0, 1, and 5 are typically used by the SQL Server, understanding the construct, advantages, and disadvantages of those levels will help select the correct level given a particular scenario.

RAID Design Considerations

Transaction log files should be mirrored for the simple reason that you want the transaction log files to be in a fault tolerant setup. You don't want to risk losing the transaction log data and mirroring keeps a copy of every write on a separate disk.

Data files in most cases should be stored using RAID 5. You want the improvement in read performance (which you get with some cost of write performance) and fault tolerance as well. RAID 5 provides a good balance of performance (needed for I/O intensive - large databases) and fault tolerance using the parity error detection and correction.

The best setup for data files in most cases would be a combination of striping and mirroring (known as RAID 10). The cost for having higher availability and higher fault tolerance is that you need more drives to mirror the data.

I/O Throughput Considerations

Server I/O throughput is an important consideration with your SQL Server and selecting the right RAID level is a key to maximizing throughput. Other important keys that we have already discussed are selecting the right processor, disk read and write times, and managing memory properly.

Data compression, discussed later, is another key to optimizing I/O throughput. A database, once compressed, will remain compressed on the disk and during I/O disk transfers.

Backup and restore, triggers, bulk inserts and large stored procedures are some of the most I/O intense operations you will execute on the SQL Server. Compressed data increases throughput. This is because fewer pages need to be transferred as compared to non-compressed data. Backup and restore reads or writes an entire database from or to the physical disk(s).

Transactions per Second

A transaction is either one database operation or more than one combined into a single operation that is either fully committed or not performed.

The transaction rate is affected by many factors including system performance, number of users, I/O limitations, cache size and the complexity of the requests.

The Performance Monitor (discussed in detail in Domain 5, Designing a Monitoring Strategy) provides counters to monitor system performance:

- Transactions/Sec counter
- Write Transactions/Sec counter

The Transactions/Sec counter indicates the number of transactions that were started in the last second. The Write Transactions/Sec counter indicates the number of transactions that wrote to the database and were committed in the last second.

A high rate of transactions can indicate that some transactions are not completing.

Data Compression

Data compression on SQL Server 2008 helps the database administrator with I/O throughput issues by reducing the size of the database and thus reducing the I/O overhead required to access that data. When a database is compressed, the data is stored on fewer pages in the database and queries read fewer pages from the disk.

When data is exchanged in an application, extra CPU resources are used to compress and decompress the data. Therefore workload characteristics must be taken into consideration when deciding which data to compress.

Methods of compression include:

- Row compression which stores fixed length data in variable length format.
- Page compression which removes redundant data.

There is a stored procedure, **sp_estimate_data_compression_savings**, that will estimate the space savings one table at a time. This procedure takes a portion of the data and compresses it in **tempdb** (temporary database - discussed later) to determine the estimated savings of compressing the entire table.

You can determine which databases are using compression by a query of the dynamic management view **sys.dm_db_persisted_sku_features**. You can also query the **data_compression_desc** column in the catalog view **sys.partitions** to find out what is compressed and how.

Other considerations when deciding whether or not to compress a database are the actual data stored. Columns where there are a significant number of nulls, significant number of repeating data, and columns with a large number of data types where most values don't require the number of bytes allocated are all good candidates for compression.

The data compression feature is included in the Enterprise and Developer editions of SQL Server 2008. Databases that are compressed in these editions cannot be restored using any other editions.

Planning Capacity for tempdb

Each time you start an instance of SQL Server a new **tempdb** (temporary database) is created based on the *model* database. This is done to be sure your database starts clean. As you read on you'll see what is stored in tempdb and why it's important to start clean.

The default size for tempdb is 8MB and by default is set to expand by 10% when necessary. In SQL Server terms this default size is a pretty small file and with lots of user activity it can fill up quickly and need to expand frequently.

The Problem: Each time tempdb expands that database is locked and can result in slow server response time or the server may even appear unresponsive.

The Solution: You can permanently expand tempdb to accommodate for space needed during busy periods. Once expanded, tempdb will retain its new size during subsequent starts of SQL Server regardless of the size of the model database.

Note: You should, when possible, store tempdb for each server on its own drive.

For each instance of SQL Server the tempdb database must be configured for capacity requirements and its usage must be monitored in order to insure maximum performance.

Each instance of SQL Server contains the tempdb database as a global resource available to all users in that instance. This database is used to store user objects, internal objects, and version stores.

Temporary User objects - objects that are created by a user. These objects may be created directly by the user in a user session. Or they may be created indirectly by the user in a routine which could be a stored procedure, a trigger, or a user-defined function.

However they are created, here is a list of user objects.

- System tables and indexes
- Local and Global temporary objects
- Table variables
- Cursors
- Temporary Stored Procedures

Internal objects - creations of the SQL Server Database Engine that process SQL Server Statements. Here is a list of internal objects:

- Work tables for cursor or spool operations and temporary **LOB** (Large Object) storage.
- Work files for hash join or hash aggregate operations.
- Intermediate sort results for operations such as creating or rebuilding indexes.

The internal objects use a minimum of nine pages each; one IAM (**Index Allocation Map**) page and an eight page extents.

Version store - a collection of data pages containing the data rows required to support features that use row versioning. Version stores contain the following:

- Row versions that are generated by data modification transactions in a database that uses snapshot or read committed using row versioning isolation levels.
- Row versions that are generated by data modification transactions for features such as: online index operations, **MARS** (Multiple Active Result Sets), and AFTER triggers. Note: MARS was added in SQL Server 2005 and enables interleaved execution of more than one request on a single connection for applications accessing the database engine.

In order to determine the correct size for tempdb in a production environment, follow these steps:

1. Set tempdb to **autogrow.**
2. Execute individual queries or workload trace files and monitor tempdb space use.
3. Execute index maintenance operations, such as rebuilding indexes and monitor tempdb space.
4. Use the space-use values from the previous steps to predict your total workload usage; adjust this value for projected concurrent activity, and then set the size of tempdb accordingly.

To increase the size of tempdb manually - complete the following steps:

1. Start SQL Server Management Studio. In the Object Explorer view, select the server to connect to.
2. Select the Databases folder.
3. Right-click tempdb to configure.
4. Select Properties and you will arrive at the Database Properties window:

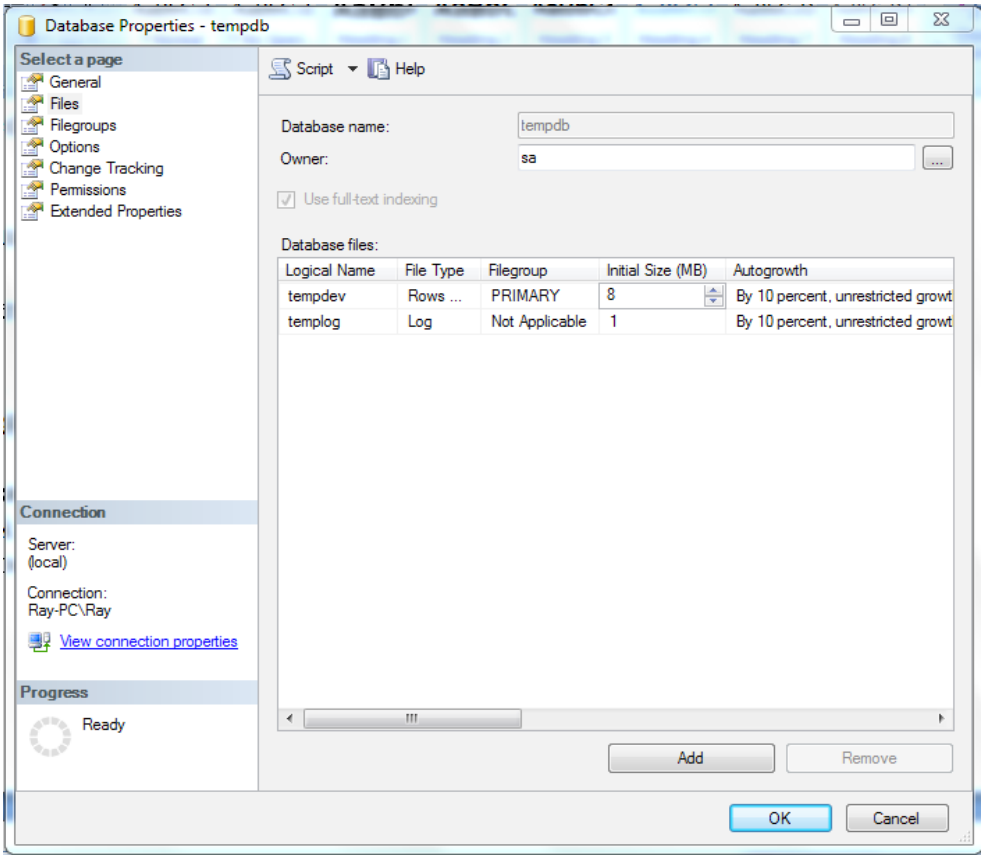


Figure 2: Increasing the size of tempdb

1. Select Files from the Select A Page list in the dialog box.
2. To expand the data file, click in the Initial Size Box, then enter the larger size.
3. Click OK and SQL Server locks the database while expanding the file.

Optimizing tempdb

To optimize the performance of tempdb:

- Pre-allocate the initial size of tempdb large enough for the estimated workload.
- Then set autogrow for minimum interference.

To do this enable autogrow with no max limit and use the following Microsoft recommended guidelines for autogrow increment:

Estimated Size	Autogrow Increment
< 100MB	10MB
100-200MB	20MB
>200MB	10%
You should maintain a maximum of two minutes time for autogrow. For example, if the system can initialize a file @ 50MB/sec, then 120 seconds equals 6GB max autogrow size.	

Figure 3: Understanding Autogrow

Estimating the Size of a Table

When considering a database design you need to consider your hardware configuration to be sure you have enough physical space for your database once it's in use. You will also need to consider the design of your database and whether you need more normalization. On the other hand if the estimated size of your database is small compared to your storage, you may want to denormalize your database in order to improve query performance.

For these reasons, before you design a database you may want to estimate the size of the database once it's filled with data.

If you are using a clustered index the estimated size of your database would be:

Clustered Index + Nonclustered Index

If you are not using a clustered index the estimated size of your database would be:

Heap Size + Nonclustered Index

Estimating the Size of the Heap

To estimate the size of the heap you will need the following information:

- **Num_Rows** = Number of rows
- **Num_Cols** = Number of Columns
- **Fixed_Data_Size =** Size of all Fixed-Length Columns
- **Num_Var_Cols** = Number of Variable Columns
- **Max_Var_Size** = Maximum Byte Size of All Variable-length Columns

1. Calculated the null bitmap as follows:
   Null_Bitmap = 2 + ((*Num_Cols* + 7) / 8)
2. Calculate the variable-length data size
   Variable_Data_Size = 2 + (*Num_Variable_Cols* x 2) + *Max_Var_Size*
   Note: If there are no variable-length columns, Variable_Data_Size = 0
3. **Calculate the row size**
   Row_Size = *Fixed_Data_Size* + *Variable_Data_Size* + *Null_Bitmap* + 4
4. **Calculate the number of rows per page**
   Rows_Per_Page = 8096 / *(Row_Size* + 2)
5. **Calculate the number of pages required for the heap**
   Heap size (bytes) = 8192 x *Num_Pages*

Designing a SQL Server Instance

SQL Server allows you to create multiple instances of SQL Server on a single system. In all respects, each instance is completely independent of the others. Even though they share the same physical resources, such as CPU, memory, and disk, logically, they have no more in common than SQL Servers running on different computers.

There are two types of SQL Server instances that can be created using the SQL Server Installation Wizard, the default instance and a named instance. The naming convention for a server instance is **server_name\instance_name**.

You can have multiple instances of SQL Server on a single machine. Only one instance can be the default instance and is always named MYSQLSERVER1\MSSQLSERVER or MYSQLSERVER1\SQLExpress for SQL Server Express. You can name an instance or use the default instance on the Instance Configuration page of the Installation Wizard.

Each instance shares tools and client software, but has its own:

- Registry entries
- Folder locations
- Services with accounts and setup
- Executables
- TCP Ports

Naming Convention

If you select the Named Instance option here are some conventions you must follow when selecting a name for the instance you are creating:

- The first character in the instance name must be alphanumeric.
- Certain characters are not allowed in an instance name including: Space ( ), comma (,), colon (:), semi-colon (;), ampersand (&), and at sign (@).
- Other characters like the underscore (_) and the dollar sign ($) are allowed.

The instance name will also be the **Instance ID** by default. You can use a different instance ID by entering it in the Instance ID field during setup.

The **Instance Root Directory** by default will be C:\Program Files\Microsoft SQL Server\. You can specify a different root directory during setup by changing the value in the Instance Root Directory field.
Note: The Database Engine, SQL Server Agent, and Analysis Services are all instance aware.

Surface Area Configuration

When a new instance of SQL Service is created, a minimal number of features are enabled. This is for security reasons because the less features that are enabled, the less features that can be attacked by a malicious user. In SQL Server 2005, the feature load out of an instance -- referred to as the instance's surface area -- was managed through Surface Area Configuration. This tool has been replaced by SQL Server Policy Based Management in SQL Server 2008.

Default features can be changed by the system administrator installation or features can be selectively enabled or disabled on a running SQL instance through SQL Server Policy Based Management.

The SQL Server Configuration Manager can be used to perform the following:

- Enable protocols and other connection options
- Start and stop services
- Configure start up options

To make these changes you must first start the SQL Server Configuration Manager. On the Start Menu select:

- All Programs
- Expand Microsoft SQL Server 2008
- Expand Configuration Tools
- Click SQL Server Configuration Manager

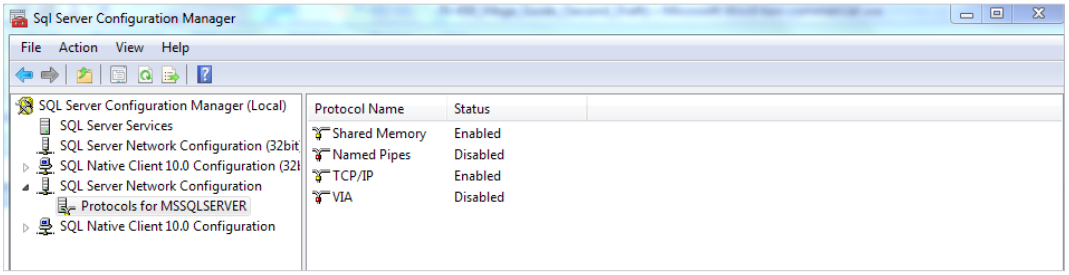


Figure 4: SQL Server Network Configuration

Use the SQL **Server Network Configuration** to stop/start services, enable connection protocols and to enable connection options like fixed TCP/IP ports or forcing encryption. Shown above are the protocols for the default instance of SQL Server along with their status.

CPU Affinity

In a multiprocessor system, CPU affinity provides masking to determine which CPUs are allowed to run threads from an instance of SQL Server. You can use the affinity mask to select which CPUs will run SQL Server threads and which CPUs you want to be dedicated to system processes. A binary bit mask can be set to enable or disable system processors in their access to run threads in SQL Server.

The **affinity_mask** option can be set to determine which CPUs will run SQL Server processes other than disk I/O. The **IO_affinity_mask** determines which CPUs will service SQL Server disk I/O processes.

The **IO_affinity_mask** is a number that holds the bit map for all of the CPUs in the system. The bit map is the same for I/O affinity as it is with affinity mask.

When an instance of SQL Server is created, the default setting for **IO_affinity_mask** will work best in most cases. However, if you have to change the mask value, it can be entered as a decimal which is not easy for larger systems or you can enter the number in hexadecimal if you enter the number with a "0x" prefix. The risk of setting the **IO_affinity_mask** to the wrong value is that you could create bottlenecks where some processors are tied up and others are mostly idle.

For instance, too many processors set to service SQL server I/O could cause an imbalance where those processors are not very busy and the ones that are left for system processes are overloaded. The opposite is true not enough processors are dedicated to service SQL disk I/O.

Microsoft suggests that **affinity_mask** and **IO_affinity_mask** be used in conjunction. Here's an example of how that would work. Set the affinity mask as follows:

1. **sp_configure "affinity mask", 0x000000FF**;
2. **go**
3. **reconfigure with override**
4. **go**
5. Shut down SQL Server
6. Start SQL Server with the following switch: **sqlservr –I0x0000FF00**

This configuration would assign the first eight processors CPU 0 – CPU 7 to SQL Server processing other than disk I/O in step 1. In step 6, eight more processors CPU 8 – CPU 15 would be assigned to SQL Server disk I/O.

Note: You should never have a corresponding bit set to 1 in both the **affinity_mask** and the **IO_affinity_mask**.

Note: There is no need to restart SQL Server after changing the affinity settings.

Memory Allocation

Once started, SQL Server will dynamically allocate memory as needed. You may notice that the Task Manager and Performance Monitor will show physical memory on the computer steadily decreasing down to between 4 and 10 MB. This is because SQL Server changes the size of its buffer pool (cache) according to the available memory reported by the operating system. Don't worry, this is normal behavior for SQL Server and SQL Server buffer will release memory as needed.

It is also normal to see the SQL Server private bytes used exceed the maximum memory setting. This is due to the memory that is set aside for other components such as extended store procedures.

You can use the **min server memory** and **max server memory** configuration options to limit the amount of memory used for SQL Server.

You **may not** need to change these memory settings, especially if you are on a 64 bit machine and have installed only one instance of SQL Server. You **may** need to change the memory settings of you are on a 32 bit machine and need to use AWE or if you have installed multiple instances of SQL Server.

For a dedicated SQL Server the max memory should be set to:

Total memory
- Application memory needed
- OS memory needed

Max memory for SQL Server

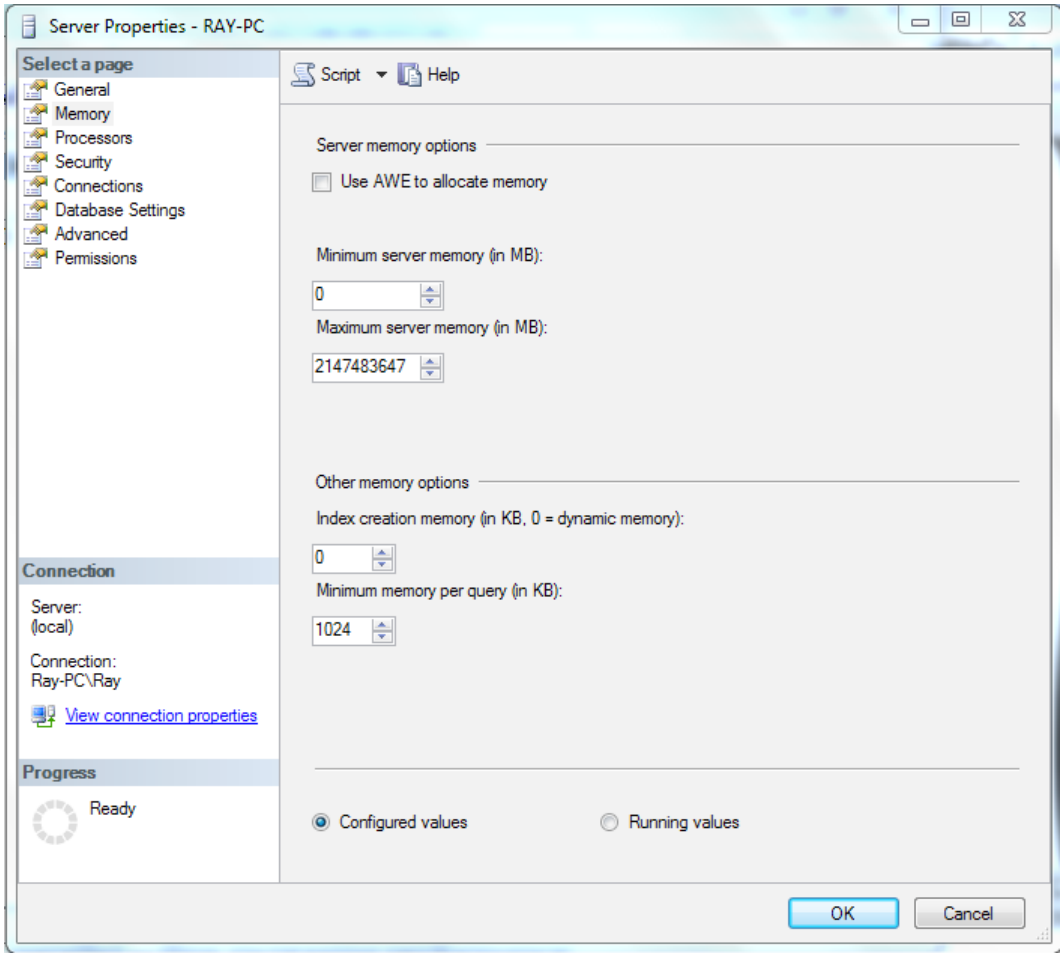


Figure 5: Configuring Minimum and Maximum Memory Settings

To select the min and max memory using the GUI, open SSMS and, in Object Explorer, right click the SQL Server instance and select properties to display the properties window. Then select memory and you will see the screen shown above. Here you can set memory properties including min and max values.

MAXDOP

Max Degree of Parallelism (MAXDOP) refers to the number of processors employed to run a single statement, for each parallel plan execution. A parallel plan execution is a feature of SQL Server that takes advantage of multiple processors on a computer to perform a query or index operation in parallel -- thus increasing performance.

When SQL Server is running, and there is more than one processor on the computer, it detects the best degree of parallelism. In other words, SQL Server selects the number of processors employed to run a single statement.

The **max degree of parallelism** configuration option can be used to limit the number of processors to use in parallel plan execution (up to 64). You can limit the number of processors used in parallel plan execution by setting this value; 1=No parallel plan generation, a number greater than 1 sets the maximum number of processors to use for parallel plan execution. The default value is zero, which means that all processors can be used for parallelism.

Note: If the value max degree of parallelism is set greater than the number of processors in the system, all of the processors will be used for parallel plan execution.

It's common practice in SQL Server to set MAXDOP to a value less than the number of processors in the systems. This reserves some system resources and prevents any single operation from dominating all of the processors.

You can set the value of MAXDOP in SSMS or T-SQL. For SSMS, right click on the server instance in Object Explorer and select **Properties**, then click on the **Advanced** tab in the Server Properties window as shown below. Under **Parallelism** you will see the setting for Max Degree of Parallelism and can enter the new value there.

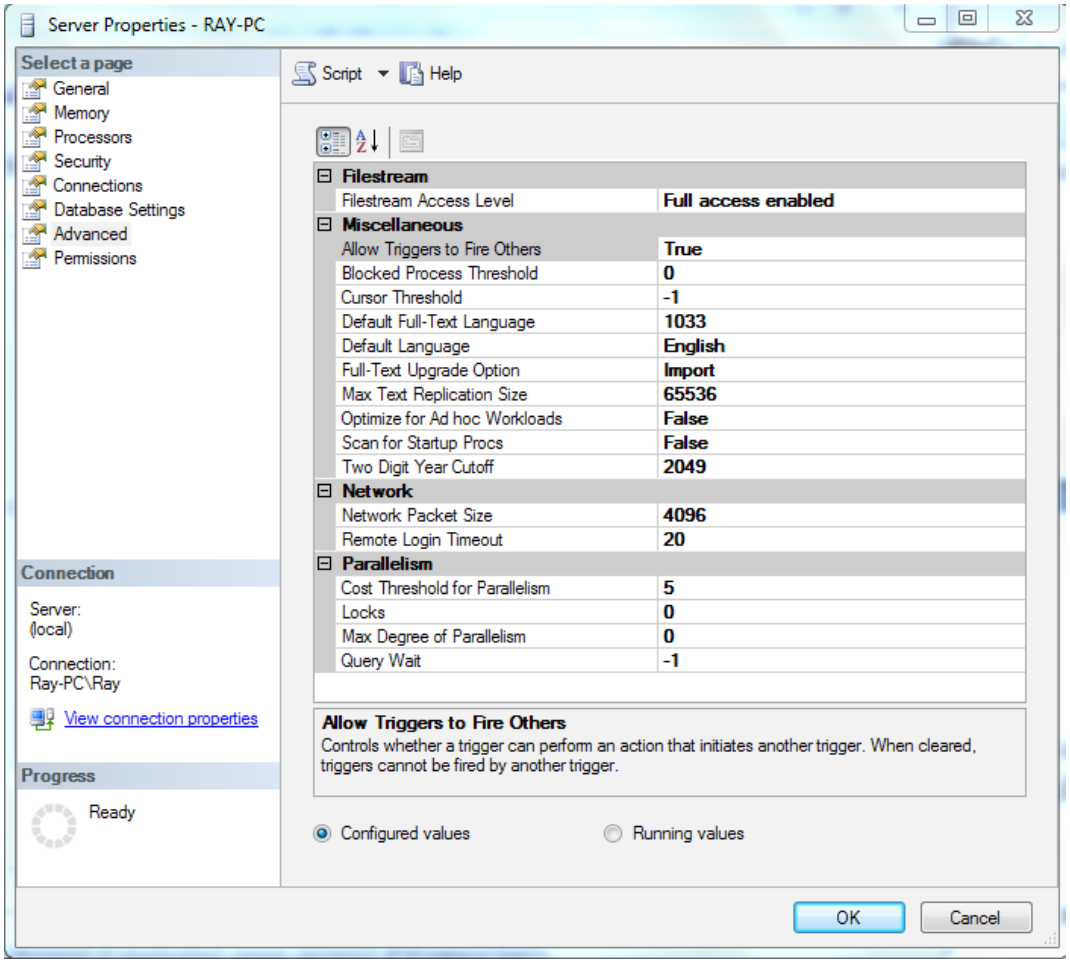


Figure 6: Configuring Maximum Degree of Parallelism

Collation

Collation defines how SQL Server 2008 stores and sorts data for different locales and languages. When an instance of SQL Server 2008 is installed you can set the default collation. Collation can be set or overridden at multiple levels listed here -- after the installation:

- The server
- A database
- A column
- An expression

There are two types of collation:

- Windows
- SQL

Collations differ based on character sets used around the world -- The most common collations are:

- Latin1_General collation for English and German
- Arabic collation for Arabic languages
- French collation for French
- Modern Spanish collation for Spanish

When you install SQL Server you have the option to choose the default collation and select the sort order. Sorting sensitivity can be set by:

- Case
- Common character types
- Accents
- Character width

Windows Collation

With Windows collation the alphabet will match the regional or locale settings in the operating system. The default collation for English in Windows is **Latin1_General**. A collation tab is present in the instance configuration and you can change the default -- and you can choose the sort order:

- Binary
- Case-sensitive
- Case-insensitive

Dictionary sorting is the default sorting method in Windows.

An advantage of Windows sorting is that single byte and double byte characters are treated the same in sorts and comparisons.

SQL Server Collation

With SQL Server collation allows you to select the code page character set. You can also select the sort order and Unicode collation sequence – and it allows you to sort the single-byte and double-byte characters separately.

Although SQL Server collation affects the code page used, it does not affect the Unicode data types. However, Unicode data types may compare and sort differently which could be a disadvantage.

Windows collation works best in most cases. But you may want to use SQL Server collation for backward compatibility or when matching another server's collation.

SQL collation names always begin with SQL. So any collation name that does not begin with **SQL_** is a Windows collation.

Note: A collation tab is present and the default collation can be changed in the instance configuration during instance installation.

Note: Once a collation has been selected for a server instance, changing the collation will require a complete database rebuild.

Note: To maintain compatibility with Analysis Services select Windows collation. Analysis Services does not use SQL collation.

Test Tip: Collation affects the sorting, character sets, and case sensitivity of data stored by SQL Server.

Collation in T-SQL

To see a complete list of available collations in SSMS, run the query:

SELECT * FROM fn_helpcollations()

You will see a list of Windows as well as SQL Server collations available in the results .You can set the collation for a single database when you create the database as follows:

CREATE DATABASE test COLLATE Latin1_General_CS_AS

You can change the case sensitivity in a table within the database as well:

CREATE TABLE mytable (no_case_data VARCHAR(128) COLLATE Latin_General_CI_AS NOT NULL)

This query creates the table 'mytable' with a table entry 'no_case_data' that is Latin General Case Insensitive.

Designing for Physical Databases and Object Placements

There are two files created when you use the Database Wizard to create a new database. These files are the database data file named **database_name.mdf** and a transaction log file named **database_name_log.ldf** – where **database_name** is the name selected for your database.

The **database_name.mdf** is the primary database file and the **database_name_log.ldf** is the transaction log file associated with that database. Both of these files are stored in the SQL Server's data directory. These files are stored in SQL Server's data directory.

The default directory for these files is:

C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA

SQL Server uses three file types to store data and log information:

- The **.mdf** files are the main database file. Each database has one primary database file.
- The .**ldf** files are the log files associated with the database file for each database.
- The .**ndf** extension -- used for extending databases but are not necessarily created at the same time as the database files are created.

Although SQL Server does not enforce the extensions listed here (.mdf, .ldf and .ndf), they do help you identify the different types of files and how they are used.

Filegroups

Filegroups allow you to spread data across many files which allow I/O parallelism, help with partitioning/ isolating indexes and views. Tables and indexes are stored in filegroups.

Why Use Filegroups

Filegroups enhance performance by spreading the I/O among several files and providing the ability to separate read-only data from data that will be updated. Maintenance is easier because you can backup and restore the data independently, use **DBCC** (Database Console Command) statements on groups of tables in a filegroup and separate user objects from system objects.

SQL Server 2008 creates a default filegroup for each new database created called the PRIMARY filegroup. Unless you specify other filegroups all new data files will be placed in this filegroup. SQL Server uses the PRIMARY filegroup to store a variety of system information. Log files are never part of a file group because log and data files are managed separately.

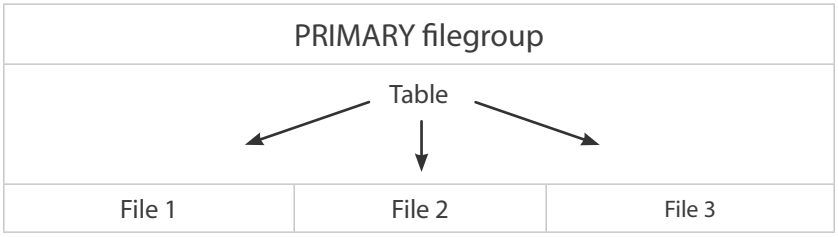


Figure 7: Filegroups

In the above illustration the PRIMARY filegroup contains three data files and the table is striped across all the files in the filegroup. Like disk striping, this allows I/O parallelism.

You can specify a filegroup as read-only so you can separate read-only data from data that will be updated. For instance, if you store yesterday's data in a 'history' filegroup you can specify that group as read-only.

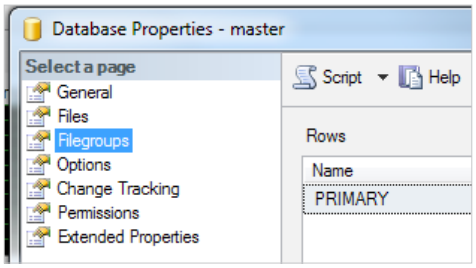


Figure 8: Creating Filegroups in the GUI

Here's how you can add a new filegroup to a SQL Server 2008 database:

1. Open SQL Server Management Studio and navigate to the desired database
2. Right-click the database object and select **Properties**
3. In the Database Properties window click **Filegroups**
4. Click the Add button
5. Enter your filegroup name in the Name cell
6. Click OK

After the filegroup is created, the filegroup name will appear in the drop-down box the next time you add a data file to your database.

Note: You will want to separate transaction logs (serial I/O) from data (random I/O) to improve performance.

Note: You will also want to use multiple drives to eliminate drive contention.

Data Files

In SQL Server there are two names for every file, a logical and a physical name. The physical file name (**physical_name**) is the name used in all T-SQL statements. The logical file name (**os_file_name**) is the physical file name with the path included and it must follow OS file name rules.

Data File Pages

In a data file pages are numbers sequentially starting at zero (0). Each file name has a unique file ID number that is used to reference the pages in that file. Both the file ID and the page number are required to access a page.

The first page in a file is the header page and contains attribute information about the file.

Primary Data File: ID 01			
Page 01:0000	Page 01:0001	Page 01:0002	Page 01:0003
Secondary Data File: ID 02			
Page 02:0000	Page 02:0001	Page 02:0002	Page 02:0003

Figure 9: Data File Pages

Log Files

Each time SQL Server Setup is executed it creates new log files in the log file folder at: programfiles\Microsoft SQL Server\100\Setup Bootstrap\Log\.

The log files are stored in a new timestamped folder with a name format of **YYYMMDD_hhmmss**. All old or filled log files are archived into the log*.cab file in their respective log folder.

FILESTREAM

A great portion of data that is created and stored these days is unstructured data -- like videos, text files, audio files and image files – data that you wouldn't store in a normal database. This data includes any data that won't easily fit in a structured database.

Problem: Unstructured data needed be stored outside the database and separate from the structured data. Managing unstructured data outside the database can be difficult or if it is stored in the database structure, streaming capabilities and performance will be limited.

Solution: FILESTREAM was introduced to solve this problem. FILESTREAM allows SQL Server to work easily with **BLOB**s (Binary Logic Objects**)** that are stored outside the database.

FILESTREAM is a new feature of SQL Server 2008 that allows files outside the database to be referenced as if they were in the database.

To distinguish normal BLOBs that are stored inside the database from BLOB's stored outside the database they are referred to as FILESTREAM BLOBs. Unlike normal BLOBs, FILESTREAM BLOBs don't have a 2GB file size limit.

FILESTREAM BLOBs are stored as **varbinary(max)** data types in the database and can include any type of unstructured data. Rather than using the SQL buffer pool for FILESTREAM BLOBs, SQL Server Database Engine uses the features of the **NTFS** (New Technology File System) for caching data while streaming data files. This leaves memory available for query processing and reduces the effect that FILESTREAM data may have on SQL Server Database Engine performance.

BLOBs can be standard varbinary(max) files that are stored in a database or they can be FILESTREAM varbinary(max) files that are stored in the file system.

From SQL Server Configuration Manager (shown below), FILESTREAM can be enabled by first selecting SQL Server Services and right-clicking the SQL Server instance that you want to configure; then select **Properties**. Use the SQL Configuration Properties dialog box to enable FILESTREAM if you want to use it, enable file I/O streaming access, and enable remote clients to have streaming access.

Once enabled, you can use T-SQL statements to search, query, insert, update, or delete FILESTREAM data. With the Win32 file streaming interface, large amounts of data can now be streamed more efficiently. You can use the Win32 interface that now works in the context of SQL Server transactions. Use the **OpenSqlFilestream API** (Application Programming Interface) to obtain a file handle -- then the following Win32 streaming interfaces are available, like **ReadFile()** and **WriteFile()**.

FILESTREAM data is stored in special filegroups that contain system directories instead of files. The system directories stored in the special filegroups are the interface between the Database Engine and the file system storage and make it possible to reference files outside the database as if they were in the database. When you want to store data to a FILESTREAM, simply specify the FILESTREAM attribute on a varbinary(max) column. The data is then stored on the file system and not in the database.

Most SQL Server Management tools and functions work well with FILESTREAM data. You can back up and recover FILESTREAM data along with the relational data in the database. Or you can exclude the FILESTREAM data if you like.

FILESTREAM data is secured using the NTFS security and permissions are granted at the table or column levels.

Here are a few need-to-know facts about FILESTREAM:

1. You can't use mirroring and snapshots on FILESTREAM filegroups.
2. FILESTREAM works with replication, log shipping and full-text indexing.
3. FILESTREAM works with clustering if the following is true:
   a. All servers have FILESTREAM enabled.
   b. The FILESTREAM filegroup is a shared resource.

To implement a FILESTREAM for interoperability you must:

- Enabled FILESTREAM on the SQL Server service.
- Create a share name.
- Use sp_configure to enable FILESTREAM on the engine:

```
EXEC sp_configure filestream_access_level, 2
RECONFIGURE
```

- Create a database in the FILESTREAM filegroup.
- Create a table with FILESTREAM columns that have varbinary data type.

When considering whether data should be stored in a FILESTREAM or in the database you should determine if objects will be larger than 1MB on average and whether you need fast access to the data. If either of those are the case -- use a FILESTREAM.

LOB Placement

When you create a table with **LOB** (Large Object), use the TEXT ON syntax in CREATE TABLE and place the table on a separate filegroup than other data and indexes to separate I/O. This will increase parallelism and provide balanced I/O.

Full-text Indexes

Full-text indexes can be placed in three locations:

1. Primary filegroup
2. Same filegroup as the base table
3. User-specified filegroup

It's common practice to place the full-text indexes on separate user-defined filegroups for maintenance and backup reasons.

Note: Since full-text indexes can consume lots of I/O, you want to place them on fast drives.

Designing a Migration, Consolidation and Upgrade Strategy Multiple SQL Instances

In considering multiple instances of SQL Server you must balance the needs of isolation vs. manageability. You might use multiple instances to allow for:

- Hotfixes
- Upgrade isolation

When using multiple instances of SQL Server you will want to group databases that have the same configurations and security requirements in the same server instance.

SQL Server creates a dynamic port each time the server is started. Best practice however, is to assign port numbers during install. If you assign port numbers it's best to use a value greater than 30,000. You want to make sure that the port you assign is not already in use.

Only one default instance of SQL Server is allowed. If a default instance is present from an earlier version of SQL Server (2000 or 2005) then SQL Server 2008 cannot create another default instance.

Note: With multiple instances don't let SQL Server default to dynamic memory assignment. You should assign a proper non-overlapping memory configuration.

The benefits of multiple instances include better isolation and different versions of software and service packs on the same machine. The risk is that a system failure can bring down all servers.

If you only plan on using one instance of SQL Server 2008, then it should be the default instance. If you plan to have more than one instance, then one should be the default instance and the other(s) will be named instances.

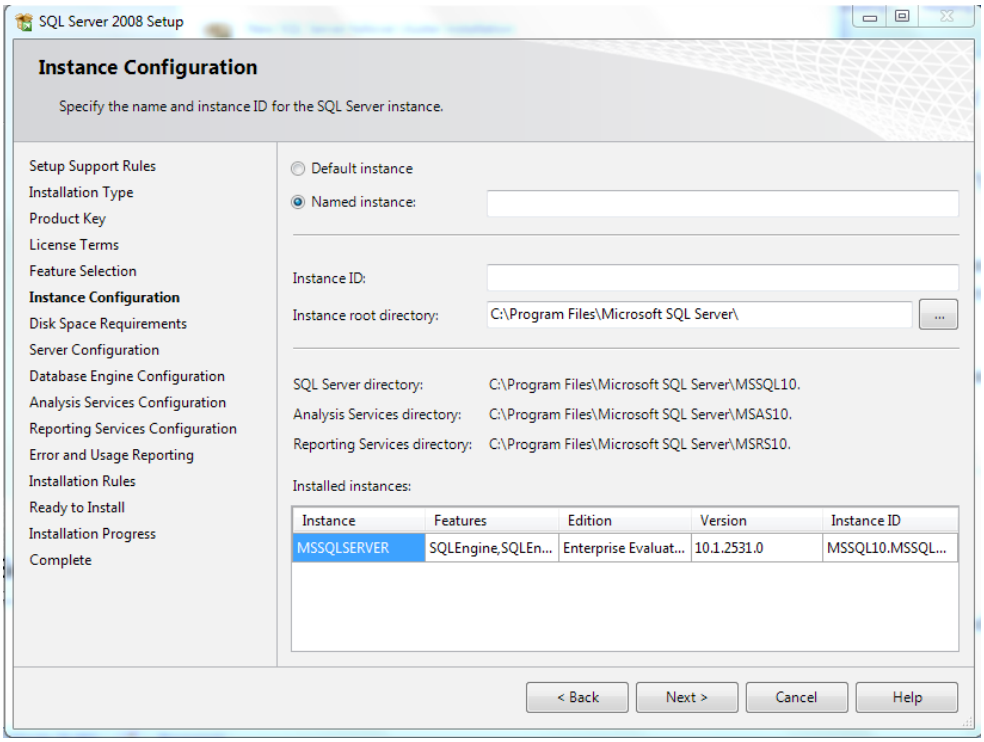


Figure 10: SQL Server Instance Configuration

In the SQL Server Installation Wizard you will have the option to select **Default Instance** or **Named Instance**. As shown above in the SQL Server Installation Wizard, the default instance already exists (it was created first), now you must select a named instance by entering the desired name for the new SQL Server instance.

Instance names are not case sensitive, cannot contain "Default", and if you select MSSQLSERVER as a name for the instance you are creating, a default instance will be created.

Besides "Default" there are other reserved words that cannot be used as well. Using a reserved word will result in an error.

Planning for Upgrade

The Problem: SQL Server has to be upgraded from time to time as new versions become available or new Service Packs are released. Poorly planned upgrades can lead to loss of functionality or even loss of data.

The Solution: The solution to the upgrade problem is to carefully plan your upgrade strategy so that these problems are identified and addressed before you actually perform the upgrade.

The first step to planning an upgrade is to check for known upgrade issues. You can do this by researching the Microsoft site and the internet for issues that are already documented.

Some things to consider before proceeding with the upgrade are:

- Upgrade type (Side by side or restore)
- Consolidation options
- Replication
- Mirroring
- Collation

It's important when considering the upgrade to create an upgrade checklist. This will insure that all known issues and your upgrade strategy are clear. It's **very important** that you do a trial run before you actually do the upgrade!

You can generally upgrade from one level of SQL Server on the same platform to the next. Just be sure that the older version has the latest Service Packs installed.

Cross-platform upgrades are not supported by SQL Server. If you are going from a 32-bit to a 64-bit platform you can backup your data, install the new version and then restore your data on the 64-bit platform.

When upgrading from older versions, always check hardware and software requirements to be sure you don't run into compatibility issues.

Run the Upgrade Advisor for every upgrade.

For mirrored data you will need to follow these steps:

1. Back up the data
2. Remove the witness
3. Change from high-performance to high-safety mode
4. Upgrade the primary server
5. Failover
6. Upgrade the secondary server
7. Resume monitoring

For replicated databases:

1. Ensure log reader is running
2. Stop publishing activity
3. Let the log reader catch up
4. Restart agents
    a. Run snapshot agent for publication when using merge replication
    b. Run merge agent for each subscription
    c. Redo replication scripts
5. Perform the upgrade
6. Register servers in SQL Management Studio
7. Execute **sp_updataset**
8. Configure the new installation

Note: After the upgrade be sure to backup your data.

Consolidation

When planning an infrastructure you will want to consider consolidation.

Problem: When server instances are spread across hardware (different machines) there can be waste and management issues.

Solution: Consolidate instances to one machine and/or one instance.

The benefits of consolidation include the possibility of less expense, simpler administration and simpler security. Less expense because of less duplicate data and fewer potential licenses resulting from using less hardware. Simpler administration from less complex upgrades and backups – And Simpler security because of a reduction in server to server queries.

When you consolidate to one machine you are, however, creating a single point of failure by having everything on one machine. And you are lowering the isolation between servers and databases.

Note: The new consolidated servers must be properly sized for hardware requirements. For instance, you should consider having at least as much memory on the consolidation server as is required on the original server. This applies to other resources also, such as CPUs and memory.

Note: Memory required will be the sum of server memory.

Three Types of Consolidation

1. **Logical Consolidation** – where databases are migrated into a single instance.
2. **Physical Consolidation** – where servers from different locations are moved to a single location.
3. **Virtual Consolidation** – where SQL Server instances from individual servers are migrated to a single server using multiple instances.

Domain 2: Designing a Database Server Security Solution

In this section we will examine database server security and how to design security into all levels of interaction within a server and from server to server. This includes security design for logins and interactions with the data base including encryption strategy, keys, certificates, end point security policy, audit planning, permissions and roles.

Designing Instance Authentication

Choosing an Authentication Type

You have two choices for authentication modes on SQL Server, Windows only or mixed. Windows is always available and can be used alone or SQL Server authentication can be added to Windows authentication.

Although Windows authentication offers more password options, there are some advantages to using SQL Server authentication also. For example, there are some older applications that require SQL Server authentication, for those applications using SQL Server authentication would be necessary. Users may come from Windows domains that are not trusted by Windows or other operating systems and they would need to use SQL Server authentication to log in.

Among the disadvantages are more logins and passwords to keep up with and less security.

Logon Triggers

Logon triggers are used to fire off stored procedures in response to logon events. The stored procedures run after a login is authenticated but before the user gains control. These triggers do not fire if the authentication fails.

You can use logon triggers for any number of things. One use would be to count the number of logins. The messages from the triggers will go to SQL Server Error log but no way is provided to communicate with the person logging on.

Logon triggers can be used to:

- Audit users
- Control sessions
- Manage licenses
- Preload data

Note: A **ROLLBACK** in the logon trigger denies login to the user.

Note: You can set **EVENTDATA** inside the logon trigger for more information about the user logging on.

C2 Audit Mode

You can turn on C2 Audit Mode in the **GUI** (Graphical User Interface) in **Server Properties**. With C2 Audit Mode enabled, SQL Server creates a profile trace and logs events to a file (**mssql\data directory**). It records failed and successful access to statements and objects.

The C2 Audit log file has a maximum size limit of 200 MB. When it reaches that limit it will create a new file, close the old file and start saving event information to the new file.

You can query the **sys.traces** catalog view to see the status of a C2 trace.

Note: C2 Audit mode causes overhead which could slow down system processes. If C2 Audit Mode is enabled but cannot log, SQL Server will shut down and must be restarted with the –f parameter.

To enable C2 Audit Mode in SSMS right click the server instance in Object Explorer, then select **Properties**. In the Server Properties window select **Security**. Under Options select **Enable C2 audit tracing** to enable the C2 Audit Mode.

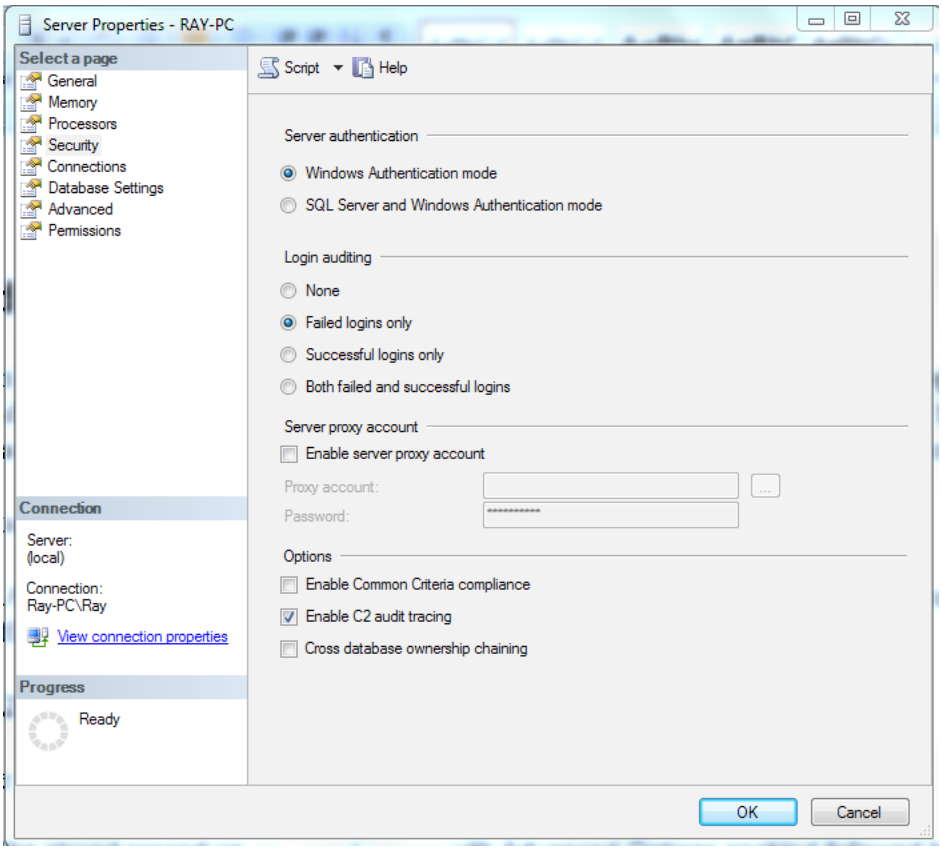


Figure 11: Configuring C2 Audit Mode

- For T-SQL run sp_configure with Advanced Options enabled, then reconfigure:

```
sp_configure 'show advanced options', 1;
    RECONFIGURE
```

- Then enable C2 Audit Mode as follows:

```
EXEC sp_configure 'c2 audit mode', 1;
    RECONFIGURE
```

Once C2 Audit Mode is enabled, restart the server.

Login Auditing

Login auditing is a simple feature that can be turned on or off. It can log unsuccessful logins, successful logins or both. Results are written to the SQL Server error log. Once set, login auditing requires a restart of SQL Server to change.

You can enable login auditing in Server Properties by selecting **Security** and under Login Auditing select the type of auditing you're interested in.

Designing Instance-Level Security Configurations

SQL Service Accounts

When you install SQL Server, you will choose the service accounts that the services will run under. Here are some of the service account options SQL Server provides:

- **Local User Account** – with a local service account the logins are created at the local SQL Server level. This type account does not have administration rights and is recommended if the SQL Server is not part of a domain.
- **Local Service Account** – although you can use the local service account, it is not recommended for SQL Server and SQL Server Agent services. Network resources would be accessed as a NULL session, without credentials. However, you are allowed access to resources and objects at the same level as the users groups. With the limitations of the local service account, you would not be likely to use it as an account option in SQL Server.
- **Domain User Account (Minimally Privileged)** – a minimally privileged domain user account should be used if the service must interact with network services – uses linked server connections to other computers – or must access domain resources like file shares. This is the most common account option. This account type is best if you have many server-to-server connections and there are many server-to-server activities can be services ONLY with this account type.
- **Network Service Account** – with a network service account you have more access to resources and objects than with members of the users groups. This is a built-in account that gives you access to network resources by the credentials of the computer account.

FILESTREAM Security

You can set FILESTREAM security at the service and within the instance. You can choose the following for FILESTREAM security to allow:

- No FILESTREAM access.
- T-SQL only access
- T-SQL and Win32 access

You may also set up a server proxy account so that you can connect through the new Forefront Threat Management Gateway (formally known as Internet Security and Acceleration server, or ISA).

Note: Be sure to set the FILESTREAM security so the no one is allowed to update the directory structure directly.

SQL Server Agent Services

You will want to use a Windows account for SQL Server Agent. It should be a low-privilege domain account.

You secure SQL Agent by assigning members to one the following roles:

- SQL Agent User Role
- SQL Agent Reader Role
- SQL Agent Operator Role

Credentials

A credential provides the authentication needed for a user to connect to a resource outside of SQL Server. A credential provides the information needed for a user -- who is connected to SQL Server -- to access resources outside the SQL Server instance they are connected to.

Credentials, in most cases, contain Windows login information and are mapped to SQL Server logins. Once a user is connected to SQL Server, that user can access resources outside the instance using the credential that is mapped to the user.

You can map a single credential to multiple SQL Server logins, but each SQL Server login can only be mapped to one credential. Users with ALTER ANY CREDENTIAL permission can create or modify credentials.

When you create a SQL Server Agent job step you will first need to determine the permission needs for you users for each job step type created. Once your user permissions are determined then you will:

- Create credentials
- Create proxies
- Assign users to the proxies

Here is the syntax to create a credential in T-SQL:

```
CREATE CREDENTIAL credential_name WITH IDENTY = 'identity_name'
        , SECRET = 'secret'
```

IDENTITY will be the name of the account that will be used to connect to resources outside SQL Server. SECRET is the secret required for outgoing authentication. If you are using a Windows login, then SECRET is the Windows login password.

To create a credential in SQL Server Management Studio, expand **Security** in the Object Explorer, then right click **Credentials** and select **New Credential**. You will get the window below where you can enter the credential name, the identity and the password.

Once a credential is created, can use the T-SQL statement, ALTER LOGIN to map any login to the credential you created.

```
ALTER LOGIN test
        WITH CREDENTIAL = credential_name;
```

Now the credential is mapped to the login for access to an external resource.

Instance Level Permissions

Permissions can be granted at the Windows level, at the SQL Server level and at the database level. Instance level permissions are the permissions that can be granted to users at the server level. These permissions include the following list of fixed server roles:

- **Sysadmin Fixed Server Role** – the sysadmin role can perform any activity without restriction. The sysadmin members are chosen during SQL Server instance install.
- **Serveradmin Fixed Server Role** – a member of this role can change server-wide configuration options and can shut down the server.
- **Securityadmin Fixed Server Role** – a member of this role is responsible for managing logins and their properties. In this role a member can grant, deny and revoke server-level and database level permissions and can reset passwords for SQL Server logins.
- **Processadmin Fixed Server Role** – the processadmin role allows a member to end processes running in an instance.
- **Setupadmin Fixed Server Role** – in this role the member can add or remove linked servers.
- **Bulkadmin Fixed Server Role** – a member can run bulk operations.
- **Diskadmin Fixed Server Role** – a member of the diskadmin role manages disk files.
- **Dbcreator Fixed Server Role** – a member of this role can create, alter, drop and restore any database.
- **Public** – everyone is a member here and all members are restricted by the default permissions.

Server Permissions

In addition to permissions granted by the role, you can grant additional permissions to individual members in a role.

Here's the syntax for T-SQL:

GRANT permission [,..] TO <grantee> [,...] [WITH GRANT OPTION] [AS <grantor.>]

The permissions that can be granted are listed below along with the implied server permissions.

The 'grantee' is who is getting the permission from the 'grantor'.

The grantor must have higher permissions that imply the permission being granted or have been granted permission with GRANT OPTION. Members of the sysadmin fixed server role can grant any permission.

Note: If you use **GRANT... WITH GRANT OPTION** you also give the member permission to grant the same permissions that you granted to them.

Note: Permissions at the server level can only be granted when the current database is **master**.
Here is a list of server permissions:

Server Permission	Implied by Server Permission
ADMINISTER BULK OPERATIONS	CONTROL SERVER
ALTER ANY CONNECTION	CONTROL SERVER
ALTER ANY CREDENTIAL	CONTROL SERVER
ALTER ANY DATABASE	CONTROL SERVER
ALTER ANY ENDPOINT	CONTROL SERVER
ALTER ANY EVENT NOTIFICATION	CONTROL SERVER
ALTER ANY LINKED SERVER	CONTROL SERVER
ALTER ANY LOGIN	CONTROL SERVER
ALTER ANY SERVER AUDIT	CONTROL SERVER
ALTER RESOURCES	CONTROL SERVER
ALTER SERVER STATE	CONTROL SERVER
ALTER SETTINGS	CONTROL SERVER
ALTER TRACE	CONTROL SERVER
AUTHENTICATE SERVER	CONTROL SERVER
CONNECT SQL	CONTROL SERVER
CONTROL SERVER	CONTROL SERVER
CREATE ANY DATABASE	ALTER ANY DATABASE
CREATE DDL EVENT NOTIFICATION	ALTER ANY EVENT NOTIFICATION
CREATE ENDPOINT	ALTER ANY ENDPOINT
CREATE TRACE EVENT NOTIFICATION	ALTER ANY EVENT NOTIFICATION
EXTERNAL ACCESS ASSEMBLY	CONTROL SERVER
SHUTDOWN	CONTROL SERVER
UNSAFE ASSEMBLY	CONTROL SERVER
VIEW ANY DATABASE	VIEW ANY DEFINITION
VIEW ANT DEFINITION	CONTROL SERVER
VIEW SERVER STATE	ALTER SERVER STATE

Figure 12: Server Permissions

Using Keys and Certificates

A certificate is used by SQL Server to associate a public key with a key holder to provide certificate-mapped authentication as well as encryption. It does this by associating an individual user's identification with the certificate with information like:

- Name
- Email address
- Valid date range
- Issuer information
- Digital signature

You can use the CREATE LOGIN statement to create logins that are mapped to certificates for login authentication.

Some certificates are created manually and some are created automatically at many levels in SQL Server. SQL Server maintains a hierarchy of keys from the server level down to the database. The SQL Server Master Key is used to encrypt all of the keys below it in the hierarchy.

You can create Database Master Key which could be used for database data encryption. This key would be encrypted using the Server Master Key.

Note: It's important to remember to back up and restore certificates when you are moving from one server to another.

Encryption Methods

You can use keys and certificates to digitally sign code. You will use this method to prevent tampering with the code. You can also use keys or certificates for data encryption. Of course, data encryption can be processor intensive so use caution and be selective about what data you encrypt. And you can use keys and certificates for authentication from server to server.

How to Manage Keys

Back up your keys; SQL Server Reporting Services, Service Master Key and the Database Master Key(s). If the keys have passwords, make sure you keep the passwords.
Keys can interact with:

- Mirroring
- Clustering
- Encrypted data

If you are backing up and restoring to a new server, you can restore key or certificate. This will allow the new server access to the encrypted data; otherwise the new server will not be able to decrypt the data.

Important: If you change the SQL Service account, you must be careful to manage the change or you could lose access to the master key. If this happens, everything that is encrypted by it will be lost! Use the SQL Server Configuration Manager to manage a SQL Server service account change. If you are moving SQL Server to another machine, you must migrate the service master key using backup and restore.

Enterprise Key Management

With SQL Server, **Enterprise Key Management** (EKM) is not a standard function. Microsoft created hooks so that third party vendors could supply the code for EKM. So you must buy software from a third party in order to make EKM functional in SQL Server.

The third party software will integrate with SQL Server and it should manage keys, key storage and encryption.

This can be done through software or **Hardware Security Modules** (HSM).

Benefits of EKM

EKM provides an additional authorization check and if the third party module is hardware based, you will get higher performance encryption. Hardware based EKM can generate keys outside of SQL Server to help recover, retrieve and dispose of outdated keys.

Using third party vendors you will want to be aware of what is provided for you. Some offer certificate aging and rollover which maintains a valid date range. Some offer bulk encryption of data as well.

Implementing EKM

In SQL Server you implement EKM with sp_configure by enabling EKM Provider (set to 1) -- then run re-configure to implement the change.

Copy the DLL that came from the third party vendor to SQL Server. Then create a cryptographic provider and give it a name; **FROM File=<filename>.**

Once you have the EKM enabled and the third party software installed -- The third party DLL will be then used for the T-SQL commands:

- CREATE KEY
- CREDENTIAL
- ENCRYPT
- DECRYPT

Securing Ports

Any communication external to SQL Server is accomplished via a port. When you are designing a SQL Server instance you should be aware of the necessity of port security and pick a strategy to secure the ports.

SQL Server, by default, uses ports 1433 (TCP) and 1434 (UDP) for SQL Server browser. By default, named instances will automatically use a dynamic port unless a port is assigned as a fixed port.

For named instances, every time a SQL Server instance starts, it searches for available ports. Once a port is found the browser registers that port and directs connections to the appropriate port for that named instance.

The obvious port security is accomplished by placing SQL Server behind a firewall. The default ports for SQL Server (1433 TCP and 1434 UDP) are normally blocked to prevent access by people outside the firewall. If you are using a browser you may wish to open port 1434 UDP.

More security can be achieved by changing the default port for each instance of SQL Server. You can also hide the server and the browser will not respond with server info.

Using SQL Server Configuration Manager you can create an alias for the connection -- that maps to a friendly name -- which include the network server name, the protocol and the port number.

Securing Endpoints

A SQL Server endpoint can be defined as an entry point into SQL Server -- and it's a good place to provide security for SQL Server. It is implemented as a database object and SQL Server routes all network connections through endpoints.

Endpoints provide a layer of security that is very similar to a firewall on a network. This security is provided for the point of connection between a network and a server or a client and allows the DBA to limit the type of traffic allowed.

Endpoints can have a protocol such as TCP/IP or HTTP and can be used by T-SQL, database mirroring, Service Broker and **SOAP** (Simple Object Access Protocol).

End Point Management

The system administrator can manage endpoints using **DDL** (Data Definition Language) and the owner's endpoints are managed by the owner themselves. Owners can grant connect permissions to the endpoint. Owners can also grant, revoke or deny either the control or alter endpoint permissions.

Here is the T-SQL to create an endpoint called my_user_connection for all available TCP addresses on the server -- on port 1500:

```
CREATE ENDPOINT [my_user_connection]
STATE = STARTED
AS TCP
        LISTENER_PORT = 1500, LISTENER_IP = ALL
FOR SQL();
```

Once you create an endpoint for your server, all 'Public' connection permissions will be revoked and you will have to GRANT CONNECT back to allow those connections.

To grant 'Public' access to this endpoint you have to reapply the permissions using the GRANT CONNECT ON ENDPOINT as shown below:

```
GRANT CONNECT ON ENDPOINT::[TSQL Default TCP] to [public]
```

Secure HTTP endpoints can be set up by the owner or the system administrator only. And they are set up with the **CREATE ENDPOINT** command as well. Then you can **GRANT CONNECT** to the other users you want to have access to the connection. There are no default endpoints or Web methods.

All HTTP connections have to be authenticated and you can choose from a variety of authentication methods:

- Basic
- Digest
- NTLM
- Kerberos
- Integrated

SSL Encryption

In order to use **SSL** (Secure Sockets Layer) you have to have certificates and they have to be installed on the client and the server. When SSL is used it will encrypt data between an SQL Server instance and a client. For SSL to be used it must be requested by the client.

SSL uses certificates so certificates must be installed on both the client and the server.

Note: Performance suffers when you use SSL for encryption and decryption.

SSL Certificate Requirements

Certificates for SSL must:

- Be in the local or current user certificate store
- Have a valid date range
- Be a Server Authentication ticket
- The certificate must be created using the **KeySpec** option **AT_KEYEXCHANGE**
- In the Subject property of the certificate, it must indicate that the common name (CN) is the same as the host name or the fully qualified domain name (FQDN) of the server computer

Note: Windows Server 2008 **AD CS** (Active Directory Certificate Services) is often used for **PKI** (Public Key Infrastructure) certificates commonly used by companies for their digital certificates.

In addition to the above requirements, the property **AT_KEYEXCHANGE** must use the **keyspec** option and the subject property must be:

- Common name
- **FQDN** (Fully Qualified Domain Name)
- Server Virtual Name
- Cluster Virtual Name

Implementing SSL

1. Get or create a certificate.
2. Import the certificate on SQL Server with **MMC** (Microsoft Management Console).
3. Configure SQL Server to accept encrypted connections on the server and the client side.
4. On the client side you should import the certificate using MMC and use the force encryption option to request encryption.

Designing Database, Schema and Object Security Parameters

In this section we will go down to the next level of security—from server security to database security.

Principals

Principals are identities that can that can be granted or denied access to securables. They provide the means for a user to authenticate and be identified at both the instance and database level. They consist of both individual logins and collections like a Windows NT group. Each principal has its own **SID** (Security Identifier).

The hierarchy of principals starts with the Windows principals (this can be a group), domain login or local logins.

The next level in the hierarchy is the SQL Server principal and contain login only. In database security there are users and these users map to one of the higher principals.

What are Securables?

Securables are simply anything that can be secured in SQL Server. In Windows you can secure files and registry data. On the server you can secure endpoints, logins and databases. In Schemas you can secure against types, XML schema collection, objects, etc.

In the table below is a list of securables at the Windows, SQL Server and database levels.

Principals	Securables
Windows level Windows Group Windows Local Login Windows Domain Login	 Files Registry
SQL Server Level Fixed Server Role SQL Server Login	**Microsoft SQL Server** SQL Server Login Endpoint Database
Database Level Fixed Database Role Database User Application Role	**Database** Application Role Assembly Asymmetric Key Certificate Contract Full-text catalog Message type Remote service binding Role Route Service Symmetric key User Schema

Figure 13: Principals and Securables

Principals are given access to securables by permission statements. SQL Server permissions are:

- GRANT – grants a principal access to an object.
- REVOKE – Revokes a principal access to an object.
- DENY – a deny generally overrides a grant.

You can determine what permissions are given to you with this formula:

```
  All permissions granted to you
+All permissions granted to public
+All permissions granted to each role you're a member of
 -All denies
----------------------------------------
=Permissions available to you
```

If you '**Grant control**' at the database level, this gives you permission on all securables as well as all schemas. If you 'Grant select' permission on a schema, then select permissions are given on all of the objects.

Basic Security Steps

1. You must have a valid SQL Server login.
2. You must be added as a user of the database.
3. You must have permission to all objects.

DBO - User

The system administrator and sysadmin roles are the **Database Owners** (DBOs) in all databases and anything the sysadmin creates belongs to the DBO automatically. You cannot delete the DBO user.

Guest User

Guest user exists in all databases but is disabled by default. If you enable guest user then all SQL Server logins will be allowed access without having to create a user. The guest user permissions include permissions that are specifically granted plus all permissions granted to the public role.

Understanding Roles

Just as there are server roles -- there are database roles also. You can have a fixed database role like you have a fixed server role, but in a database, you can also have user-defined roles and application roles. These roles are database specific and you will want to use roles when more than one user needs the same set of permissions.

Roles can be used to create logins for individual members or you can create a login for a Windows Active Directory. With a Windows Active Directory, members can come and go from the group without having to add them and subtract them from the role. They are included because they are a member of the Windows Active Directory group.

A user can be a member of many roles. In roles permissions are accumulative minus any denies. A role may also be a member of another role.

Once a role is created and permissions are assigned to that role, the permissions can't be changed. You can add new members after the role is in use but you can't change permissions.

The public role works opposite of other roles in that with the public role you can't assign membership (because everyone is a member), but you can assign permissions.

Here is the syntax to create a role using T-SQL:

```
CREATE ROLE role_name [ AUTHORIZATION owner_name]
```

Roles are securables at the database level. Once a role is created, you can configure the database-level permissions by using the GRANT, DENY and REVOKE statements.

To add members to the role you can use the stored procedure:

sp_addrolemember role_name, user_name.

Roles should be used when more than one user needs the same permissions. For instance, you could have a Teacher role and a Student role. All of the teachers (users) in the Teacher role would have the same permissions granted. All of the students (users) in the Student role would have the same permissions granted to them, but the student's permissions would be different than the permissions granted to the teachers.

The teachers could also be members of the Student role and have access to the permissions granted to the students in addition to the permissions granted in the Teacher role.

Here are the steps to use to create a role:

1. Analyze your database security needs
2. Create a role to group individual users who have the same permission requirements
3. Add those users to the role and grant the permissions

Schemas and Application Roles

A schema is a namespace that contains a collection of objects that are scoped within a database. The objects in a schema are not exclusively owned by the schema. Objects within a schema may have many owners and an owner may own objects within several schemas.

server.databasename.schema.object

To create a schema:

- **Select the database where you want the schema to exist:**

  ```
  USE database1
  ```

- **Create a login:**

  ```
  CREATE LOGIN user1
        WITH PASSWORD = 'password',
        DEFAULT DATABASE = database1,
        CHECK_EXPIRATION = ON
        CHECK_POLICY = ON
  ```

- **Create a user:**

  ```
  CREATE USER user1
  ```

- **Add the user as a member of the database data reader role:**

```
EXEC sp_addrolemember 'db_datareader', 'user1'
```

- **Create the schema which is specific to the database:**

```
CREATE SCHEMA schema1
CREATE SCHEMA schema2
```

- **Then you can create objects within the schema (like a table):**

```
CREATE TABLE schema1.table1
        ID (varchar(100) NOT NULL)
CREATE TABLE schema2.table1
        ID (varchar(100) NOT NULL)
```

- **Now to associate a user with the schema:**

```
ALTER USER user1 WITH DEFAULT_SCHEMA = schema1
```

The user identified by 'user1' is an owner of the schema 'schema1'.

Notice that there are two tables with the same name, 'table1', one exists in schema1 and the other in schema2. You can do this with schemas because the fully qualified name includes the schema name. If you don't fully qualify an object name, if you use **SELECT * FROM table1** for instance, SQL Server will first check the users default schema for the object. Then if the object is not found in the user's default schema, SQL Server will look for it in the DBO schema. And if the object is not found there, you'll get an error.

Application Roles

Application roles are associated with applications and are activated when a user runs those applications. This allows a level of protection where the application can have permissions that are not necessarily granted to the user.

With application roles, a user has access to a database through an application. The application has permissions assigned to it and the user inherits those permissions through the application. Therefore, any user who runs an application that is assigned an application role can access data according to the permissions of the application - rather than the permissions assigned to the user.

Application roles have no members and are activated when a user runs the application. They can't access metadata and can only access databases as a guest.

Once an application role is evoked, all connection permissions are replaced by the application role permissions. A new feature that has been added to SQL Server 2008 allows you to turn on and turn off application roles. Application roles can't be called from stored procedures and they are database specific. You create an application with T-SQL as follows:

```
CREATE APPLICATION ROLE role_name
        WITH PASSWORD = 'password' [ , DEFAULT_SCHEMA = schema_name ]
```

CLR Security

Common Language Runtime (CLR) security is defined in three locations:

- **Machine Policy** – policy set for managed code on the box that SQL Server is on.
- **User Policy** – policy set for the user SQL Service is running under.
- **Host Policy** – policy set for the host SQL Server.

The effective permissions are only where the three intersect and not the accumulated sum of the three. For instance, permissions granted in the machine policy may be limited by the user policy. It's only where the permissions in all three policies agree that permissions are granted.

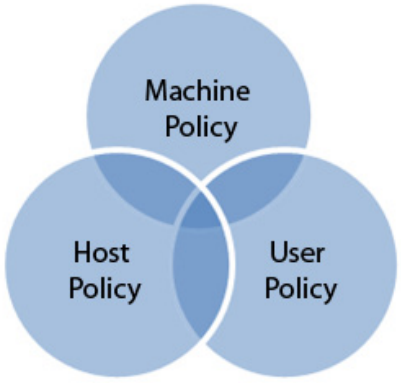


Figure 14: Understanding CLR Security

By default CLR has minimum permissions assigned under the permissions of the SQL Service Account. Permission Sets:

- **SAFE** – the preferred permission setting. By default the CLR runs under the permissions of the SQL Server Service account so you should make sure that this account has minimum permissions. SAFE permissions don't allow access to resources external to the instance of SQL Server like files, environmental variables, the network or registry.
- **EXTERNAL_ACCESS –** choose this if you want the CLR assembly to have access to external resources not available with SAFE permissions.
- **UNSAFE** – have unrestricted access to resources both within and outside an instance of SQL Server (only **sysadmin** can register UNSAFE assemblies).

Shared State

When using CLR, shared state refers to information shared between user sessions. This can cause problems. And for this reason, shared static variables or data members are not allowed to be updated for the **SAFE** or **EXTERNAL_ACCESS** permissions. Read-only static members, however, are allowed for **SAFE** and **EXTERNAL_ACCESS**.

Service Broker

Service broker applications send, receive and process messages. These messages constitute a conversation between applications in two databases. Service Broker maintains the queues for send and receive data at both the sending and receiving database. This feature is built into SQL Server 2008 and it allows asynchronous processing of information (messages) through those queues.

This information passes through the endpoints and is transported via TCP/IP connections.

Service Broker security uses certificates to establish credentials to map operations from a remote database to a local user and to establish credentials at remote databases. The certificate is shared between databases.

These conversations can be used to automate processes like changing the content of tables in a remote database.

There are two types of security for Service Broker:

1. Dialog Security
2. Transport Security

Dialog Security

Authenticates and encrypts data between services which protects in-transit data from someone seeing it or making changes. Dialog security is optional and is configured at the service level using certificates.

Transport Security

Transport security is supported by Service Broker and applies to all messages passed between two servers. It blocks messages from unauthorized databases and detects alterations that occur to messages in-transit. You can place point-to-point encryption between servers and it uses certificates or Windows Authentication based on the endpoint authentication option.

Certificates and Service Broker

For Service Broker you make certificates **ACTIVE FOR BEGIN_DIALOG;** that makes the certificate available to Service Broker. Service Broker also uses the valid time references in the certificates and will deny access if the time has expired.

When creating a certificate for Service Broker, the key modulus must be less than 2048. The key length must be less than 32K and must be a multiple of 64 bits.

Designing a Security Policy and an Audit Plan

Policy Based Management

Policy Based Management (PBM) is new to SQL Server 2008. PBM allows administrators to define rules and enforce those rules on one or more SQL Servers. This feature makes the administrator's job easier, allowing them to manage one or more servers and keep the servers in compliance with the policies. You can set up a policy to prevent some things from occurring that violate your policy. And you can potentially fix other things with the policy by changing a database option or server option so that it complies with your policy. This can be done either locally or across multiple servers.

You can create either required policies or optional policies on a database. And you can create SQL Server Agent alerts that occur on policy failure.

In your policies you can set up conditions such as a requirement that every database has to be backed up weekly or enforces naming conventions on databases.

The first thing you do for PBM is create a policy. You can choose a target and a policy to check or force compliance to that policy.

A target has a type like:

- Table
- Surface Area Configuration
- Server
- Database
- Index

Each target has a set of properties that you can check or set, each of these properties are defined as a **facet**. Here are some of the facets that are available with Surface Area Configuration:

Surface Area Configuration
DatabaseMailEnabled
SQLMailEnabled
XPCmdShellEnabled
CLRIntegrationEnabled

Figure 15: Surface Area Configuration Facets

You can set a condition from a set of facets and from a specific target type. The condition is the thing that you wish to be true.

If the condition is violated you can set the policy to respond in two ways:

1. Check and log the violation (uses event notifications)
2. Prevent the violation

Evaluation Modes

You can specify demand-only when specifically requested, on demand with no alerts or on change: prevent (uses DDL triggers). Prevent can only work if the change or facet can occur within a transaction and can be caught and rolled back by a DDL trigger.

In SQL Server Management Studio you can set up the policies -- you can also:

- Evaluate the policy item
- Evaluate and report only
- Apply (This will change the items that are fixable to come into compliance with your policy)

Selecting Permissions

Permissions are stored in a bit mask integer. In this integer each bit can enable (bit =1) or disable (bit = 0) a permission.

The function **sp_helprotect** returns information on permissions that are granted in SQL Server. The parameters for **sp_helprotect** are **object_statement**, **security_account**, **grantor**, and **type**.

If you execute **sp_helprotect** without any arguments, you get information on all of the permissions granted as shown below:

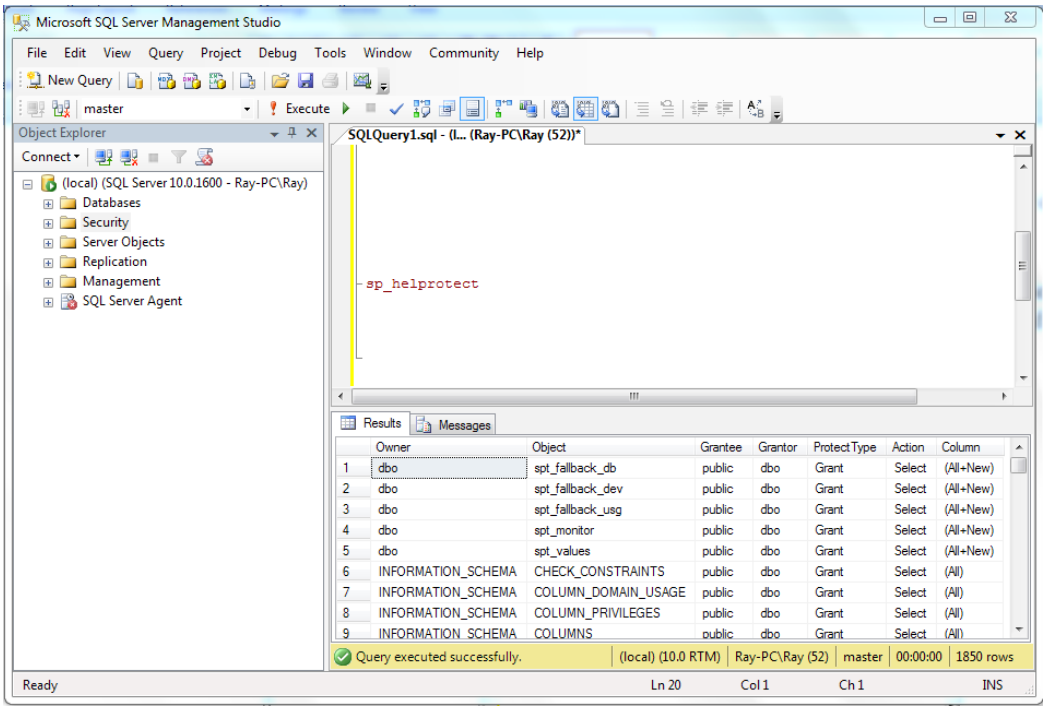


Figure 16: The sp_helprotect Function

You can view the permissions granted to the current user as an integer using the T-SQL - **SELECT PERMISSIONS()** - and as shown below, you can check for individual permissions as well. The integer returned will represent the bit mask described above.

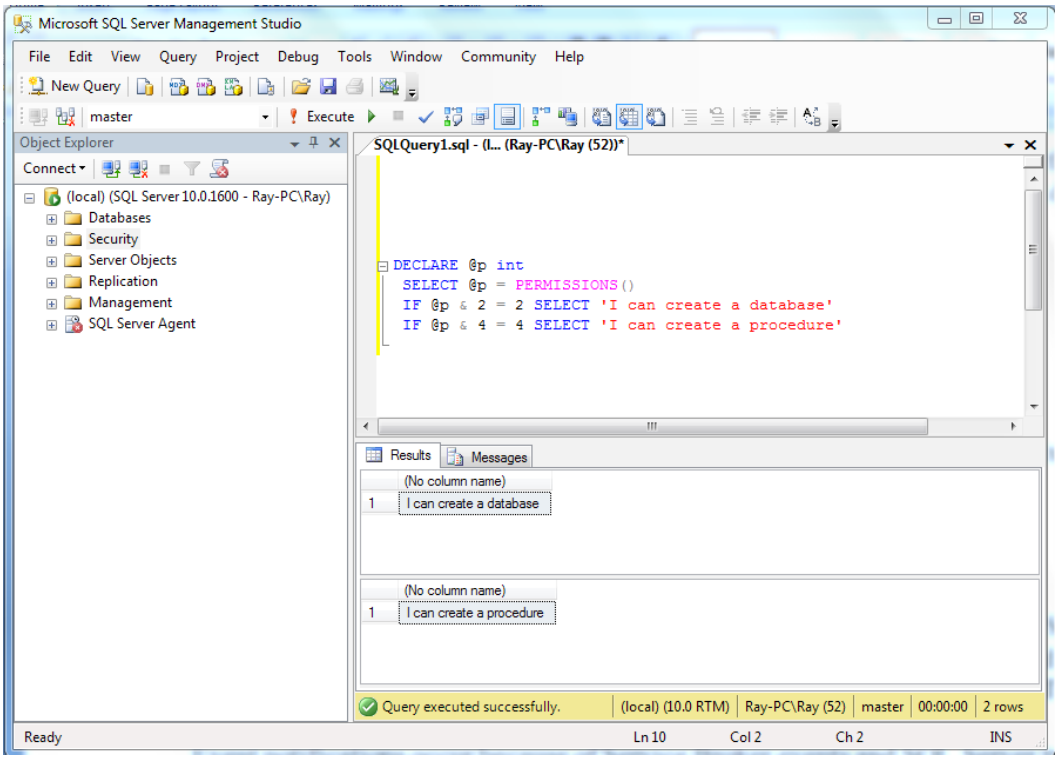


Figure 17: Selecting Permissions

To view a list of permissions for the current user in the server, execute the following query:

```
SELECT * FROM fn_my_permissions (NULL,'SERVER') ;
```

Permissions for the database Adventureworks:

```
USE Adventureworks;
SELECT * FROM fn_my_permissions (NULL,'DATABASE') ;
```

Event Notifications

Event notifications send information to a Service Broker service in response to a variety of T-SQL DDL statements, DML (Data Manipulation Language - Transact-SQL) items and SQL Trace events. They allow you to capture and respond to these events as they occur on SQL Server. They occur outside the transaction unlike DDL triggers so they don't use transaction resources.

You can use event notifications as an alternative to DDL triggers and SQL Server trace. Event notification is asynchronous in processing with the message so it will scale higher than DDL triggers.

They can notify you when an event occurs like creating databases or tables.

Extended Events
Extended events like other events can be captured and logged. SQL Server Extended Events work with **ETW** (Event Tracing for Windows).

Designing an Encryption Strategy

Transparent Data Encryption (TDE) is new to SQL Server 2008 and encrypts data, transactions logs and backup files at the database level. If a database has TDE active, tempdb will be encrypted along with the database. This could produce a performance issue (Microsoft says that the overall performance impact is 3-5%).

Note: FILESTREAM data is not encrypted with TDE.

If you are using replication it may require additional steps. You may want to encrypt the distributions or channels. You may also want to encrypt the BCP (Bulk Copy Program) intermediate files.

When you encrypt a database, the backups are also encrypted.

When you are encrypting you must keep a certificate for each database backup. If you lose the certificate then restore or attach is not possible.

Data Encryption

There are cases where you won't need to encrypt an entire database (for example, if a table includes a customer's credit card information or personal data). In this case you may only need to encrypt the columns with the sensitive data.

With **DE** (Data Encryption) you can encrypt data down to the column level rather than the entire database. You can hide the data using a password or a key using:

- Symmetric keys
- Asymmetric keys
- Certificates
- T-SQL

For best practice you will want to use symmetric keys because they are faster than certificates or asymmetric keys. Just remember, asymmetric keys have higher security than symmetric keys, so the tradeoff for speed is that the data is less secure.

Special Considerations for FILESTREAM

Since FILESTREAM is not supported by TDE, you have two options you can use to encrypt FILESTREAMS:

1. Use DE by columns to encrypt the varbinary columns.
2. You could place the FILESTREAM on an encrypted Windows drive.

Domain 3: Designing a Database Solution for High Availability

In this section we will turn our attention to high availability in planning and designing a database solution. An understanding of this section can help you increase overall availability and decrease time that data is unavailable to users. You will learn how to manage failovers in case of system or disk failure, how to uses things like replication, mirroring and log shipping in your database design for increased availability.

Designing a Failover Clustering Solution

When clustering is implemented, two or more servers present themselves to a network as a single virtual server. Simply stated, clustering allows one physical server to take over the responsibilities of another physical server in the event of a failure. If one server fails, another server can take over without the user noticing the problem.

Each server in the group is generally referred to as a node. The maximum number of cluster nodes that may be supported on SQL Server Enterprise is 8.

You can select one of two types of clustering:

- Active/Active – both servers on one machine.
- Active/Passive – once server on each machine in the group of two or more.
- You will use clustering when you want to protect an entire server or when you want to provide high availability.

A cluster can be administered from any node (or server) in the cluster and you can failover to any node as well, automatically or manually. Although clustering can protect an entire server, it can't protect from individual disk failure, so you should provide disk protection using RAID.

Clustering also provides a virtual server name and virtual IP address so that clients can be redirected automatically in case of server failure. It also supports full-text search and both the SQL Server engine and the **SSAS** (SQL Server Analysis Services) can be clustered.

Note: The maximum number of SQL Server instances per cluster is 25.

The Clustering Resource Group

In each cluster created, there is a single resource designated as the quorum resource that maintains the data necessary for the recovery of the cluster.

The cluster log and state information is vital, therefore access to the quorum resource is necessary for a node to form a cluster. And communication with the quorum resource is necessary for nodes to remain in an existing cluster. This information is node independent and should be on a separate disk – 500MB or larger.

Cluster Setup Considerations

When designing a cluster configuration the first thing to consider is how many nodes you want to include in the cluster. SQL Server Standard supports two nodes in a cluster, Enterprise supports up to eight. You will also need to install **MSDTC** (Microsoft Distributed Transaction Coordinator) which is required for the following:

- Distributed transactions
- **SSIS** (SQL Server Integrated Services)
- Workstation components

Here are some dos and don'ts for using cluster setup:

1. You cannot compress the SQL Server disk for clustered instances or setup will fail.
2. You should disable NetBIOS on the private network cards.
3. You must be the local administrator in order to install clustering.
4. Clustering cannot be installed on a domain controller.
5. DO NOT install antivirus software on clustered instances.

When using MSCS you must have a trusted connection and the connection should not be in the local Administrator group.

Clustering supports rolling upgrades, meaning you can apply a service pack to a node in a cluster and then failover and apply that service pack to another node. As mentioned earlier, failovers can be either manual or automatic.

To failover manually you can either pull the plug on a hard drive or pull the plug on the heartbeat. Either method will cause the cluster to failover.

Designing Database Mirroring

Clustering, log shipping and database mirroring each protect the data in a database in different ways. Clustering provides a way to fail over one server to a backup server in case of a disaster. Log shipping periodically ships the transaction log of a database to one or more secondary server instances. A manual failover to a secondary server is required in case of a disaster.

Mirroring maintains an up to date backup copy of a database that can be used in case of a failure.

Clustering	Protects an entire server
Log Shipping	Ships the log of one database
Mirroring	Keeps a mirrored copy of an entire database

Figure 18: Comparing Backup Types

Database mirroring, like disk mirroring, allows you to keep a hot or warm standby for a database. With mirroring there will be an exact copy of a database maintained on a different server which will increase availability.

The primary database and the mirror database must be on different instances of SQL Server. The principal server will contain the primary database and the mirror server will have the copy of the protected database that's on the principle server.

You can also have an optional witness server. The witness server will monitor activity and provide automatic failover if you choose automatic failover.

You can have client failover as long as you provide a connection string that identifies the mirror partner.
Note: The databases that will be mirrored must use the full recovery mode (Discussed in Domain 4: Designing a Backup and Recovery Solution).

Note: You cannot mirror any of the system databases.

When Should You Choose Mirroring?

- Mirroring is a less expensive alternative to clustering. So you may want to use it if you don't want to or can't invest in the hardware required for clustering.
- Mirroring can be easier to implement and maintain than clustering.
- Your needs for high availability may only apply to a single database and not the entire server. If this is the case, mirroring would work better for you than clustering.

How Mirroring Works

Remember with replication, every transaction log record is sent to a distributer and then distributed to all the subscribers. With mirroring, every transaction log record is sent from the principal -- directly to the mirror. The mirror then replicates each insert, update and delete in the same order as the principal.
If you want to give clients the ability to failover you have to set the mirror up with a witness and in the high safety mode. And the client connection must specify a failover partner.

About the Mirror Database

You can create the mirror database using RESTORE WITH NORECOVERY. Therefore you won't be able to access the database directly; a snapshot of the mirror can be created however.

Note: The database name of the mirror has to be the same as the principal database name.

Mode Benefits

There are three modes available with mirroring:

- High availability mode
- High protection mode
- High performance mode

Each mode comes with an advantage and a tradeoff as shown below:

MODE	Automatic Failover	Full Protection From Data Loss
High Availability Mode	X	X
High Protection Mode		X
High Performance Mode		

Figure 19: Comparing Mirroring Modes

To set up for the high availability mode or the high protection mode, you set Safety = full in the setup. If you want automatic failover (high availability mode) you must provide a witness server.

High Availability Mode Description

With high availability, a transaction written to the mirror server is verified by the principal server before the next transaction. So every transaction is guaranteed before the principal server proceeds with the next. If a write fails on the mirror server, then it can't complete on the principal. This guarantees no loss of data but it comes with a performance cost.

High availability also provides automatic failover. In this case you will need to supply a witness server that will serve as a monitor and performs the failover. When the principal goes offline, the witness fails over to the mirror. Then when the principal comes back online, it becomes the mirror and the original mirror is then the principal.

Note: This mode is best when you have fast, reliable network connections.

High Protection Mode

This mode is the same as the high availability mode without the witness. In this mode you get full protection but you don't get automatic failover -- failover is manual.

High Performance Mode

With high performance, data is written first on the principal and then the mirror. But the difference between high availability and high performance is this: With high performance the primary continues the next transaction without confirmation from the mirror that the data was written there.

This mode works faster but with some risk of data loss and you can't get automatic failover whether you have a witness or not. You cannot do a manual failover in this mode. You can do a forced service failover but you could potentially lose data.

Syntax for forced service:

ALTER DATABASE A SET PARTNER FORCE_SERVICE_ALLOW_DATA_LOSS

The high performance mode is usually chosen when the primary and mirror servers are separated by a long distance. In this case network glitches could cause frequent failovers if you use automatic failover. An alternative to the high performance mode would be log shipping where you would have more control.

Steps to Mirroring

1. Set up communication between the primary and the mirror.
   a. For trusted servers - create a login and grant CONNECT on the endpoints.
   b. For servers not trusted – create a certificate and make the certificate active for Begin_Dialog.
2. Create endpoints (Must be done by the sysadmin).
3. Set recovery mode to full on the primary.
4. Backup the database and log, then restore them on the mirror. (Database on the mirror must have the same name as the original with the NORECOVERY option.)
5. Make a plan to copy logins, jobs, alerts etc.

Suspend vs. Stop

If you have a failure such as the witness going away -- you have to make a decision to suspend mirroring or stop mirroring altogether.

If you suspend mirroring your transaction logs will no longer truncate and you must resume quickly. If your transaction log will likely fill up before you can complete the repair you may want to stop mirroring instead.

If your transaction log fills up while you are fixing the problem, you can resume mirroring and let it catch up or you can stop mirroring.

Note: If you stop mirroring at any point, you will have to redo the mirroring from scratch.

Automatic Page Repair

Automatic page repair detects torn pages and repairs them. Once a torn page is detected, then a request is automatically made from the other partner and the page is repaired on the primary server.

Automatic page repair can fix **CRC** (Cyclic Redundancy Check) errors and logical errors (torn page, bad checksum and pages marked as restore pending).

Snapshots for Reporting

Because the mirror database in the recovery mode, you cannot access it directly. But what you can do is create a snapshot against the mirror for reporting. A snapshot is a picture of the database at one point in time (the time of the snapshot).

Note: You must have a plan to recreate the snapshots periodically.

Problem: How do you create a connection that will connect to the most recent snapshot?

Solution: If you use the database name (or some other name) for the snapshot then you will have to delete the previous snapshot and give the same name to the new snapshot. Then you have only one name to reference.

Designing a High-Availability Solution Based on Replication

We have discussed clustering which protects an entire server and mirroring, which protects an entire database. Replication is even more localized and can protect individual tables in a database. Increased availability is achieved by replication at the database or table level.

Replication increases site independence by making local copies of data on each site. If one site goes down the other sites are not affected. This is accomplished through data distribution from a publisher to all subscribers of that data. Let me explain…

In a replication setup you have a publisher, a distributer and one or more subscribers. The publisher is the source of the original data and the distributer handles the distribution of the data to the subscribers and the changes from the subscribers. This data is in the form of multiple small sets of logically connected information -- like sales records for several retail stores.

Simply stated and simply illustrated below: The publisher originates and maintains the source data and sends changes to the distributer. The distributer receives those changes from the publisher and delivers the changes to subscribers. And finally the subscriber database receives the changes and keeps a current copy of the data.

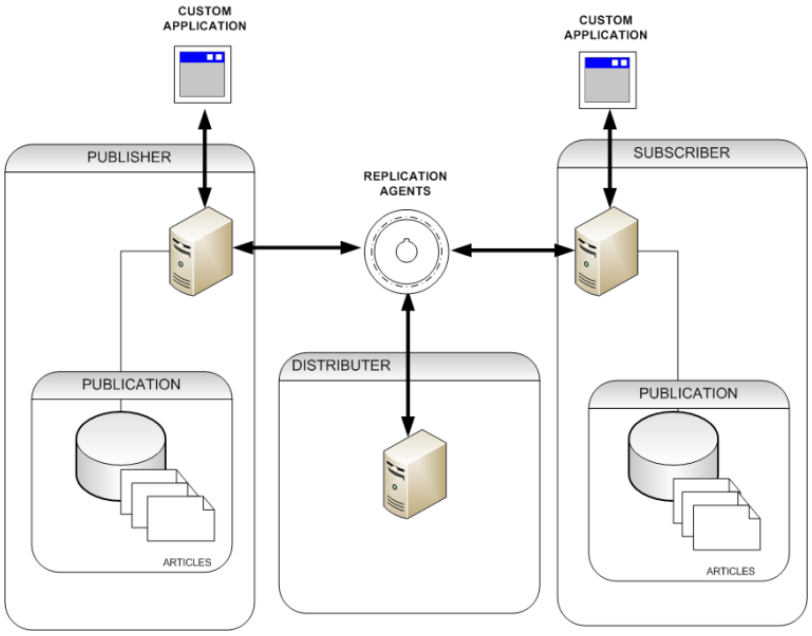


Figure 20: The Replication Process

You can have many publications in a database and publications are what you replicate. A publication can be one or more articles.

For instance, you could have an employee database stored on the publisher. One person may need basic information from the employee database such as contact information. Another person may need more detailed information like pay scale and current salary.

An article can be a table, a subset of a table, a view or a user defined function. An article can be some or all of the columns, some or all of the rows. You can have customized filtering for an article such as in the case of an employee database, only employees in a certain department or managers only.

A publication is a collection of one or more articles and is the unit of replication. Subscribers will subscribe to an entire publication but not an article. Subscriptions can be pushed which means the publisher creates the subscription. Or they can be pulled, which means that the subscription is created by the subscriber.

You can also replicate stored procedures. For instance, if you wanted to adjust the pay of all your employees for a cost of living increase, rather than make the changes and have the tables with the changes distributed, you could create a procedure that would adjust the pay and replicate it to the subscribers of that particular publication (all of your stores for instance).

Replication can be set up and managed in SSMS using the **Replication** tab in Object Explorer.

The Distributer Agent

The distributer reads from the distribution database and delivers the data to the subscribers. It does all of the work for the subscribers. And push adds even more work because the distributer has to push to each subscriber.

For high scalability you can have the subscribers share the workload of the distributer by having the distributer agent run on each subscriber. This way each subscriber does its own distribution to even more subscribers. This allows you to scale replication to a much higher level.

Types of Replication

Transactional replication starts with an initial snapshot when the subscriber first subscribes to the publication and then replicates the transactions only. You can run continuous replication where the updates run real time or you can have the agent run on a schedule where updates are periodic (e.g. every 5 minutes).

With transactional replication the subscribers are allowed to update the data as well. You can select from two modes for the Updating Subscribers Option:

- **DTC** (Distributed Transaction Coordinator).
- **Queued Updating Subscribers**: In this mode a table is created on the subscriber's server. Then at sync time the publisher requests the table and the changes are uploaded. There can be conflicts if subscribers make changes to the same data.

Merge Replication allows independent changes to be made by the subscribers but you must have a plan to capture and resolve conflicts that may occur.

Snapshot Replication

Snapshot replication will refresh the entire publication on some periodic basis. Choose this type if you have major but infrequent changes and updates and they are not required immediately by the subscribers.

With snapshot replication changes made by the subscribers are not updated on the publisher or on any of the other subscribers. Changes made by the subscribers will be overwritten the next time the snapshot refreshes.

Transactional Replication

Choose this type of replication when the subscribers need minimal latency. You can also improve latency by allowing the subscribers to distribute to other subscribers as we discussed earlier. Choose this type also if the publisher performs many inserts, updates and deletes.

Transactional replication gives you every change to the record. For instance in the case of invoices, the subscriber will receive the changes like invoice opened, invoice assigned, invoice evaluated, invoice scheduled for payment, invoice paid and closed.

Where with merge and snapshot the subscriber may only get one change recorded at the time of the update.

You will need to use transactional replication if the publisher or the subscriber is not an SQL Server instance.

Transactional replication requires a reliable network. It also requires a primary key restraint on the tables. If you use identity columns, there are issues like the identity column values on the subscriber that must be dealt with. LOBs are supported here.

Peer to Peer Replication

Where transactional replication is hierarchical – peer to peer is not – everyone is a peer. This type of replication is intended to scale out reads in an environment where you can minimize writes.

Best practice is to make each node its own distributer. Objects can only be published once and nodes can be added without quiescing.

Note: SQL Server 2008 provides a GUI to set up peer to peer topology.

Merge Replication

You will want to use merge replication when the subscribers need autonomy. If they need to make changes to the data offline – you should set replication up so that few conflicts are expected. There are three ways to define a conflict in merge replication.

1. At the column level
2. At the row level
3. Logical level conflicts

Set your conflict detection at each of these three levels to catch all conflicts.

Merge works great when you have an unreliable network where you have mostly disconnected independent subscribers. It requires a unique identifier column and it must be tagged with the ***rowguid*** attribute. It also adds several new triggers and system tables at the publisher and the subscriber.

With merge replication you can have customized filtering. This allows much more flexibility in what data is offered to which subscribers.

Here are some new features available on SQL Server 2008 for replication:

- You can perform replication over secure connections like HTTPS.
- Oracle can publish in SQL Server 2008.
- You can use tracer tokens. These tokens will track the time it takes for the token to get from the publisher to the distributer to the subscriber. It provides real time latency measurement.
- In SQL Server 2008 you can also initialize from a backup.

The first step in designing replication for high availability is to decide:

- What data should be published
- Who, where and how many subscribers you will have
- And how up to date your data needs to be

You want to plan for an extreme disaster and how to get the system back as fast as possible. You also need to plan for more data to be widely available. You should also consider your network capability – Is it reliable? Is it high capacity? Is it fast?

Snapshot	Transactional	Peer to Peer	Merge
Use when you have major but infrequent changes - not required immediately by subscribers.	Use when subscribers need minimal latency. Changes are updated at the subscribers as they occur.	Use in environments where you need to scale out reads - and you can minimize writes.	Use with independent subscribers that are mostly disconnected - and you can minimize conflicts.

Figure 21: Replication Types

Work Load Considerations

It is difficult to move the distributer once it's been set up. So you must be careful to choose an appropriate distributer. If the distributer is with the publisher, it may decrease the capacity of the publisher.

The subscribers are another workload consideration. You should consider how many copies will need to be distributed. How will those copies be divided? Will you use filtering? Will your replication be done regionally or locally?

For high availability, you will want to schedule the subscriptions and do the administration. You also need to decide where the agent work will be done – on the subscriber or the distributer. You need to compare the subscription type you will be using – push or pull. Most often you will use push subscriptions but you need to consider both push and pull subscriptions.

Choosing a Solution Type

To choose a solution type you must:

- Choose a topology design.
- Pick your replication type:
    - Snapshot
    - Transactional
    - Merge
    - Peer to peer
- Choose between push or pull subscription.
- Determine how many copies or subscribers must be maintained.

Once you have chosen your solution type you should consider what requirements you will have for the distributer.

Recovering from a Replication Failure

To recover from replication failure you should sync the distributer with the backup. And optionally you can sync the publisher with the backup based on performance.

Replication failures can occur in many ways and can occur on the publisher, on the distributer or on the subscriber. For instance, if the publisher fails, when you restore it, data may be lost and it may fall behind the distributer. You need to learn about each type of failure that can occur and plan how you will recover from each one.

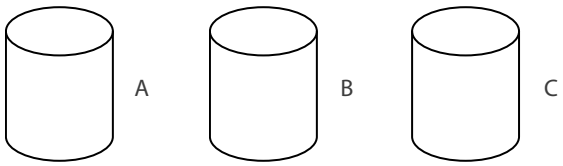
Note: Make sure you script everything during the initial setup. This will make recovery much easier.

Peer to Peer Recovery

Peer to peer replication is transactional replication without the hierarchy. Every database is a peer.

The Problem: If one peer drops out for some reason while the others keep running. The updates will fall behind.

The Solution: Use all the available data to sync all the databases and restore any lost transactions.



In the example above, there are three peers (A, B and C). If peer B drops out then the recovery steps are as follows:

1. Run the distributer to sync A with C – A and C are now in sync.
2. If B is available -- run the distributer to attempt to sync B with A and B with C – This makes sure that you recover as much data as you can from B and B is in sync as well.
3. Delete the replication information from B. Use the stored procedure **sp_remove_distributer_db**_replication. This takes B completely out of the peer to peer relationship with A and C.
4. Backup database A.
5. Restore it to B.
6. Recreate the publication on B and the subscriptions at A and C.
7. Run the distribution agent to sync B with A.
8. Reseed the identity columns in B.
9. Recreate the subscription from B to C and from C to B.
10. Then run the distribution agents.

In doing this you will make sure that A and C is in sync and B has as much data as possible and is in sync with A and C to the greatest extent possible.

Synchronization

Synchronization occurs immediately after initialization. It is the process of propagating all data and schema changes from the publisher to the subscribers.

Transactional synchronization runs continuously and all the insert, update, delete and schema changes are applied to the subscriber after they occur on the publisher.

Snapshot synchronization occurs on a schedule. During a sync, all of the data is copied from the publisher to the subscriber and the schema is refreshed if there are any changes.

For merge synchronization the data changes are replicated on demand – when the subscriber requests it. Changes are uploaded from the publisher to the subscriber. Then the subscriber changes are downloaded to the publisher where conflicts are detected and resolved.

Monitoring Replication

You can use the System Monitor to watch things like transactions per second and latency. You also have the SQL Server Agent alerts for replication. And in SQL Server Management Studio there is a replication monitor tool.

Validation

Validation compares the data on the subscriber with the data on the publisher to validate the published data. There are two types of validation you can choose:

1. Count the rows in each table
2. Binary checksums for each row in the table

Designing a High-Availability Solution Based on Log Shipping

Another way to increase availability of your servers is to use log shipping. Where clustering protects an entire server, you can use log shipping to ship the transaction log of a single database from a primary server (production server) to one or more secondary servers at regular intervals. The primary goal of log shipping is to provide security around the file system.

There are three operations that occur during a log ship:

1. The transaction log is backed up on the primary server instance (job 1).
2. The transaction log is copied to the secondary server instance (job 2).
3. The transaction log is then restored on the secondary server instance (job 3).

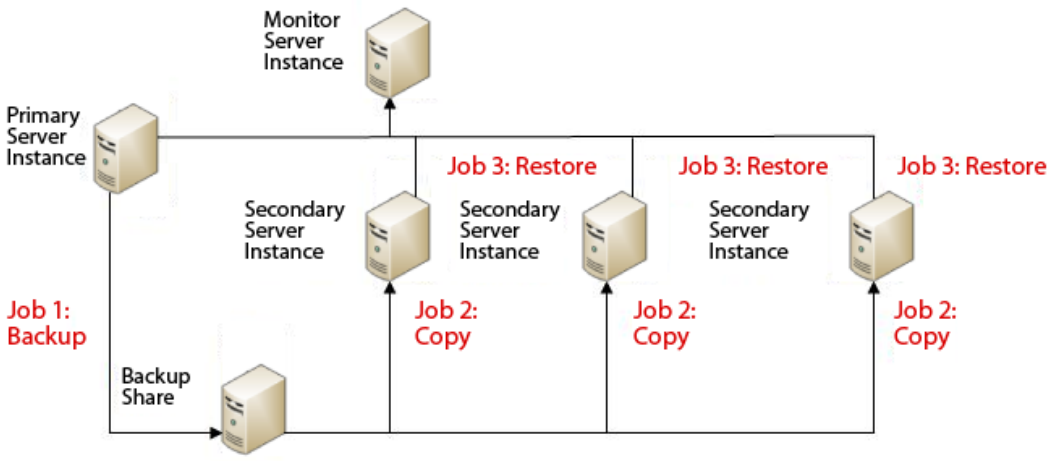


Figure 22: Understanding Log Shipping

If there are more than one secondary server instances, steps two and three are repeated for each additional secondary server instance.

There is an optional third server instance known as the *monitor server*. This optional server instance records the history and status of the backup and restores – it can also raise alerts if there are failures during the any of the scheduled operations.

Unlike clustering, log shipping configuration does not automatically failover. If there is a failure of the primary database and it becomes unavailable, the secondary database(s) must be brought online manually.

All administration of the log shipping configuration is performed from the primary database using SQL Server Management Studio. And you must use the full or bulk-logged recovery model on the primary database.

To initialize the secondary database you should restore a full backup of the primary database using either the NORECOVERY or STANDBY option.

The monitor server continuously tracks the details of log shipping, including:

- When the last backup occurred.
- When the last copy and restore occurred on the secondary server instance.
- Any backup failure alerts.

In the illustration below you see that the primary server instance performs job number one, the backup. The backup share is available to everyone and the secondary servers perform job two, the copy job.

Then the secondary server instances also perform job three and restore the transaction log for the final step. All the while the monitor server instance is watching and recording the events and any failures that may occur.

Interoperation Requirements

Log shipping interoperates with mirroring with one restriction; the log ship principal has to be the same as the mirroring principal. Log shipping also interoperates with replication. Replication will stop however, when the log ship fails over.

Switching Roles

One major distinction between clustering, mirroring and log shipping is that with log shipping you have to manually switch roles after a failover. To switch roles you should:

1. Copy and restore any undelivered logs.
2. Perform a tail-log backup on the primary and restore it on to the secondary server.
3. Run recovery on the second database -- the syntax for this is RESTORE mydb WITH RECOVERY.
4. Transfer the metadata across. Make sure you bring all the necessary information (anything that the database depends on) across, like -- logins, SQL Server Agent jobs, endpoints and messages.
5. And finally -- disable the log shipping jobs.

Once you have completed the log ship failover and you've switched roles you will need to redirect your clients to the new server.

When you fail over to a secondary server instance there are some instance level objects that you will need to manage. Here is a list of the objects that need to be consistent between the databases for a successful failover:

- Server configuration
- Any credentials associated with the database
- Cross-database queries
- Encrypted data and the database master key
- User defined error messages
- Event notifications
- Extended store procedures
- Full-text
- SQL Server agent jobs
- Logins
- Permissions
- Replications
- Server Broker applications
- Startup procedures
- Server triggers

Reinitializing the Secondary

To reinitialize the database after failover you can simply drop the database and recreate the log shipping. You can back up the database and the log -- then restore the database with no recovery or standby – then restore the logs with no recovery or standby as well.

Using Log Shipping for Reporting

You can set up a secondary log shipping database for reporting. To do this you need to leave the database in the Stand-by mode. This allows read-only connections to the database for reporting.

When a log restore occurs the users must be disconnected or log restore will wait until there are no users before it proceeds.

Consistency Check

You can check consistency on the secondary rather than the primary. This will take some of the work load off of the primary. To do this you can run **DBCC** (Database Consistency Checker) on the secondary. If the DBCC fails, then run it on the primary to see if the problem originated there, if so, it must be fixed on the primary.

Monitor Server

As I mentioned earlier, the monitor server is optional for log shipping. You can retrieve some valuable information, however, if you choose to use it.

The monitor server monitors when logs are backed up on the primary and when logs are copied to and restored on the secondary. You can set up some alerts for failure notification as well. These alerts work with SQL Server Agent job notification and must be set up when you implement log shipping.

You can get reports using SQL Server Management Studio or you can use the stored procedure:

SP_HELP_LOG_SHIPPING_MONITOR.

Monitor Server Tables

There are some tables that support the monitor service that you can access:

- **Log_shipping_monitor_alert** – has the SQL Server Agent job ID of the log shipping alert job.
- **Log_shipping_monitor_error_detail** – this table stores the error details for log shipping jobs. You can query this table to see any errors that may have occurred for an agent session.
- **Log_shipping_monitor_history_detail** – this table store the log shipping agent history information. You can query this table as well.
- **Log_shipping_monitor_primary** – stores one monitor record for the primary database in each log shipping configuration.
- **Log_shipping_monitor_secondary** – stores one monitor record for the secondary database including information about the last backup file and the last restore file.

Domain 4: Designing a Backup and Recovery Solution

Designing a Backup Strategy

Planning for database recovery not only accounts for the largest portion of this test, it is also important to have a working plan for recovery or vital data could be lost forever. In this section you will learn strategies for backup and restore, how to design for various types of failure and recovery solutions and how to design for minimum loss during catastrophic system failures as well.

Recovery Models

There are three recovery models we will discuss in this section: simple, full and bulk-logged.

Simple Recovery

With the simple recovery model, the transaction log is automatically truncated so it will not fill up. One catch here is that the truncating will not start until you do your first full backup. So you could have a situation where your transaction log file keeps growing and doesn't truncate – the solution is simply to do a full backup of the database.

Since the transaction log is automatically truncated, changes that occurred since the last full backup will be lost when you recover.

Choose simple recovery when point of failure recovery is not necessary. For instance, in a situation where you have a read only database where you can risk the loss of data from the last full backup. If the changes are easy to redo or there just aren't many changes, you might choose this recovery mode.

Some other factors that you might consider in choosing simple recovery are if you don't have the resources (staff, time or training) to backup and restore the transaction logs.

Full Recovery

Full recovery allows data to be recovered without work loss (no committed transactions lost). In the event of a catastrophe, you can insure full recovery by keeping appropriate log backups, mirroring the log, and keeping the master, data and log physically separated – and when there is a failure, you immediately do a full tail-log backup.

Choose full recovery if you want point-in-time recovery or if you want to restore individual pages. Of course you have to be willing to bear the additional administrative costs of the transaction log backup to use this mode.

Bulk-Logged Recovery

Bulk-logged recovery mode is designed to be used with full recovery mode. If you are doing bulk-logged inserts and you don't want to fill up the transaction log -- you can switch to bulk-logged recovery mode, then perform the bulk inserts and switch back again to full recovery mode and back up the transaction log.

This mode does require transaction log backup but it minimizes required log space. The down side is if a bulk operation occurred since the last backup, you could lose everything that occurred since the last full backup.

There is no point in time recovery in this mode. You will use this mode when you need minimally logged bulk operations and you can disconnect the users during the backup or accept the possible loss of data. Here are the steps for bulk-logged recovery:

1. Use the full recovery mode.
2. Perform an extra transition log backup.
3. Switch to the bulk-logged recovery mode.
4. Perform the bulk-logged operations.
5. Switch back to the full recovery mode.
6. Perform another transition log backup.

Once you switch to bulk-logged recovery mode, you are at risk of losing your data until you switch back to full recovery mode and do a transaction log backup.

Backup Compression

There are many third party vendors that supply good applications backup compression for SQL Server. SQL Server Enterprise edition, however, provides built in backup compression.

Note: All editions of SQL Server can restore compressed backups.

FILESTREAM is fully supported for backup compression as well.

Backup compression is off by default – you can turn it on with sp_configure **'backup c ompression default' 1**.

The system performance cost of compression is that it uses lots of CPU time compressing and uncompressing the data. You can use Resource Governor to allocate CPU resources to compression and limit the amount of time the CPU will spend actually doing the compression.

Compression Benefits
Compressing backups may not produce any benefit if your backup is encrypted or compressed already. Types of data that compression benefits:

- Text data compresses well.
- So does data where values are duplicated several times.

If a page contains a single row and not much duplication, compression doesn't help much.

Backup Methods

In this section we will discuss the backup methods available in SQL Server. Here is a list:

- Full database backup
- Differential backup
- Transaction log backup
- Tail-log backup
- Partial backup
- Partial differential backup
- Copy-only backup

You can backup a file or a file group using full or differential backup.

Full Backups and Restores
The most common type of backup and restore is the full backup and restore.

You should put your database backup on a regular schedule and automate it. You can use the SQL Server Agent or the database maintenance wizard for this.

Backup the master database as well as other system databases on a daily schedule. If you have to do any intensive system work, you will want to back up the databases before you start.

You should backup the **msdb** database on schedule with your user databases. This database is used by SQL Server Agent, Service Broker and Database Mail.

Make sure you backup databases that you created, immediately after they are created and initially loaded. You might want to backup the user databases after you create a large index or after reorganization.

If you have to truncate the transaction log for any reason, you will want to do a full backup at that point.

Note: The full database backup captures the state at the end of the backup, not the beginning.

Backup Media

Backing Up to Disk
To backup a disk you must backup to a local disk and you may use a **UNC** (Universal Naming Convention) name format - **\\servername\share_name** as well.

You have to have permission to backup to the network shares and you should verify your backups.

Backing Up to Tape
In order to backup to a tape drive it must be a local drive (You can't share tape drives). Tape backups record the name, time, date, and data type of the backup.

Tape backups use the **Microsoft Tape Format** (MTF) to record the data. In earlier versions of SQL Server backups were created using the SQL Server TAPE format and the operating system used the OS TAPE format. Microsoft changed the format to MTF so that the SQL Server backups would be compatible with the OS backups. Now the SQL Server backup and the OS backup can exist on the same tape.

SQL Server backup automatically spans multiple tapes and it supports hardware compression.

The Backup Process

In SQL Server you can create and backup to a ***dump device.*** A dump device provides a logical association to a path (local or UNC mapped) where the backup will reside.

You can specify the dump device with a friendly name in the GUI with the following syntax:
EXEC **sp_adddumpdevice** 'disk', prodbak, 'c:\production.bak'.

Where 'prodbak' is the friendly name you are assigning to the dump device and production.bak is the file name of the backup file.

Then you can use the friendly name when you backup the database. It looks like this in T-SQL:
BACKUP DATABASE production TO prodbak

Where 'production' is the database name and 'prodbak' is the dump device friendly name.

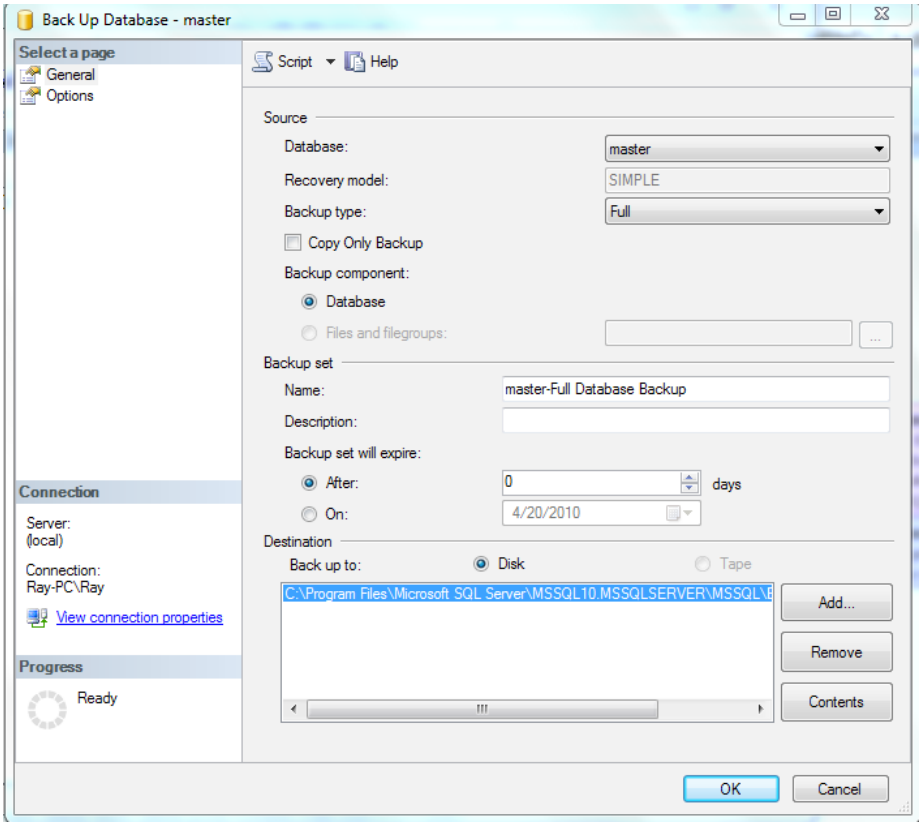


Figure 23: Backing Up the Database

In SSMS right click the database you want to backup, select Tasks and then select Backup from the Tasks menu. As shown above, you can select the database, backup type, name, etc. Once you are satisfied with your selections, click OK and the database will be backed up.

You can select restore from the Tasks menu as well and you will see the screen shown above.

Mirror to

When you specify **MIRROR TO** as an option, SQL Server creates an additional copy of the database during the backup. The backup can be local or remote. You may want to create two mirror backups, one local and one remote to a fileshare in a different physical location.

MIRROR TO works with tapes or disks with the restriction that you have to use the same device type with all of the mirrors. If you don't use the same device type you will get an error 3212.

MIRROR TO is not supported in **SSMS** (SQL Server Management Studio), you have to use a script to do it and it requires the **FORMAT** option as well.

Backup the database 'test' with a mirror:

```
BACKUP DATABASE test To DISK = 'c:\FullBackup.bak'
        MIRROR TO Disk = 'c:\FullBackupCopy.bak'
        WITH FORMAT
```

Designing a Recovery Strategy

Automatic Recovery

To understand automatic recovery we must first take a look and how transactions are stored in memory and on the database:

1. A transaction begins.
2. Changes are immediately logged into memory and the transaction log.
3. Once the record has been inserted or deleted, the transaction is committed (Completed).

When a failure occurs, like a power outage, automatic recovery will attempt to update the database with all changes in the transaction log.

When the system is brought back up -- automatic recovery will update the database with transactions that are found in the log that were committed, but not yet written to the database. Transactions that were started but not yet committed, will be rolled back.

Problem: Over time the transaction log can become quite large and automatic recovery can be a time consuming process.

Solution: Use **checkpoints** to write all of the changed pages to the database disk periodically to reduce the automatic recovery time if the system goes down.

Although configurable, checkpoint runs every minute by default and writes all of the changes in the transaction logs to each database. With checkpoint, the recovery time will be reduced to the time it takes to update from the last checkpoint.

Note: If there are pages occupying memory and SQL Server needs that memory, the pages will be written to disk and the memory will be freed for use. This process is called ***lazy writer*** and it writes pages to disk when the system is not busy doing other things.

Restore Types

- Full
- Differential
- Log
- Piecemeal
- Page
- With Stop AT
- Transaction log
- Restore to log mark

Restore Process

Before SQL Server restore performs a database restore, it does a safety. This safety check is to insure that data is not lost in the restore process by writing over existing data. If the database already exists, and if the name in the restore statement is different than the database name in the backup file, a restore will not be done.

Restore will also fail if some of the files in the backup database are missing or incomplete.
If restore passes the safety check, then it will restore all of the files associated with the database and write the backup data into those files.

- **RESTORE HEADERONLY** – shows the header information.
- **RESTORE FILELISTONLY** – shows file information.
- **RESTORE LABELONLY** – shows media information.
- **RESTORE VERIFYONLY** – verifies that the files are complete and readable.

You can restore a database from SSMS by right-clicking the database you want to restore, then selecting **Tasks** → **Restore** → **Database**.

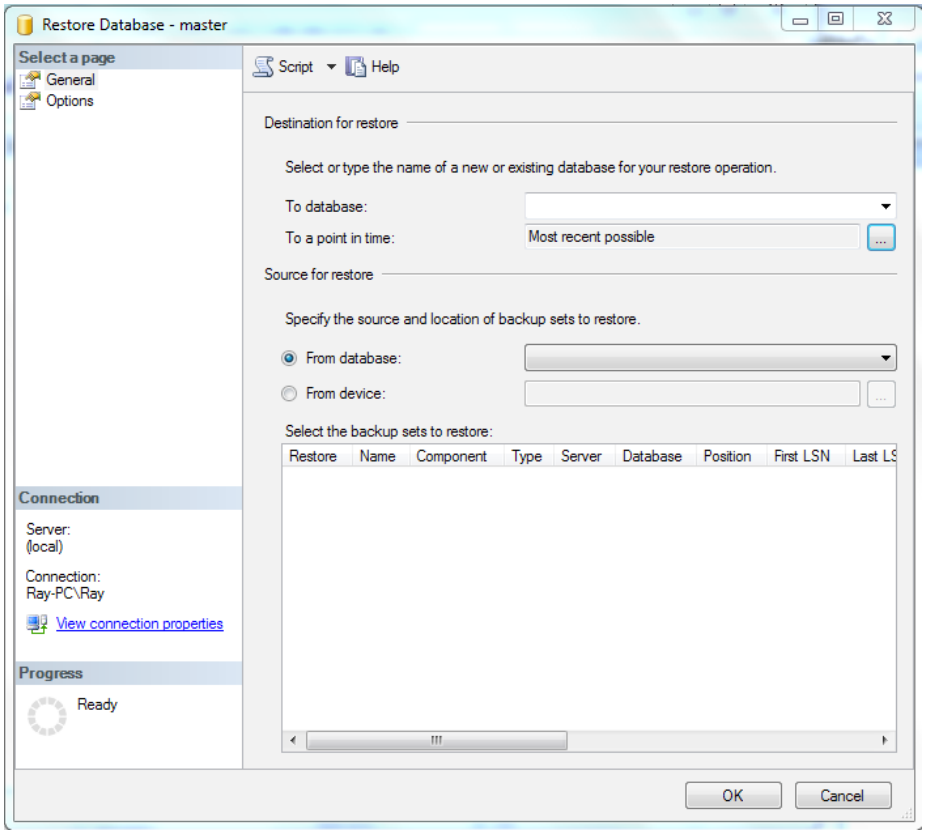


Figure 24: Restoring the Database

From here, you can select the restore from and to databases, as well as **To a Point in Time**. Click **OK** to perform the restore.

Restore Command

- Syntax to restore from disk:
  RESTORE DATABASE mydb FROM DISK = 'c:\mydatabase.bak' WITH NORECOVERY
- Syntax to restore from tape:
  RESTORE DATABASE mydb FROM TAPE '\\.\tape0' WITH NORECOVERY

Restore Command Options

- **RECOVERY/NORECOVERY** – RECOVERY will run unless NORECOVERY is specified.
- **File =** – use this to specify multiple backups on the same disk or tape.
- **Move 'myfile' To 'destination path'** – use this option to restore to a different location from the original database location.
- **Replace** – overrides any safety checks.
- **Stop on error**, **continue on error**
- **STOPAT**
- **Restricted User** – Leaves the database unavailable.
- **CDC** (Change Data Capture) and Service Broker options

Transaction Log Backup and Restore

The Problem: When the transaction log fills up, the database essentially becomes a read-only database because inserts, updates and deletes are not allowed at this point.

The Solution: When the transaction log is backed up, it is also truncated (cleared out) and the space that was used is free for more logging.

You can use the simple recovery mode if you don't need up to the minute recovery, and as we discussed earlier, the transaction log will be truncated periodically.

However, if you are in the full or bulk-logged mode, the transaction log is not automatically truncated and will grow. Transaction log backup backs up all of the changes entered in the log since the last backup. So the solution to the ever growing transaction log is to back it up, so that the log is truncated and space is made available for future transactions.

Before you can do a transaction log backup, you have to have at least one full database backup and you must be in the full or bulk-logged recovery mode.

Here is the syntax for to use to backup the transaction log:

BACKUP LOG database TO DISK 'c:\databaselog.bak'

Bulk Log Recovery Model and Transaction Log Backup

When you are in the bulk-logged recovery mode, every extent that is changed during bulk operations is flagged. When the transaction log backup occurs, after the transaction log is backed up, it checks the flags and backs up all of the flagged extents. This is how the changes made during the bulk-logged recovery mode are protected.

Backup Log Options

- **NORECOVERY** – use this option to backup the tail of the log - it leaves the database in the restoring state.
- **COPY_ONLY** and – this option instructs BACKUP to continue the backup even if errors are encountered like bad checksum or torn pages. It doesn't truncate the log and is equivalent to the NO_TRUNCATE option.

How Often Should You Backup the Transaction Log?

There are two factors that together determine the backup frequency of the transaction log:

1. How often inserts, deletes and updates are done in the database.
2. The size of the transaction log.

You need to decide on the frequency of your backups based on the largest amount of transactions you will have during the interval between backups. If you are going to backup every hour, how many transactions, max, will you have in one hour?

One other consideration is this: what is the largest single transaction you will have? A large transaction can keep your log from truncating while it's working.

Again, with transaction log size, bigger is better. Consider the risk involved with a transaction log that is too small.

How to Restore a Transaction Log

Here are the steps to take, after a crash, to restore the transaction log:

1. Backup the tail of the transaction log. This will retrieve all of the transactions that were committed before the crash, but not written.
2. Restore the database with **NORECOVERY.**
3. Then restore each log with **NORECOVERY** in chronological order from the last database backup to the crash.
4. Restore the tail log that was backed up in step one.
5. Run recovery.

Point-in-Time Recovery

Point-in-Time recovery allows you to restore the log up to a specified time. A new feature in SQL Server 2008 allows you to use the **STOPAT** clause in every sequential log restore from the last database backup up to the point of failure.

Here's how it works:

When you have a failure, you will have the last database backup plus any log backups from the time of the database backup until the failure. You can recover all of the data up to the time of the failure by restoring the database first, then sequentially restoring the log backups using the syntax below for each log backup and up to the time the failure occurred.

1. **RESTORE DATABASE database_name FROM full_backup WITH NORECOVERY**;
2. **RESTORE DATABASE database_name FROM full_differential_backup WITH NORECOVERY**;
3. **RESTORE LOG database_name FROM log_backup WITH STOPAT = time, RECOVERY**;
4. Repeat step three for each transaction log up to the time of failure.

New to SQL Server 2008 is the option to use **STOPAT** with each consecutive transaction log backup—and SQL Server 2008 will not do the **RECOVERY** until it reaches the log backup that occurred on or just before the time of failure.

This simplifies the recovery process, because, before SQL Server 2008 you had to be sure you had recovered all of the data and then you could run RECOVER.

Log Marks

You can also restore with log marks. To do this you can leave log marks with the transactions by:

- **BEGIN TRAN test WITH MARK "description"**. Then when you restore the log you can set the stop point with:

  RESTORE LOG WITH STOPMARK = "test" - This includes the transaction "test" in the restore.

- Or you can use:

  RESTORE LOG WITH STOPBEFOREMARK = "test" - This will NOT include the marked transaction "test".

Differential Backups and Restores

Although differential backup is completely optional, the main advantage of doing differential backups is you can save time on backup and restore operations.

The difference in differential backup and transaction log backup is that transaction log backup stores every change that occurs on each record; differential backups store only the last change on each record.

In other words, if one record is changed more than one time between full backups, the transaction log will have recorded each change that occurred. When a differential backup is used, it stores only the last change that was made to each record. It is faster because it doesn't bother will all of the transactions that led up to the final data in the record.

You can use differential backup with full backup and transaction log backup to speed up the backup and restore process.

If you, for instance, scheduled a full backup for every Saturday and each day between you scheduled a log backup, you could combine the two to recover most or all of the data if there was a failure.

But if you included a Tuesday and a Thursday differential backup and you had a failure on Wednesday, you could restore the Sunday database backup, and then restore the final changes only through Tuesday from the differential backup.

You could be up to date through Tuesday without restoring the log backups from Sunday through Tuesday. You would still have the last change made for every record that had been changed -- but you wouldn't have to restore every transaction that led up to the final change as you would if you were restoring from the transaction log backup.

Note: If the differential backup becomes unavailable for some reason, you still have the log backups to recover from.

How to Backup and Restore Files and Filegroups

With SQL Server you can backup a file or a filegroup by itself and it can be done with both the full backup and differential.

The syntax to do a file backup is:

BACKUP DATABASE database_name FILE = 'myfile'

The syntax for a filegroup is:

BACKUP DATABASE database_name Filegroup = 'myfilegroup'

You can also restore an individual file or filegroup in SQL Server. If you are in the simple recovery mode you can only restore read-only files and filegroups, but in the full recovery mode, you can restore and file or filegroup.

Here are the steps to restore a file or a filegroup:

1. Mark the file offline.
2. Do a tail log backup to capture changes since the last full backup.
3. Restore the database files.
4. Restore the differential.
5. Restore the transaction logs in chronological order from the last differential backup or the last full backup.

Note: With the Enterprise edition, you can restore the file or filegroup while the other files/filegroups are online.

Online Restore

Online restore improves uptime of tables during the restore process in bulk logged and full recovery modes. The improvement is due to the database staying online and available during the restore and only the parts being restored staying offline until the restore is complete.

If you use SQL Server Enterprise edition 2005 or later you can do online restores of files, pages, and piecemeal restores. Piecemeal allows the user to restore parts of the database while other parts go online.

You can do online restores of databases that have multiple files and filegroups or simple recovery databases with read-only files and filegroups.

A database is considered online as long as the primary filegroup is online. During an online restore the file that is being restored and its filegroup are offline while the rest of the database remains online and available.

Note: The syntax for online restore is the same as for a normal restore.

How to Fix Orphaned Users

The Problem: If you restore a database to a different server with a different master, the users that are tied to logins in the original master will not be able to access the new database—they become orphaned. The exception to this would be if the users also had a login tied to the new master.

The Solution: After restoring a database to a different server, you have to reconnect all of the users previously connected to the database.

For Active Directory users the fix is simple, you just re-add the logins to the new master. Since the **SID** (Security ID) comes from Active Directory, it will be the same on both servers. So it will map correctly back to the user.

On SQL Server however, the SID will be different when you re-create the login on the new server so it will not link to the user record in the database. To fix this problem, you will use **sp_change_users_login**. Here are the parameters for **sp_change_users_login**:

- Report - Shows a list of orphaned users (users who's SIDs do not match correctly).
- Auto-fix - Looks at the user names and matches them with the newly created login user names and reconnects the users. It will create the new login if one doesn't exist.
- Update-one - Works the same as auto-fix except that you can specify a particular user.

Restoring Pages

You can restore a page or a group of pages. A page can be marked suspect because of errors found with DBCC -- they may be torn pages or have checksum errors.

You can check suspect pages with **database_suspect_data_page_event**.

In order to repair the page you need to know if it is a clustered or nonclustered index. You find this by looking at the **INDID** (Index ID).

- If the INDID = 0, then it's a heap with no cluster.
- If the INDID = 1 this indicates a clustered index on the table.
- If the INDID > 2 and < 255 it's a nonclustered index.

If the page marked suspect is a non-clustered index, simply drop it and re-create it. For a page that is in the table or the clustered index, perform a page restore. The syntax for a page restore is:

RESTORE DATABASE database_name PAGE

plus the list of pages that you wish to restore.

Designing a Recovery Test Plan

Log Shipping

For log shipping you should plan a failover and you should fail over all the way to the point where you are doing a client connect.

Replication

- For replication recovery testing, you should fail the publisher, the distributer and the subscribers - in all possible combinations.
- Allow some subscriptions to expire and plan recovery accordingly.
- Plan to back up **msdb** and master after you make any topology changes.
- You can use sync with backup as an option to coordinate the backup and recovery process and make sure everything stays in sync.

When you use sync with backup as an option, you can use either sync on distributer or sync on publisher. If you choose **sync on Distributor**, the publisher log is not truncated until it is backed up in the distributer - this option puts the distributer in sync with the publisher. This way the distributer can be restored from the last backup and replication can continue.

If you choose the option **sync on publisher** with this option transactions will not be delivered to the distributer until they are backed up on the publisher - this slows down replication. But in this case, the last publisher backup can be restored and both the distributer and publisher will have the same transactions recorded.

Hardware Considerations

When considering the hardware for testing make sure the hardware you use is exactly the same as the disaster recovery hardware -- or it has to have the same capability for files/filegroups, disk size, network speed and **NIC** (Network Interface Card) configuration.

Hardware Cluster Failure

For hardware cluster failure you could have a disk failure or an OS failure. In the case of a hardware failure, you will want to remove the failed node from the cluster with SQL Server setup. Then remove the failed hardware, fix or replace the hardware, bring the machine back up and use SQL Server setup to install windows clustering and add back the failed node.

For an OS cluster failure, if the node is offline - try to recover the node first and test failover. Otherwise take the same actions that you would take if it were a hardware failure.

Instance Rebuild

For an instance rebuild you have to restore the OS as well as SQL Server. Here is a list of what you must do to rebuild the instance:

- Restore the server configuration.
- Install the SQL Server Software.
- Restore all of the system and user databases.
- Change the server name in master if you need to.
- If needed, redo the following:
    - Replication
    - Mirroring
    - Log Shipping

Your Test Plan

In your test plan you should include all of the following:

- Hardware and software information
- Contacts for key people
- Connection information
- Client testing

Make sure you document the software and version including and service packs you will need to restore - and you know where your recovery software is located. Keep the document updated as you upgrade your software or add service packs.

Keep up with your backups as well and make sure they are always readily available for restore operations.

Domain 5: Designing a Monitoring Strategy

In this section we will discuss the various tools available for monitoring activity in SQL Server to help plan design optimized server instance performance. You will learn how to monitor server performance at the operating system level and at the instance level. How to use tools like the System Monitor counters, Windows Management Instrumentation, event notifications, tracing and analysis tools.

List of Monitoring Tools

- System Monitor
- SQL Server Profiler
- WMI event notifications
- Data collection
- Extended Events
- DMVs and T-SQL
- Activity Monitor
- Policy Based Management
- SQL Server Agent Alerts
- Event logs

System Monitor

With System Monitor you can view data from multiple computers and export the data. You can create system alerts and run applications when those alerts occur. You can also save logs and view data from the logs.

You need to be familiar with the counters and which ones are useful.

Using the System Monitor

To start the system monitor you simply enter 'perfmon' from run prompt in Windows desktop or from the server OS. Here you can monitor the activity of things like:

- CPU - % of usage and maximum frequency
- Disk - KB/sec and % of Highest Active Time
- Network - %kbps and % of Network Utilization
- Memory - Hard Faults/sec and % of Physical memory Used

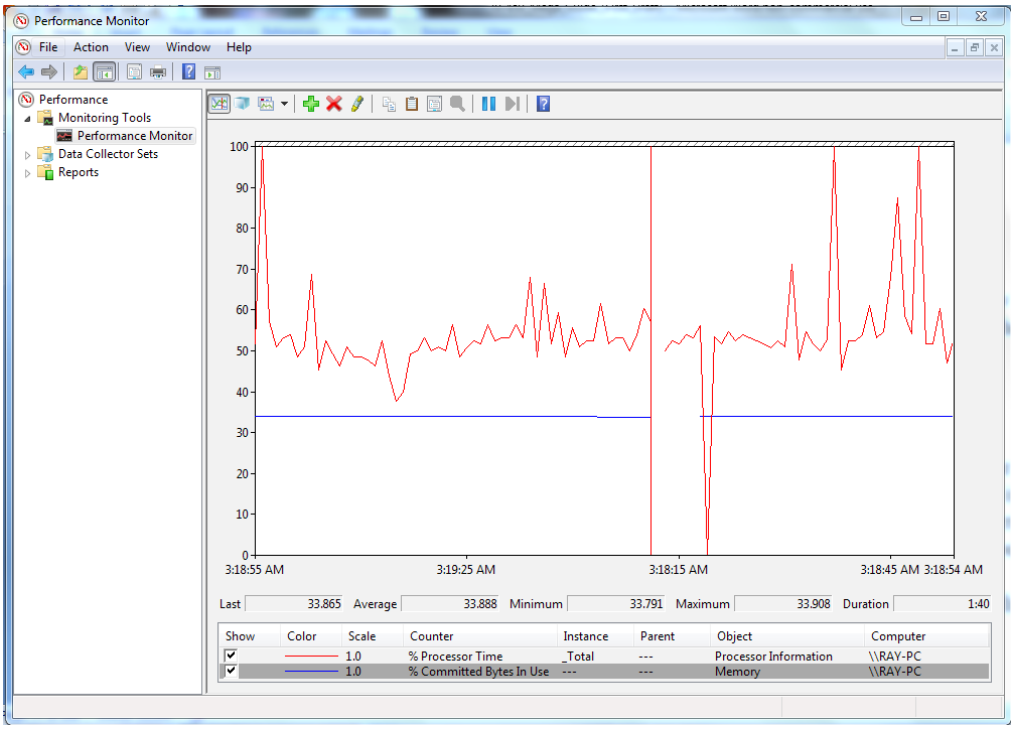


Figure 25: Performance Monitor

You can open the section associated with each of these and see more details on how the resource is divided. Here you can add counters by clicking the green plus sign at the top of the main window and selecting what you want to monitor from a multitude of available counters.

Data Collector Sets

A Data Collector Set is used for monitoring and reporting in Windows Performance Monitor. You can create a Data Collector Set and group it with other Data Collector Set and incorporate into logs. You can view it in Performance Monitor and you can configure it to generate alerts at certain thresholds. You can associate it with rules of scheduling to collect data at specified times.

You can also configure Windows Management Interface so that WMI tasks run at the completion of a data set collection.

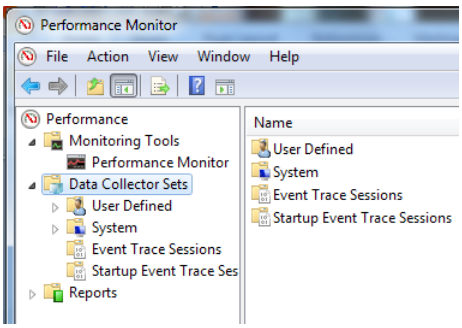


Figure 26: Configuring Data Collector Sets

To define your own data collection, expand the Data Collector Sets tab, then expand the User Defined tab and right click on New Data Collector Set and select Properties.

In the New Data Collectors Set Properties window you can:

- Add your user defined set.
- Select where to save the data collection.
- Set security options.
- Set a schedule to determine collection start and stop times.
- Have it stop if certain conditions occur, like the size of the file reaches some number.

Data Collector Sets can contain:

- Performance Counters
- Event trace data
- System configuration information

SQL Server Activity Monitor

In SQL Server Management Studio there is an Activity Monitor that is similar to the System Monitor - with the exception that the counters available are relative to SQL Server activity. To access this monitor simply right click the server tab on the left of the screen and select the Activity Monitor option.

Here you can monitor the activity associated with the database such as % Processor Time, Database I/O (MB/sec), Batch Requests/sec and Waiting Tasks.

Monitoring Memory

SQL Server automatically allocates and releases memory for data cache based on your settings for min and max memory and the availability of resources. SQL Server tries to keep memory available at all times without paging. If there is not enough memory, memory will be paged to and from the disk and performance will suffer.

You can memory using the System Monitor and the table below contains some guidelines to follow for optimum system performance.

	Counter	**Guideline**
Memory and Paging File Use	Memory: Available Bytes	Should consistently be >5000KB (5MB)
	Memory: Pages per sec	Should never be consistently >0
	Process: Page Faults/sec/SQL	Should never be consistently >0
Memory and Buffer Use	Process: Working Set/SQL	Should be >5000KB (5MB)
	SQL: Buffer Management: Cache Hit Ratio	Should be >90%
	SQL: Buffer Management: Total Pages	Should not be low
	SQL: Memory Management: Total Server Memory	Should not be close to physical memory

Figure 27: Guidelines for Monitoring Memory

- **Available Bytes** – should be at least 5000KB (5MB). Low values here indicate an overall shortage of physical memory.
- **Pages per Second** – monitors the number of pages that are written to and read from the hard disk by the OS to resolve hard page faults. Any value consistently above zero would indicate that the OS is reading and writing the disk in response to memory requests.
- **Page Faults/sec** – same as Pages per Second with the exception that Page Faults indicate soft paging rather than disk reads and writes.
- **Working Set/SQL** – how much memory SQL Server needs to use.
- **Cache Hit Ratio** – indicates the percentage of time that data is in cache when needed. Should always be a high number.
- **Total Server Memory** – total memory SQL Server is using. Should be near the total memory of the system.

Monitoring Processors

For multiprocessing systems you should monitor each processor separately. Here are some guidelines for proper processor performance:

Counter	Guideline
Processor: % Processor Time	<90%
System: Context Switches/sec	<8000 on a multiprocessor
System: Processor Queue Length	Should not be >2
Processor: % Privileged Time	Low as possible

Figure 28: Guidelines for Processor Performance

- **Processor Time** – as a general rule, should stay below 90% most of the time. Although it may peak at 100% occasionally, if it stays there you need a faster processor.
- **Context Switches/sec** – should be less than 8000 on a multiprocessor system.
- **Processor Queue Length** – should not be greater than two.
- **Percent Privileged Time** – should be a low number.

Monitoring Hard Disk I/O

Disk I/O is one on the primary causes of bottlenecks in a system. It's important to understand that excessive disk activity can be caused by insufficient memory. With insufficient memory the system uses the disk to store page files that would normally be cached in memory. So you want to make sure you eliminate insufficient memory as the possible indirect problem before you suspect an I/O problem.

Sequential I/O, as in the case of log files, can be much faster than random I/O, such as database reads and writes. This is another reason, besides increased recoverability, to separate the database files from the log files.

You tune your applications to reduce disk I/O - better indexes and better database normalization can help. You can also use faster hard drives with more spindles to speed things up. Moving some files to a different hard disk by using filegroups -- as we discussed earlier -- will help.

You can spread the files across multiple spindles using RAID also, this will increase performance. Here are some guidelines for best disk performance:

Counter	Guideline
PhysicalDisk % Disk Time	<90%
PhysicalDisk Average Disk Queue Length	<2 x Number of Spindles
PhysicalDisk Reads/sec	<Capacity
PhysicalDisk Writes/sec	<Capacity

Figure 29: Guidelines for Physical Disk Performance

- **Percent Disk Time** – should be less than 90% most of the time.
- **Average Disk Queue Length** – should be less than two times the number of spindles.

Baseline Monitoring

A good time to gather some baseline statistics is when your server is running well and is heavily loaded -- you will also want to collect statistics when it's not running as well so comparisons can be made. A good record of the resource usage at varying levels of performance can help you identify problem areas.

You want to collect information from the counters listed below:

- Database: Transactions/sec
- General Statics: User Connections
- Locks: Average Wait Times (ms)
- Locks: Number of Deadlocks/sec
- Processor, Memory and Disk performance counters discussed earlier

Once you benchmark this data when the server is running well, then you can compare the data to a time when the system is not running as well and identify problems.

Log Viewer

When troubleshooting SQL Server issues, the ability to view system and application log data is necessary at times. SQL Server logs are available to monitor events that occur within SQL Server. You can access the log viewer in SQL Server Management Studio by expanding the Management tab and then the SQL Server Logs tab. Here you can right click on any log and select View SQL Server Log.

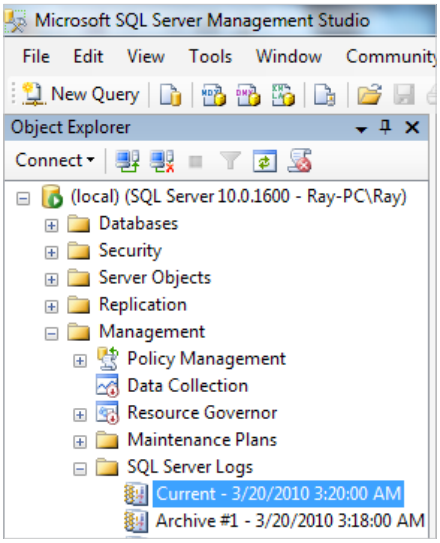


Figure 30: Viewing the SQL Server Log

Once you open the Log File Viewer you have access to the Database Mail log, the SQL Server log, the SQL Server Agent log and the System log.

One nice feature of the Log File Viewer is that you can combine log data from the different logs available and view them together for comparison by simply selecting them.

Windows Management Instrumentation

Windows Management Instrumentation (WMI) can be used to for things like checking performance counters, starting and stopping services and managing configuration objects. **WQL** (WMI Query Language) is a scripting language that is very similar to T-SQL and can be used to help automate the monitoring process and gives you control of WMI.

SQL Server Profiler

With SQL Server Profiler you can capture events that are of interest. You and see exactly what T-SQL statements or Multi-Dimensional Expressions are submitted to the server and SQL Server resolves queries internally.

With SQL Server Profiler you can:

- Create a trace.
- Monitor the trace results while the trace is running.
- Store the results in a table.
- Stop, start, pause and modify the results.
- Replay the results.

SQL Server Profiler can be helpful with performance tuning, debugging, auditing and for stress testing. You can correlate the System Monitor with the SQL Server Profiler and you can use the information captured with SQL Server Profiler as source information for Tuning Advisor.

Note: Be careful not to monitor too many events, this can affect performance and the trace file can grow very large if you are monitoring for a long period of time.

Database Engine Tuning Advisor

Database Tuning Advisor provides help in tuning databases for better query performance. It's an improved version of the Index Tuning Wizard from previous versions of SQL Server. By analyzing database performance it helps provide better indexes, index views and partitions.

Database Tuning Advisor reads the physical database. It uses workload information like profiler trace and SQL Server scripts to provide recommendations for better database performance.

Note: Although there are options to allow Database Tuning Advisor to change things while it is running, it's recommended that you DO NOT allow changes on the fly. It may delete or change data that you don't want changed, so look at the recommendations and then decide what should and should not be changed. For example: If you run Database Tuning Advisor for a period of time and certain indexes were not used during that time, it may delete those indexes!

Database Tuning Advisor is available in the SQL Server Management Studio in the Tools tab. Just open the Tools menu and select Database Engine Tuning Advisor to get started.

In the Database Tuning Advisor window, you can select the session name, where you want the recommendations stored, which database(s) to tune, stop time and tuning options like the partitioning strategy you would like to use.

Under the Tuning Options tab is where you select whether you want Database Tuning Advisor to keep the physical design structure or make changes. Remember, allowing the advisor to make changes can be dangerous, so be careful here.

You can then start the advisor and let it run for the specified time.

Once the analysis is finished, you will see the Recommendations tab; this is where you can see the estimated improvement percentage and the recommended changes to reach that performance level.

You can run Database Engine Tuning Advisor on a different server to offload the work required by the advisor. The second server does not have to be identical to the server containing the database(s) you want to tune. When you run the Database Engine Tuning Advisor this way the bulk of the work is done on the second server leaving your production server relatively free to continue its normal tasks.

Performance Monitoring with DMVs

Dynamic Management Views (**DMV**s) provide some tools for viewing important information to monitor. Here are the queries:

- **SELECT * FROM sys.dm_db_missing_index_details** – provides details of missing indexes that could have been used if they had been available.
- **SELECT * FROM sys.dm_db_missing_index_groups** – details of missing index groups.
- **SELECT * FROM sys.dm_db_missing_index_columns(12)** - Details of missing index columns.

This query will provide a list of the statements with highest avg CPU Time.

```
SELECT TOP 50
    qs.total_worker_time/qs.execution_count as [Avg CPU Time],
    SUBSTRING(qt.text,qs.statement_start_offset/2,
                        (case when qs.statement_end_offset = -1
                        then len(convert(nvarchar(max), qt.text)) * 2
                        else qs.statement_end_offset end -qs.statement_start_offset)/2)
                as query_text,
                qt.dbid, dbname=db_name(qt.dbid),
                qt.objectid
FROM sys.dm_exec_query_stats qs
cross apply sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY
    [Avg CPU Time] DESC
```

The following query will provide a list of statements with the highest execution counts.

```
SELECT TOP 50
    qs.execution_count,
    SUBSTRING(qt.text,qs.statement_start_offset/2,
                        (case when qs.statement_end_offset = -1
                        then len(convert(nvarchar(max), qt.text)) * 2
                        else qs.statement_end_offset end -qs.statement_start_offset)/2)
                as query_text,
                qt.dbid, dbname=db_name(qt.dbid),
                qt.objectid
FROM sys.dm_exec_query_stats qs
cross apply sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY
    qs.execution_count DESC
```

Management Data Warehouse

Management Data Warehouse (MDW) provides a central point for data collection for **ALL** SQL Servers. You can set it up to gather data for a single instance or multiple SQL Server instances. The data collectors can be scheduled and you can set retention periods so that the data is automatically deleted after a period of time. The data collected is stored in the MDW database.

To set up MDW, from the Object Explorer in SSMS expand Management and right click on Data Collection. Select Configure Data Management Warehouse to start the Configure Data Management Warehouse Wizard. Once the wizard is started, you will see the following screen:

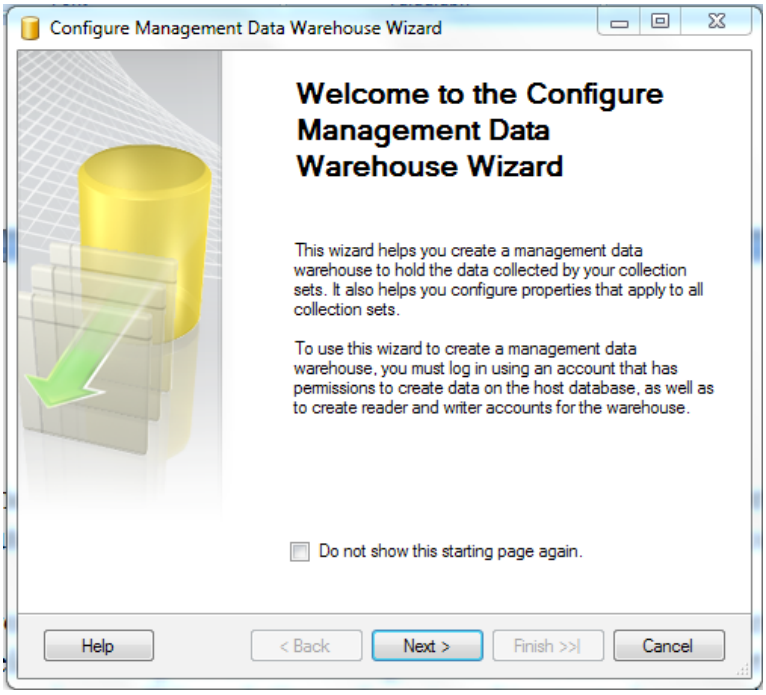


Figure 31: Starting the Data Warehouse Wizard

Click Next.

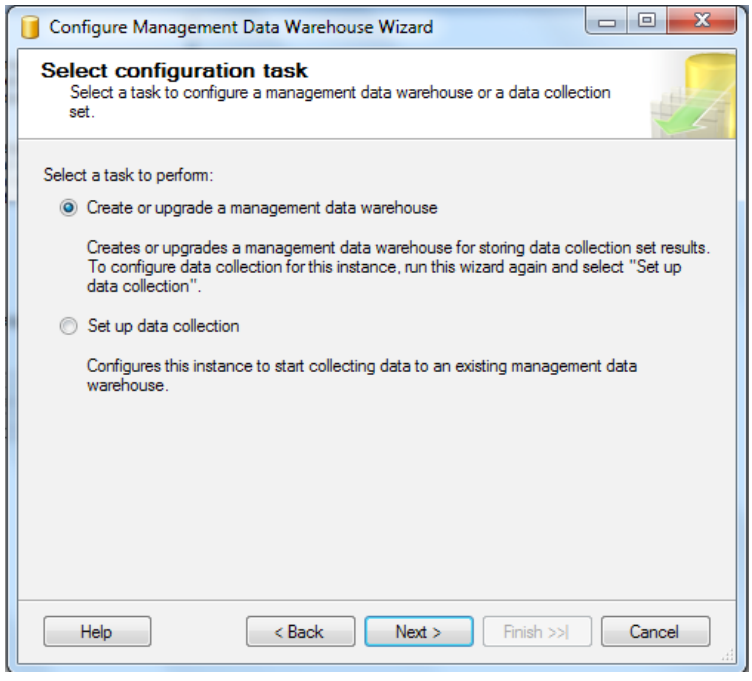


Figure 32: Creating a Management Data Warehouse

Note: You can also start MDW from the Start menu by expanding the Microsoft SQL Server 2008 folder and then expanding the Performance Tools folder and selecting the Database Engine Tuning Advisor. From here you can create the MDW database (Create or upgrade a management data warehouse) or you can configure to start collecting data (Set up data collection).

You will need to run the wizard twice to set up a MDW, once to create the MDW database and the second time to set up the data collection and start collecting the data. With MDW you can collect data from:

- SQL Server Trace
- Performance Monitor counters
- T-SQL
- Query activity

The collection process is accomplished using SQL Agent jobs and stored procedures that work with Management Data Warehouse. The SQL Agent jobs can be scheduled. The data from the collection process can be cached locally and written to the MDW database when the buffer is full or on a schedule if you choose.

As another option, you can have the data immediately uploaded to the MDW as soon as it is captured.

Management Data Warehouse Security

There are two kinds of security for MDW. The Data Collector security determines who is able to control the data collection process. The Access Security determines who has access to the performance data. There are three roles involved in MDW security:

- **dc_admin** – this role is the collector administrator for a specific instance. The person operating in this role can add new collection sets and install new collection types.
- **dc_operator** – can list, stop and start a collection set and change the scheduling of the upload and collection.
- **dc_proxy** – a person operating in this role has read-only access to the configuration.
- There are also three MDW roles for the MDW database:
  - **mdw_admin** – in this role a person can read, write and update schemas
  - **mdw_writer** – this person can read and write data only
  - **mdw_reader** – this is a read-only access role and has read-only access to data in the data warehouse.

You can set up MDW security on your primary server and here are the steps:

1. Create the necessary logins to map to the roles listed above.
2. Configure MDW.
3. Create proxies for SQL Server Agent steps.

Dedicated Administrator Connections

Problem: When your server uses all of available memory, the activity on the server can keep you locked out. You may or may not know what's wrong but you have no way to find out because the server is not responding and you can't make a connection in order to debug it.

Solution: Reserve some memory that is exclusively used for the administrator login so that the administrator can gain access and debug the system.

Dedicated Administrator Connections reserve memory exclusively for the administrator to have access to the system when they would normally be locked out.

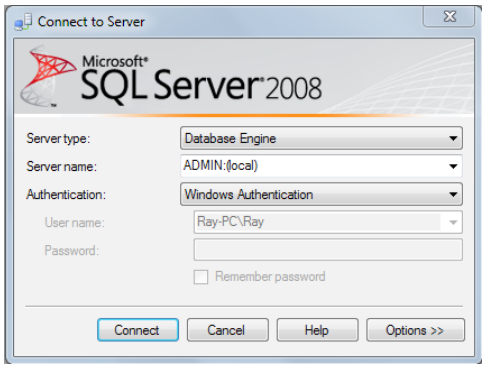


Figure 33: Creating a DAC

This is a simple connection to make. When you connect to your server, you simply include ADMIN: at the beginning of the server name (ADMIN:server_name) as shown above. Once you are connected, then you have a dedicated administrator connection.

Now you are back in and can troubleshoot the SQL Server issues that caused you to be locked out.

Locking

In SQL Server, locks provide a way to protect data from being updated by more than one user at one time. You can lock access to data at many levels in SQL Server:

Level	Lock
Row	Row
Key	Index row lock
Page	Data or index page
Extent	Eight contiguous pages
HOBT	Heap or B-tree
Table	Entire table
Database	Entire database

Figure 34: Locking Levels

Exclusive Locks

Exclusive locks protect data during a transaction from being accessed by another user. For instance, if one user were updating the price on a single row in a table and another user attempted to access the data, an exclusive lock would make the second user wait until the update completes before being allowed make a change in the price.

Shared Locks

Shared locks allow many users to read the same data at one time but prevent anybody from changing the data until all the data read operations are complete. So if many users are reading the price of an item at the same time, shared locks will be applied for each read operation, others will be allowed to read, but nobody will be allowed to update the price until all of the read operations complete.

If someone tries to change the row when there are shared locks, they will be blocked and will have to wait until all of the shared locks are released.

Update Locks

Problem: If a user attempts to update multiple records in a category with conditional updates, exclusive locks on all objects in that category would lock those objects until the writes were complete.

Solution: Allow reads on all of the objects in the category until it is determined which ones need to be updated, then exclusive lock only those objects while the updates are being made.

Update locks work in situations where records must be read and qualified before written to. Let's say that you want to increase the price of all products in the men's clothing department over $100 - by two percent. With Update Locks a shared lock will be placed on each product in the category while the SQL Server is determining which products need the prices changed. Once decided, an exclusive lock will be applied while the update is taking place.

Intent Locks

Problem: If a row is being modified in a particular table, in a particular database, someone could come along and lock the table, locking out the person modifying the row data.

Solution: Send a flag up through the hierarchy to notify everyone above that changes are happening below.

Intent Locks are these flags that are placed above the work in the hierarchy to notify others that changes are being made below.

Here are the three types of Intent Locks that are sent up through the database hierarchy:

- Intent Shared Lock is the flag that indicates that there is a shared lock below.
- Intent Exclusive Lock indicates that someone has an exclusive lock below.
- Shared Intent Exclusive Lock indicates that someone has an update lock below.

Monitoring Locks

To monitor locks you can use the dynamic management view **sys.dm_tran_locks** -- where locking data is stored. You can also use the stored procedures **sp_who** and **sp_locks**.

You can also use SQL Server Management Studio -> Current activity to monitor locks as well as some of the other monitors we've discussed in this session - like profiler.

Deadlocks
A Deadlock occurs when two transactions have locked two objects independently of one another, but then they both cross over and attempt to access the object that the other has locked. They end up waiting on each other to release the locks indefinitely.

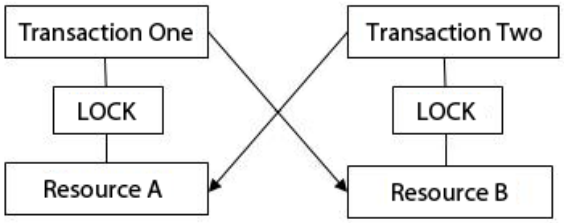


Figure 35: Understanding Deadlocks

For instance, if Transaction One locks Resource A and Transaction Two locks Resource B, then Transaction One attempts to access Resource B and Transaction Two attempts to access Resource A - a deadlock occurs.

Once the deadlock occurs, neither transaction can complete because each transaction is busy waiting on the other transaction to release the lock. SQL Server handles this by generating a deadlock error 1205. When this error is generated one of the transactions is identified as a victim and the transaction is rolled back allowing the other transaction to complete.

To monitor deadlocks you can set the trace flags 1204 and 1222 to see additional information about deadlocks in SQL Server error log. With Profiler you can capture a deadlock graph event and you have more ability see what's going on and to manage the deadlocks. Also Windows Monitor has a counter that keeps track of deadlocks per second.

Domain 6: Designing a Strategy to Maintain and Manage Databases

In this section you will learn database management as well as database maintenance. You will learn maintenance strategies using data compression techniques, index and heap maintenance and partition management -- how to govern limited resources for better performance in your design strategy -- and how to automate database management using tools like PowerShell, Windows Management Instrumentation, SQL Server Agent and event notifications.

Designing a Maintenance Strategy for Database Servers

Data Compression

Data compression is new to SQL Server 2008 and you can compress the following:

- Heaps
- Clustered index tables
- Nonclustered indexes
- Indexed views
- Partitions of a partitioned table

Compression is done with the **CREATE** and **ALTER** commands. Row and page compression are available in SQL Server 2008.

SQL Server will allow compression only if the compression is guaranteed to succeed.

Note: Data compression is only available on the Enterprise and Developer Edition of SQL Server but not on the Standard Edition.

Heap Compression
A table that does not have a clustered index is called a heap. Pages are only compressed in a heap when you do a bulk insert, an insert with tablock or an alter table rebuild. There is no page compression when you use DML statements—so if you do a row insertion on a heap, the data stored is not compressed data. You have to maintain compression on a heap by removing and adding compression back periodically.

When you change the data compression setting on a heap, all of the nonclustered indexes will be rebuilt.

Row Compression
Row compression reduces the metadata overhead associated with a record. Metadata is information about stored data, like information about columns, their lengths and offsets.

For numeric type data (integer, decimal, float etc) and types based on numeric data type (like DateTime and money), compression uses variable-length storage format. Compression stores fixed character strings using by using variable length format and it doesn't store the blank characters.

Page Compression
Page compression consists of row compression, prefix compression and dictionary compression.
When you enable page compression, existing pages are rebuilt.

Heap and Index Management

You need to manage the heap and index because they can become fragmented over time as database reads and writes occur. To check the status of the heap and indexes you can use SQL Server Management Studio and look at the index properties. You can also use the DMV, **sys.dm_db_index_physical_stats** to see the overall health of the indexes.

There are two types of fragmentation that you need to be concerned with, internal and external (logical). Internal fragmentation has to do with the percentage of pages full (or how much wasted space there is because pages are not being filled to near capacity) and external is when the physical page order does not match the logical page order.

Here are some guidelines for when and how to fix fragmentation:

For external fragmentation - if the external fragmentation percent is:

- Less than 5 percent - usually no action is needed.
- Greater than 5 percent and less than 30 percent - reorganize.
- Greater than 30 percent - rebuild.

Fragmentation can slow things down in a system. If there is internal fragmentation and the allocated pages become full, the next insert can result in allocating a new page, splitting the page intended to receive the data and changing the pointers to include that new page in the chain.

Each time this happens more I/O is necessary than would be for a simple insert if there was space available.

The key to repairing fragmentation is to have the pages full enough not to waste lots of space but not so full that new data can't be written without page splitting.

External fragmentation is at its lowest when the linked pages are contiguous on the physical disk space. In other words, when the logical pages match the physical pages the pages form an unbroken physical order. The first logical page is also the first physical page; the second logical page is the second physical page and so on.

The percent of out-of-order pages is the percent of pages where the physical and logical numbers are different.

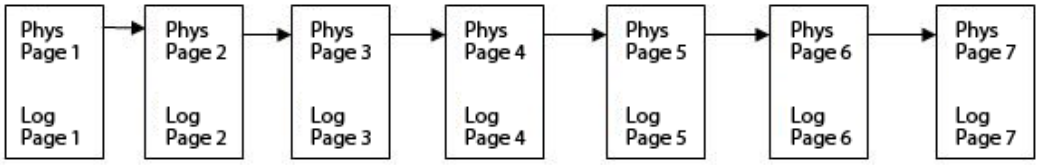


Figure 36: Understanding Fragmentation

In the above example, the percent of out of order pages is zero (fragmentation = 0%), this is the best situation. The links list points to the next physical page for all six pages after the first page.

If a new page had to be allocated for a page in this ordered sequence, SQL Server would find the space and insert the page with pointers. Then the logical order of pages and the physical order of pages will be different.

Reorganize Index

To reorganize the index you can use **ALTER INDEX... REORGANIZE** command. This replaces DBCC index defrag. It repairs the index at the leaf level and fixes the logical/physical order. Reorganizing compacts the index per the fill factor meaning that if the pages are too empty it compact them to fill them up. But if the pages are too full, it WILL NOT expand them for more space.

Reorganizing the index also compacts LOBs. It's always online so work can continue while the reorganize is going on although it can impact performance -- and the quality is not as good as a rebuild. It runs faster than a rebuild but it doesn't update the statistics. Reorganize causes less interference with online applications than a rebuild does.

Rebuild Index

You can rebuild an index with the **ALTER INDEX... REBUILD** command or the **DROP INDEX... WITH DROP** Existing command. This replaces the **DBCC DBREINDEX** command.

Rebuild index can be online or offline and it's more complete than the reorganize. It will enable a disabled index. To rebuild the index it drops and then re-creates the entire index. During the rebuild process, both the logical and physical page ordering is fixed. It does this by expanding and compacting as necessary per the fill factor.

Rebuild index can consume lots of space. To minimize space usage you can do a couple of things. You can disable nonclustered indexes and then rebuild them. You may even consider going minimally logged during the process, just be sure to restore full logging as soon as the rebuild is finished.

Remember that only the Enterprise edition of SQL Server allows online rebuilds. Microsoft recommends that you test the online rebuilds to see how the impact other users in the same environment. Deadlocks may occur because you will be competing for row locking with other users. The total time for a rebuild online will be longer than it would be offline.

The advantage to using online rebuild is that others in the same environment can continue their work while the rebuild is in process.

The online rebuild is not available for:

- XML indexes
- Clustered indexes on tables with LOB data
- Unique nonclustered indexes

Online rebuild does require extra disk space compared to offline operations because it has to keep an old and a new copy of the index.

Row Forwarding Pointers

Problem: In a heap, if someone updates a row—making it larger so that the row no longer fits on the page—it will be moved to a different page. The page address is the physical location of the row. Since the physical location changes when the row is moved to a different page—all of the nonclustered indexes that reference that row would have to be updated to point to the new physical location.

Solution: SQL Server uses row forwarding pointers to point to the new locations when rows are moved. So, rather than having to fix all of the pointers to that row, the forwarding pointer now leads to the new physical location of the row.

The lesser problem now is that with more forwarding pointers, you have more I/O to get to the rows that have been physically relocated.

The answer to this problem is to manage the heap in order to remove the forwarding pointers to make it less expensive to access the relocated rows.

Heap Maintenance

What you want to do for heap maintenance is to get rid of the row forwarding pointers, move the rows back onto pages and fix the nonclustered indexes so they point directly to the row's physical location. For heap maintenance you will create and then drop the clustered index. When you create the clustered index, the row forwarding pointers will be fixed and the pages will be expanded or compressed per the fill factor. When you drop the clustered index the index will be converted back into a heap with the pointers fixed and the physical structure remains intact.

Table Partitions

With table partitions you can split tables either logically or physically into many separate smaller tables. To the end user, the split table still looks like one table. This allows the optimizer to search and return data from one of the the smaller tables (or subsets of data) rather than having to search the entire table (or set). With partitions, you can also have line indexes (indexes that are associated with a subset) so that searches will be faster.

One advantage for the administrator is that maintenance can be done on one the subsets individually and independently of the others. For the end user, the performance will likely be better on a partitioned table than one that is not partitioned - especially on the larger tables.

Table partitions give you the ability to split large tables up and manage them in smaller subsets.

Partition Management
Rolling partitions provide a way to store data in a series of postings, archive the oldest posting and create a new slot for the posting. For instance, if you want to maintain six months of rolling data for some data that is stored by the month, partitions can be created for the most recent six month period.

Let's say that you were keeping a record of monthly sales and you wanted to store the most recent six months in a Posting table and store the old data in a posting History table.

You would have six partitions that divided the Posting table into the most recent six months.

Posting		**Posting History**	
1	Oct 1, 2009	1	<Oct 1, 2009
2	Nov 1, 2009	2	Oct 1, 2009 (Empty)
3	Dec 1, 2009	3	
4	Jan 1, 2010		
5	Feb 1, 2010		
6	Mar 1, 2010		

Figure 37: Understanding Partition Management

In the posting history, all of the postings before October 2008 are stored in partition one. In order to update the tables at the end of a month you will want to roll the oldest month's data to the Posting History table at the end of the month -- then create a new partition for the new month's postings. To implement the rolling partition:

- Split the empty partition (partition two) in the Posting History table.
  ALTER PARTITION PostingHistoryPF() SPLIT RANGE ('11/1/2009')
  This creates partition three in the Posting History table with a date of November 1, 2009.
- Create a check constraint on the Posting table.
  Define check constraint on **Posting Date >= 'Oct 1, 2009'**
- Switch partition one in the Posting table with partition two in the posting History table. This moves the October data into partition two in the Posting History table.
  ALTER TABLE Posting SWITCH PARTITION 1 to PostingHistory PARTITION 2
  Now partition 1 in the Posting table is empty and the data has moved to partition two of the Posting History table.
- Merge partitions one and two in the Posting table to remove the empty first partition.
  ALTER PARTITION FUNCTION PostingPF() MERGE RANGE ('11/1/2009')

- Merge partitions one and two in the Posting History table to add the new data to the history.
  ALTER PARTITION FUNCTION PostingHistoryPF() MERGE RANGE ('10/1/2009')
  Now the history has been updated with October's entries.
- Make a new partition available in the Posting table for the new month.
  ALTER PARTITION FUNCTION PostingPF() SPLIT RANGE ('4/1/2010')

Posting		Posting History	
1	Nov 1, 2009	1	<Nov 1, 2009
2	Dec 1, 2009	2	Nov 1, 2009 (Empty)
3	Jan 1, 2010	3	
4	Feb 1, 2010		
5	Mar 1, 2010		
6	Apr 1, 2010		

Figure 38: Understanding Rolling Partitions

The final state of the tables after this process is ready for the April 2010 postings with the October 2009 postings stored in the Posting History table.

Index Statistics

An index is only useful if it reduces the amount of work that must be done to access the data in the database. Index Statistics are what the optimizer uses to determine if an index is useful or not.

Problem: Index statistics are created at the same time the index is created with data present. But the statistics are not updated for inserts, updates and deletes. Indexes can become obsolete over time so that the optimizer chooses the wrong plan for queries.

Solution: Update the index statistics regularly enough to maintain optimum performance.

Using Auto Update Statistics
At the creation of an index with data, index statistical information is created as well and stored.

This information contains statistics about the data such as how many records have a last name beginning with the letter A. This information helps SQL Server locate data more quickly.

When index statistics get out-of-date, the index can be rendered useless or of less value than it was when it was created.

You can use auto update statistics to check and automatically update the index statistics when they get behind. With the **AUTO_UPDATE_STATISTICS** command you can specify if you want a **SAMPLE** or **FULLSCAN.** With the **SAMPLE** option you can tell auto update statistics to sample a portion of the rows and generate statistics from the sample - or you can tell it to do a full scan of the rows and generate statistics from that.

When you use auto update statistics and a query attempts to access a row -- if the optimizer sees that the statistics are out of date, it will put the query on hold and automatically update the statistics. After the statistics are updated, the query is completed.

You can use the **AUTO_UPDATE_STATISTICS_ASYNC** to update the statistics without putting the queries on hold. The update runs as a background thread. This is a database level setting.

Manually Creating Statistics
You can manually update statistics in the following ways:

- You can use the stored procedure **sp_createstats**.
- Use the **CREATE STATISTICS** command.
- Use the **UPDATE STATISTICS** command.
- Use an index rebuild to update statistics.

Designing a Solution to Govern Resources

Implementing Resource Governor

Since shared servers are common in SQL Server, you need a way to choose which workloads get the highest priority on each server instance and a way to monitor and trace those workloads.

The Resource Governor is new to SQL Server 2008. This powerful new tool allows you to manage SQL Server workload and system resource consumption. With Resource Governor you can specify limits on the amount of CPU and memory that incoming application requests can use.

You can configure Resource Governor in using T-SQL statements or in SSMS by expanding Management in the **Object Explorer**.

With Resource Governor you can balance workloads so that one workload can have better access to resources than another. It provides a way to segment SQL Server **RDBMS** (**Relational Database Management System**) workloads and to assign resources to those workloads.

For instance, you may want to give priority to certain applications that have to function even when the server is stressed—or, you may want to give preference to processing by certain users or groups. You can do these kinds of things with Resource Governor.

Note: You cannot segment server level or I/O resources with Resource Governor.

The T-SQL Resource Governor **DDL** (Data Definition Language) statements available are as follows:

- **CREATE WORKLOAD GROUP**
- **ALTER WORKLOAD GROUP**
- **DROP WORKLOAD GROUP**
- **CREATE RESOURCE POOL**
- **ALTER RESOURCE POOL**
- **DROP RESOURCE POOL**
- **ALTER RESOURCE GOVERNOR**

Resource Pools
Resource Pools allow you to segment SQL Server resources by CPU and memory and assign them to a pool. You can have up to 18 user-defined pools and two system pools. There is also an internal system pool used by SQL Server that is required. You can't modify the internal pool.

There is a default system pool that is required and this pool is modifiable. And you can assign multiple workload groups in each pool.

A **Workload Group** is a category of work assigned to a resource pool. With workload groups you can segment the workloads on SQL Server and assign these workloads to different pools. The number of workload groups you can have is unlimited and you can use them to segment the pool resources. You can also request specific limitations of the workload groups, like additional restrictions on CPU and memory.

You can also specify the **DOP** (Degree of Parallelism). Groups within a pool can be assigned a low or a high priority within the pool.

Note: A group can only be assigned to one pool.

A user **session** (or connection) will be evaluated when the user logs in and will be bound to a specific group until that session is finished. If the group is dropped during a session the session continues to use the resources specified for the group until the session ends.

A **Sessions Classifier** is a function that breaks the sessions up into groups or assigns sessions to groups. The sessions classifier is a user-defined function that is created in the master database and you can only have one classifier per instance of SQL Server. This function is changeable on-the-fly so if you create a poor classification, you can easily make corrections.

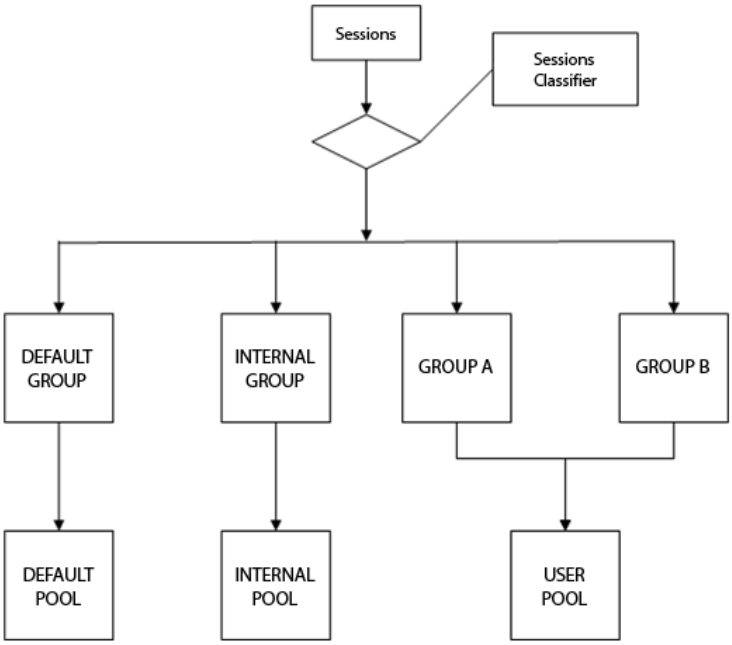


Figure 39: Understanding the Sessions Classifier Function

In the diagram above, sessions are filtered through the sessions classifier and are routed to the workload group that has been assigned by the sessions classifier function you create. Each new session is routed to the specified group (or the default group, if it's not identified in the sessions classifier).

Each workload group is assigned to a pool with certain system resources allocated for that pool. In the diagram above, GROUP A and GROUP B are USER POOL. GROUP A and GROUP B could be assigned to separate pools but neither can be assigned to more than one pool.

How Resource Governor Works
Resource Governor only uses resources that are allocated to SQL Server. Memory limitations only apply to memory granted to SQL Server.

100 percent of CPU resources are available except for the resources that are not allowed by the CPU affinity mask. CPU percentages that are segmented for pools only take effect when there is CPU contention. As long as there is no CPU contention, there is no need to limit and allocate resources.

The memory percentages, on the other hand, are set aside for a pool when the pool is created. You should divide memory resources carefully so as not to have too much unused memory in any one pool.

Workload Group Settings
The settings for a workload group only affect the group and nothing outside the group.
In a group you can set the following:

- **REQEST_MAX_MEMORY_GRANT_PERCENT** – the amount of memory a single request can use in the pool. This is relative to the maximum memory for the pool and the default is 25 percent. Large queries will run one at a time if the value is set at 50 percent or greater.
- **REQUEST_MAX_CPU_TIME_SEC** – the maximum CPU time for any query. If this time is exceeded, it won't stop the query, but it will cause the event - CPU threshold exceeded. The default value for this setting is zero - which is unlimited.
- **REQUEST_MEMORY_GRANT_TIMEOUT** – the maximum time a query will wait for a work buffer. This will fail if the timeout is reached.
- **MAX_DOP** – Maximum degree of parallelism. This overrides the sp_configure and the values for this setting are 0 to 64.
- **MAX_GROUP_REQUEST** – Maximum number of things running in a group. Zero is the default, which means unlimited.

Resource Governor Monitoring
Here is a list of DMVs you can use to monitor Resource Governor activities:
For configuration and stored values use:

- **sys.dm_resource_governor_configuration**
- **sys.dm_resource_governor_resource_pools**
- **sys.dm_resource_governor_workload_groups**

Other tools for management include using the dynamic management views for analysis. You should archive the dynamic management views periodically—then you can write procedures that will provide period based metrics.

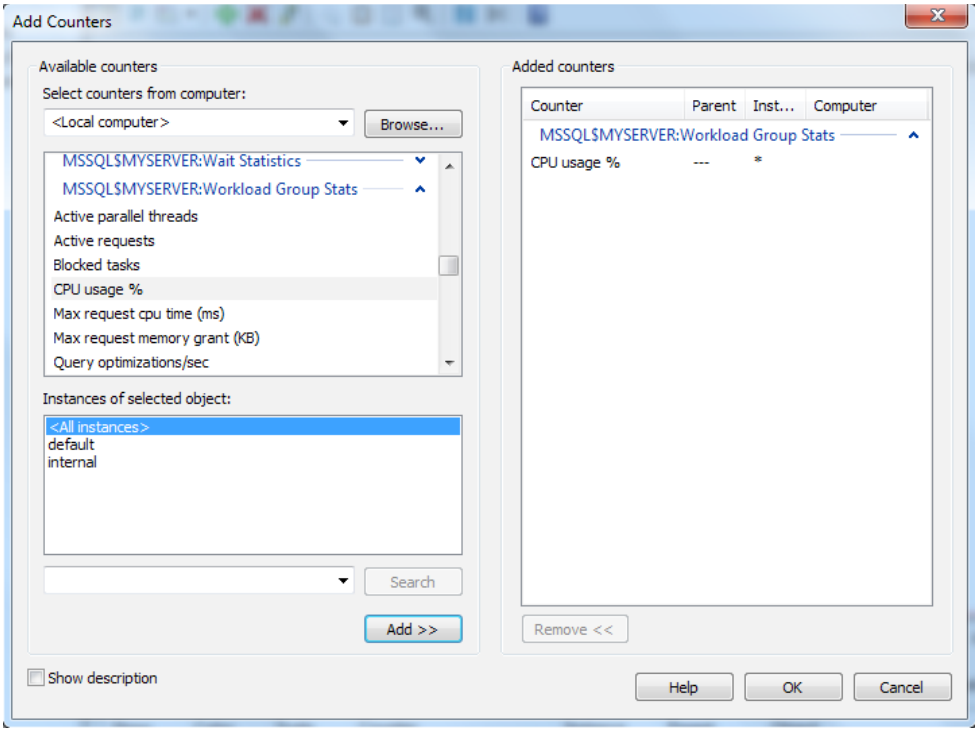


Figure 40: Adding Counters

You can also run 'perfmon' and click the green plus sign to add counters. In the screenshot above, you see that we have selected a counter for the work group stats for my default instance -- CPU usage %. We also selected all group instances to monitor.

Resource Governor Events
As we discussed earlier there is the CPU threshold exceeded event in errors and warnings. You can set up an alert for this event in SQL Server Agent as well.

There are also the following events:

- Sessions - PreConnect: starting event
- Sessions - PreConnect: completed event

There are Resource Governor performance counter groups where you can pull the:

- SQL Server: Resource Pool Stats
- SQL Server: Workload Group Stats

Some Resource Governor Usage Tips
The sessions classifier function allows you to segment the workload. You can use the classifier function to assign the workload to the following:

- SQL\Domain or the user
- Application name
- Host
- Server role
- Active Directory group that the user is a member of

You can use Resource Governor with dynamically changing workloads in response to an external stimulus or on a schedule.

Testing and Debugging Resource Governor
You should test the Resource Governor setup thoroughly before you put it into production. Because Resource Governor doesn't kick in unless there is an actual contention, you have to test with a background workload.

You should also test the sessions classifier function thoroughly especially if you have order-sensitive classifications. Make sure you test the classifier function for performance as well - because the time it takes to do the classification adds to the login time.

To debug problems that you discover when you test, use the DAC login and the DM views that we have discussed in this section. You can use System Monitor for debugging as well.

Follow the steps below to create a Resource Governor environment.

1. Create workload groups for your workloads.
2. Create a classification function.
3. Enable Resource Governor.
4. Monitor the Resource Governor.

Designing Policies with Policy Based Management

In this section we will discuss the scalability of PBM across multiple servers in the enterprise. When designing PBM, remember that there are hundreds and hundreds of facets. And you create conditions on the facets, and each condition is associated with a policy.

You can set evaluation conditions based on a facet and condition that can be: on demand policies, on a schedule and on change (which will either log the change or prevent the change). You can also set up alerts, but only for on demand policies.

To administer PBM you have to be a member of the PolicyAdministrator role on **msdb**.

PBM Scalability

To scale out PBM you must first set up a policy on an SQL Server engine. You may even want to install SQL Server engine dedicated to hosting your policies. Once the policies are set up, you can export the policies to XML files.

You can register yourself as a central management server -- or with Local Server Groups, you can register all of the servers that you want to distribute the policies to in a single group -- and then import these policies in a batch to multiple servers.

Designing a Data Compression Strategy

Page vs. Row Compression

Page compression includes row compression so it will take longer and use more resources.
Row compression mainly compresses the metadata in a row and it does compress well because prefixes are used in the metadata.

Page compression allows the opportunity to do more compression than row compression. And remember it's always a good idea to rebuild heaps once in a while because DML doesn't compress the data.

Partition and Index Compression

Partitions and indexes can be independently compressed using either "none", "row", or "page" compression. Indexes will usually compress well because the prefixes are likely to be the same.

Because partitions are mostly read-only, they are also good candidates for compression.

Designing a Management Automation Strategy

DDL Triggers

With DDL triggers you can respond to an event several ways. You can prevent an action from happening by rolling it back. You can also do other things inside the trigger. Remember a trigger is a special stored procedure so you have some control over what occurs in response to an event. You may use DDL triggers to audit or notify of a change occurring.

DDL triggers fire after a change occurs and during the transaction, so you have the opportunity to roll the change back. The triggers are synchronous and the cost is included in the cost of the original action.

Comparisons

Event notifications can fire from DDL or SQL trace events. You can use event notifications to log changes or to perform some action.

Unlike DDL triggers which happen synchronously, event notifications are asynchronous and run independent of whatever caused the event. Therefore event notifications do not use the transaction resources to respond to the event. And since they don't occur within the transaction, you cannot prevent the change from happening, there is no way to roll back the transaction.

WMI

Windows Management Instrumentation can manage any event generated by SQL Server. SQL Server Agent can query and respond to these events. This enhances the capability of SQL Server Agent. You can use the WQL language to query events that occur.

SQL Server Agent

SQL Server Agent as another tool for automation can do the following:

- Schedule and run jobs
- Create alerts that will respond to:
    - WMI events
    - Performance indicators
    - SQL Server errors
- Notify operators

Test Tip: Make sure you remember that when you setup job steps, you should be careful to enable the logging that's necessary for you to do proper debugging. You have the option to turn on or restrict logging during the setup.

PowerShell

PowerShell is a scripting language and SQL PowerShell is an extension of this language that allows you to navigate the SQL objects like the file system directory and file hierarchy. It uses familiar commands like dir, cd, rename and delete.

There are Cmdlets included that are unique to SQL Server. There is also an Invoke-SQLcmd command that can be used to run a T-SQL or XQuery scripts.

The Cmdlets that are available to SQL Server are:

- Invoke-SQLcmd.
- Invoke-PolicyEvaluation - evaluates policy compliance and reports it. This command can also be used to apply changes if policies are violated. You may want to use this to automate the evaluation of PBM.
- Encode-SQLName - it allows the use of characters that are available in SQL Server that are not supported in PowerShell.
- Convert-UrnToPath - PowerShell uses paths to identify objects so this command converts **URN**s (Uniform Resource Name) to paths for use in PowerShell.

To run SQL PowerShell you type SQLPS in the command line or right-click the server instance in SQL Server Management Studio. This loads the SQL Cmdlets and it runs in the restricted mode with no scripts allowed. You can use the Set-ExecutionPolicy Cmdlet and allow scripts to run if you need to.

Domain 7: Designing a Strategy for Data Distribution

Here you will learn how to distribute data and deploy packages across servers or in the same server. We will cover strategies for SQL Services Integration Service data distribution and security considerations to protect that data, using replication for data distribution and conflict detection and resolution.

Administering SQL Server Integration Services Packages

SQL Server Integration Services (SSIS) is one of many ways you can distribute data across the enterprise. SSIS packages can be stored in **msdb** and the file system. There are roles associated with **msdb** that allow you to access these packages.

The db_ssisadmin or sysadmin roles can do anything to any SSIS package. The db_ssisltduser role can create, view and execute only packages that belong to them. The db_ssisoperator role can list, view and execute any package but can't create or import packages.

Data Protection

You can also have reader and writer roles with packages. These roles are associated with security roles in **msdb** and can further limit access to specific packages.

Packages can have sensitive data stored in them, like an entire connection string or a password. XML nodes and variables may be tagged as sensitive. A developer may mark components or tasks as sensitive as well.

You need to be aware of the sensitive data in packages that are stores and distributed and take the necessary steps to protect that data.

You can set the property **DontSaveSensitive** to prohibit any sensitive data from being saved in a package. When you use this property all of the sensitive data is blanked out and the user must provide it - as in passwords for instance.

Another way to protect the sensitive data is to save it with **EncryptAllWithPassword**. In this case all of the data is encrypted and you will provide a password that will be used to access the package. The password will then be required for anyone who opens or runs the package.

You can also save the package with **EncryptAllWithUserKey** so that only the current user can open or run the package.

If you save the package with **EncryptSensitiveWithPassword**, the password must be used access the sensitive data. If a password is not provided, the package will be opened but the sensitive data will be blanked out and the package cannot be executed.

If you use **EncryptSensitiveWithUserKey**, access to the sensitive data is based on a key rather than a password. Without the key, the package will be opened but the sensitive data will be blanked out and the package cannot be executed.

Note: It is common practice not to save sensitive information in SSIS packages.

Deployment

Problem: In the process of package development, testing, QA and production, things like paths, logins and test information may change.

Solution: Store the configuration and make them readily available for deployment.
You can store the configurations via:

- XML files – includes the information needed for dev. Probably the best option.
- Variables – you would have to provide one variable for each configured item.
- SQL Server table – your deployment would have to include automatic scripting of the configuration information and then automatic scripting insertion of the configuration information as rows on a table in production.
- Environment Variables – the number of configuration items stored are limited by the available memory.
- Registry Entries – generally avoided because registry information is easily corrupted.

Troubleshooting Packages

For troubleshooting packages you can write and add event handlers in the package. You can also add error output that is standardized or customized. Adding logging to the package can provide additional audit information and things like failed row counts.

Adding checkpoints to your package can allow the package to rerun and restart at the point of failure.

Configuring Checkpoints

Since checkpoints are written as files, make sure you have the proper security on the file. You will have to provide CheckpointFilename and CheckpointUsage.

CheckpointUsage tells the package whether checkpoints are used.

The selections are:

- Never
- Always
- Exists (Only if the file exists)

Within the package you will need to set the property **SaveCheckpoints** as well.

Designing a Linked Servers Strategy

Linked servers provide a quick way to get access to data on another server.

You use user mapping on a local machine to users on the foreign (linked server) machine. You will need to provide a login and password for the foreign machine.

If you don't use user mapping you can choose to login without a context or you can use self mapping -- which is the default. Self mapping uses your identity on the local machine as your identity on the foreign machine.

You can also set it up so that everyone who uses the linked server has an identification on the foreign server that is a predefined login.

Delegation

If you use self mapping, you will be passing credentials from one server to another, this is called delegation. With self mapping you will be able to login on the local machine and then you can execute a linked server query on the foreign machine.

To set this up you must do some work on the local machine, on the foreign machine and on the client. On the client, the clients Active Directory Login must have access to both the local machine and the foreign machine. There is a property called account is sensitive and must not be delegated, this property cannot be selected - you want delegation. The client must use TCP/IP or named pipes to make the connection.

On the local machine a Service Principal Name must be registered by the domain administrator. The SQL Service account must be trusted for delegation and SQL Server must also use IP or named pipes.

Then you set up a linked server on the local machine to point to the foreign machine. To do this use **sp_addlinkedserver** 'server2', n 'SQL Server'. Next you set the linked server logins to self mapping using **sp_addlinkedserverlogin** 'server2', 'True'.

On the foreign machine a **Service Principal Name** must be registered by the domain administrator. The server must also use IP or named pipes.

SSMS Linked Server Setup

To create a linked server in SSMS expand Server Objects in Object Explorer, right click on Linked Servers and select New Linked Server.

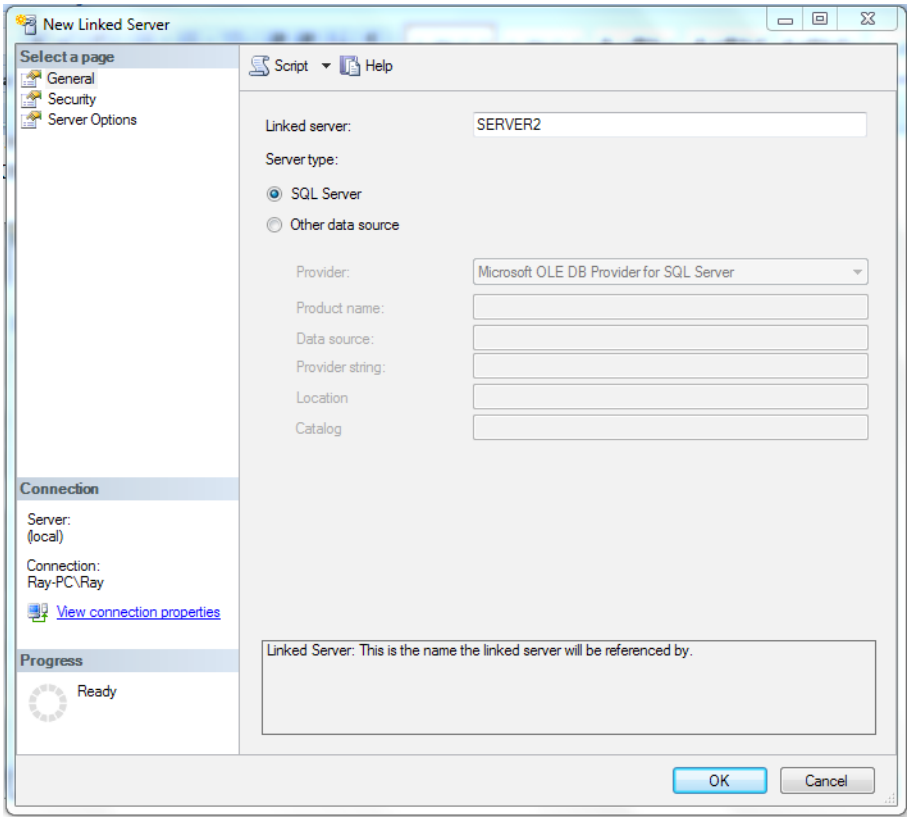


Figure 41: Creating a New Linked Server

You will get the New Linked Server window shown above. Select SQL Server and enter the name of the server you would like to link to.

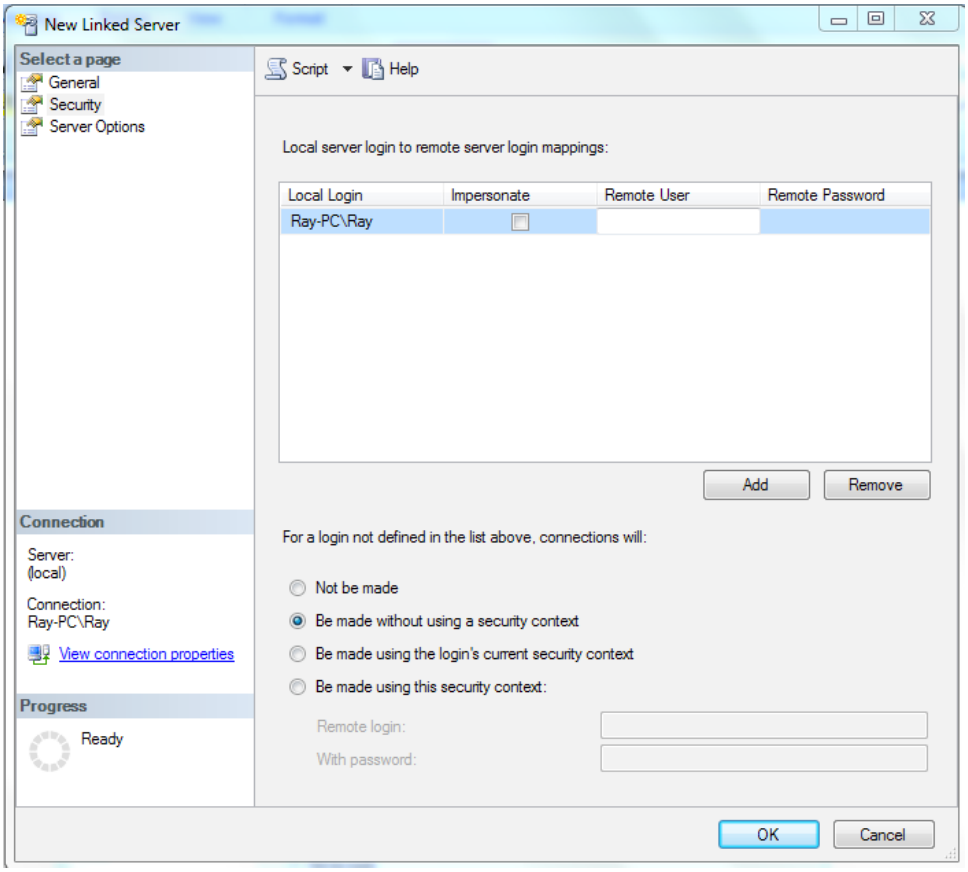


Figure 42: Setting Security for the Linked Server

Next, select Security to set up security for the linked server connection. Click Add to enter the login information. You can click the Local Login box for a list of available logins and select from there. You have the option to impersonate or you can enter a Remote User and a password.

Note: For remote user access, the password will always be sent unencrypted.

For all logins that are not mapped, you have some options on how these logins will be handled (Refer to the screenshot above):

- Not to be made - Unmapped users can't make a connection.
- Be made without using a security context - Not all providers will require a security context. Use this option if you wish users to connect and a security context is not required by the provider.
- Be made using login's current security context - Login with current login - delegation is required for this option.
- Be made using this security context - Enter remote login with a password.

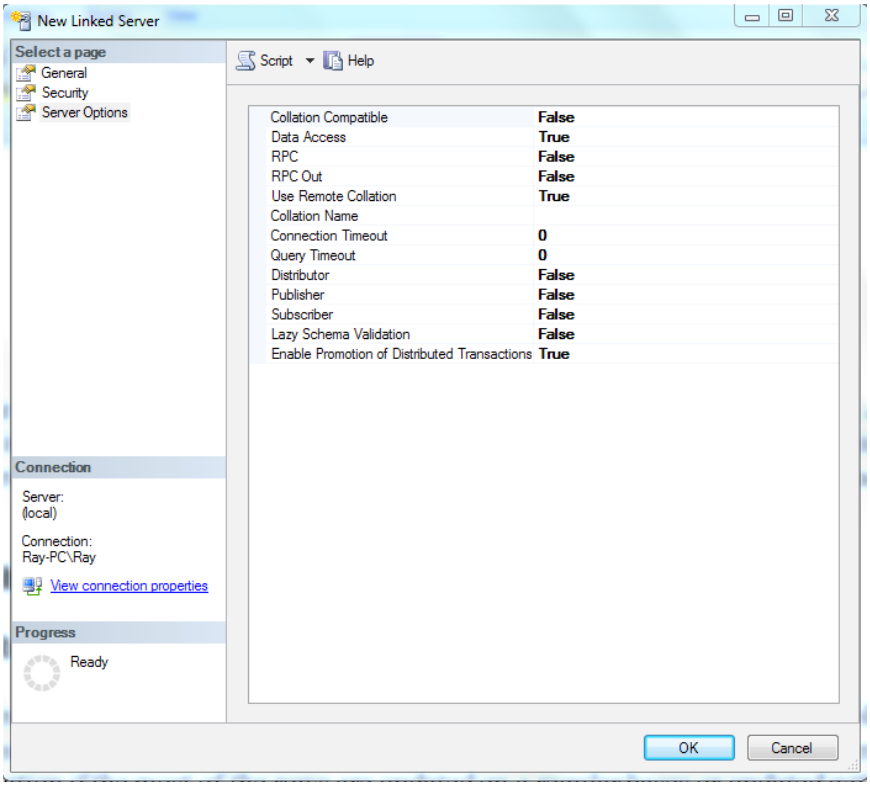


Figure 43: Linked Server Options

Under Server Options you can customize the connection. You do things like allow or disallow Distributed Transaction and select whether the collation is compatible between servers or not.

Designing a Replication Strategy for Data Distribution

Snapshot Replication

With snapshot replication, the entire table is refreshed; it's like taking a picture of the table. Interim changes are not updated as in transactional replication. You will want to consider snapshot replication if the most of the rows are updated on a regular bases or updated with nightly bulk load.

Transactional Replication

Transactional replication starts with a snapshot of the data and then updates are provided as they occur in near real time updates or on a schedule. Transactional replication works well with mostly connected systems and on systems with a reliable network.

One drawback to transactional replication is that the logs can grow large because they are not truncated until the transactions are delivered to the distributer.

LOBs are supported with transactional replication.

Also note that in SQL Server 2008, transactional replication supports partitioned tables well.

Peer to Peer Replication

Peer to peer replication is transactional replication except there is no hierarchy and all of the elements involved in the replication are peers. Peer to peer replication is intended to mostly scale reads in an environment where there are few or controlled writes.

Peer to peer also works best in an environment where there are mostly connected and reliable connections.

Merge Replication

Merge replication replicates the most recent version of each row and doesn't store each change that led up to the most recent version like transactional replication does.

Merge is intentionally bi-directional which lends itself to the possibility of conflicts. The possibility of conflicts is present any time there are multiple users of data and all of the users can update. Be sure you define your conflict detection and have a plan for conflict resolution.

Bulk Inserts

Now and then someone may have a requirement to do a bulk insert into a replicated table. Can that be done?

The answer is yes... The way to do this is when you do the bulk insert with the **BCP** (Bulk Copy Program), use the Fire_triggers option. Then after the bulk operation is completed, open the connection to the server and use the stored procedure **sp_addtabletocontents** with the name of the table that just received the bulk insert. This will allow the next merge to work as normal.

Row Filtering

With replication you can do row or column filtering. With transactional replication, the filtering is static and you supply a where clause. With merge replication you can either use static filtering, and it works just like with transactional replication -- or you can use dynamic filtering.

For dynamic filtering you will use one of the following:

- **Suser_Sname()**
- **Host_Sname()**
- User Defined Function which uses either of the above

This selection will determine which subset of rows will be delivered to each subscriber.

Column Filtering

To filter columns you can use vertical partitioning to include the targeted columns. When you replicate columns with filtering you can exclude columns that may have sensitive data or speed up replication by not replicating data that does not need to be replicated.

An exception to this exclusion is a column that contains a row that has been included in a where clause, in this case, the column must be included in the replication.

Transactional Replication Conflict Detection and Resolution

With transactional replication you will need conflict detection in place for queued updating. Where updates or inserts have been made to the same data from two locations -- or deletes have occurred in one place and updates in another.

The publisher wins in a conflict by default. You may also choose that the publisher wins and the subscription is reinitialized.

You can choose that the subscriber wins as well.

Peer to Peer Conflict Detection

Peer to peer conflict detection detects all combinations of insert, update and delete conflicts and issues peer to peer conflict alerts upon detection. Then the distribution agent stops and at this point you can choose to re-initialize the failed node from a backup or you can attempt to fix the conflict.

To fix a peer to peer conflict, on the distributer, use the stored procedure **sp_changepublication** and **select 'p2p_continue_on_conflict', 'true'**. Then restart the distribution agent and verify conflicts (the highest ID wins by default). Then run validation of data and fix the failures on the highest ID node. Allow them to replicate and then use the stored procedure **sp_changepublication** but this time select **'p2p_continue_on_conflict', 'false'** to turn off continue on conflict.

Merge Conflict Resolution

There are several options for merge resolver in SQL Server:

- Business logic handler in managed code
- COM-based custom resolver
- COM-based Microsoft provided resolver

Microsoft provided resolvers can resolve conflicts in several ways listed here;

- Additive, average, maximum or minimum.
- Datetime can be used to allow the earliest or latest entry to win the conflict.
- The text can be merged from both locations.
- The subscriber can be selected to win and you can choose whether the upload or download wins.
- A priority can be assigned to the publisher and each subscriber and the highest priority wins.
- You can write a custom stored procedure to pick who wins.

Health Monitoring

There are several options for health monitoring:

- Replication monitor (found in SQL Server Management Studio)
- Replication Managements Object and T-SQL capabilities
- SQL Server Management Studio
- Alerts are available for the replication agents
- System monitor

The replication monitor is the most important of the list above to determine the health of your system. The replication monitor will:

- Show you if subscriptions are slow
- Show latency times
- Tell you if the agents are running
- Provide information about expiring subscriptions
- Provide information about long running merges
- Tell you if you're processing fewer merge rows per second

With SQL Server Management Studio you can also start and stop replication agents, check the current status and view the last message logged for the Agents.

Practice Questions

Chapter 1

1. You are planning a SQL Server 2008 database. The database will include three data files named data1.mdf, data2.ndf, and data3.ndf. All three files will be stored in the Primary filegroup. The transaction log will be stored in a single file named data_log.ldf. The server on which the database will be deployed includes two physical drives for data storage. What is the best way to implement this database on the target system? Select the best answer.

   - ❍ A. Place the data files on one drive and the transaction log on the other drive.
   - ❍ B. Place the data files and the transaction log on one of the storage drives.
   - ❍ C. Place the data1.mdf file on one drive and the data2.ndf, data3.ndf, and data_log.ldf on the other drive.
   - ❍ D. Place the data1.mdf and data2.ndf files on one drive and the data3.ndf and data_log.ldf file on the other drive.

2. You have several SQL Server 2000 database server machines. Each machine includes a 600 MHz processor with 1 gigabyte of RAM. All applications support named instances, but most databases must run in a dedicated instance. You are planning an upgrade to SQL Server 2008. You are installing new servers that include two 2.1 GHz dual-core processors and 4 gigabytes of RAM. Other than virtualization, what can you do to reduce the number of physical machines required to support the SQL Server 2000 databases as you migrate them to SQL Server 2008? Select the best answer.

   - ❍ A. Consolidate the databases into fewer instances.
   - ❍ B. Install multiple named instances of SQL Server 2008 on each of the new servers.
   - ❍ C. Use Hyper-V to deploy the SQL Server 2008 servers.
   - ❍ D. Configure the new servers to multi-boot between server installations of Windows Server with a SQL Server instance installed in each boot option.

Chapter 2

1. You are designing a SQL Server 2008 implementation. The SQL Server instance will run on Windows Server 2008 Enterprise. Most users will access the server from Windows Vista or Windows XP clients. Several users must access the server from Linux clients. What authentication mode should be used and why? Select the best answer.

   - ❍ A. Windows authentication, because it supports all client types.
   - ❍ B. Mixed, because it can support all client types.
   - ❍ C. Windows, because Linux clients cannot connect to SQL Server regardless of the authentication type.
   - ❍ D. Mixed, because it is more secure than Windows authentication.

2. You are designing a database solution that runs on SQL Server 2008. The solution includes a view named vSalesQuarterly, a table named Sales and another table named Marketing. The Marketing table is owned by the dbo as is the vSalesQuarterly view. The Sales table is owned by DBA1. The vSalesQuarterly view includes columns from the Marketing and the Sales table. You have given Susan SELECT permissions on the vSalesQuarterly view and have not granted any permissions to the Sales or Marketing tables. What will occur when she queries the vSalesQuarterly view and includes columns from both table in her query? Select the best answer.

   - ❍ A. She will be granted access and all data will be retrieved.
   - ❍ B. She will be granted access to the Sales columns, but the Marketing columns will not be returned.
   - ❍ C. She will be denied access and the entire query will fail.
   - ❍ D. She will be granted access to the Marketing columns, but the Sales columns will not be retrieved.

3. You are creating a security audit plan document. In the document, you wish to list the security functions available in SQL Server 2008. You want to list a security function that returns the name of the current user context. Which of the following functions will provide this information? Choose all that apply.

   - ❍ A. RETURN_USER
   - ❍ B. CURRENT_USER
   - ❍ C. USERNAME
   - ❍ D. USER_NAME

4. When you use Transparent Data Encryption, in what scenarios will manual import of the certificate be required? Choose all that apply.

   - ❍ A. When moving the database to another server.
   - ❍ B. When recovering from a complete system failure.
   - ❍ C. When adding new rows to the database tables.
   - ❍ D. When selecting more than 100 rows from a table in the encrypted database.

Chapter 3

1. You are implementing a database mirroring solution in SQL Server 2008. You have five SQL Server 2008 server instances in your environment. All five are located in the same server room. SQL1 contains the primary copy of the database and SQL4 will contain the mirror. SQL2 and SQL3 are each running at capacity and may require an upgrade in the near future to keep up with demand. SQL5 runs at approximately 23 percent capacity. You want to allow SQL4 to automatically become the primary should SQL1 fail. What should you do?
Select the best answer.

   - ❍ A. Make SQL4 the witness server as well as the mirror server.
   - ❍ B. Make SQL2 the witness server.
   - ❍ C. Make SQL5 the witness server.
   - ❍ D. Configure SQL4 to automatically self-promote.

2. You must estimate the time required for an automatic failover using database mirroring in SQL Server 2008. The servers run the Enterprise Editions of both SQL Server 2008 and Windows Server 2008. Which two of the following items must be used in the calculation to estimate out-of-service time? Choose the two correct answers.

   - ❍ A. Failover time
   - ❍ B. Delay time
   - ❍ C. Error detection
   - ❍ D. Lag time

Chapter 4

1. As the DBA for West Water Main Works, LLC, you must develop a backup and restoration strategy. As part of this process, you have determined that backups should be verified. The backups are stored on a SAN. What is the keyword used in the RESTORE statement to verify the backup after the backup has been performed with a separate BACKUP DATABASE statement? You do not want to actually restore the database. Select the best answer.

   - ❍ A. RESTORE VERIFY
   - ❍ B. RESTORE VERIFYONLY
   - ❍ C. RESTORE DBCC
   - ❍ D. RESTORE CHECK

2. You are performing a verification of a backup. The RESTORE VERIFYONLY statement is being used. Which of the following are checks performed by RESTORE VERIFYONLY? Choose all that apply.

- ❍ A. Checksum (if present on the media).
- ❍ B. That the backup set is complete and all volumes are readable.
- ❍ C. That the backup set includes a page check sequence.
- ❍ D. Some header fields of database pages, such as the page ID.

3. You must plan the backup strategy for a large database names Sales. The database is stored in two different filegroups. The first filegroup, named Primary, contains three database files: Sales1.mdf (logical name Sales1), Sales2.ndf (logical name Sales2), and Sales3.ndf (logical name Sales3). The second filegroup, named Secondary, contains two database files: SalesArchive1. ndf (logical name SalesArchive1) and SalesArchive2.ndf (logical name SalesArchive2). The Secondary filegroup contains all read/write files and the database is in the simple recovery model. You are attempting to execute the following statement:BACKUP DATABASE SalesFILE = 'SalesArchive1'TO DISK = 'G:\SQL Server Backups\Sales\SalesArchive1.bck'You receive an error. The G: drive has plenty of free space and the backup process has proper permissions for access to the G: drive. What is the likely problem? Select the best answer.

- ❍ A. You must specify both SalesArchive1 and SalesArchive2 in the backup since the files are read/write.
- ❍ B. The backup must be stored on an NTFS volume.
- ❍ C. The BACKUP statement must specify NO_FILEGROUP_OVERRIDE.
- ❍ D. The BACKUP statement must include the WITH COMPRESSION clause when creating a file-only backup.

4. You are implementing a SQL Server 2008 backup solution. During the planning phase, you must make decisions regarding the backup media. You are considering the use of mirrored media sets. The server in question will run SQL Server 2008 Standard Edition and has access to three identical backup devices. Can you use mirrored media sets and why? Select the best answer.

- ❍ A. Yes, because SQL Server has access to more than one identical media device.
- ❍ B. Yes, because SQL Server 2008 supports mirrored media sets even with hard drive-based backups.
- ❍ C. No, because mirrored media sets are not supported in SQL Server.
- ❍ D. No, because SQL Server 2008 Standard edition does not support mirrored media sets.

5. You are planning for database backups. You must ensure that a copy of each backup will be available. Right now, backups are made to a single NAS device containing a RAID5 volume. You want to backup to more than one location. Which of the following options would help you achieve this? Choose all that apply.

   - ❍ A. Use the MIRROR TO clause of the BACKUP DATABASE statement.
   - ❍ B. Use multiple BACKUP processes in the same job.
   - ❍ C. Use the COPY ONLY backup option.
   - ❍ D. Store the backup files on mirrored media.

Chapter 5

1. You are designing a monitoring solution based on WMI. You want to read WMI information related to SQL Server 2008. Which of the following technologies can be used to read WMI data? Choose all that apply.

   - ❍ A. VBScript
   - ❍ B. PowerShell
   - ❍ C. System Monitor
   - ❍ D. WMIC

2. You are monitoring the performance of a SQL Server 2008 instance using operating system level tools. In the System Monitor, you want to add a counter that will show the buffer cache hit ratio. How do you add a counter in System Monitor? Select the best answer.

   - ❍ A. Click the add counter button.
   - ❍ B. Press the CTRL+M shortcut key.
   - ❍ C. Click File > New > Counter.
   - ❍ D. Click Insert > New > Counter.

3. You must access a SQL Server as the administrator. You want to use the dedicated administrator connection (DAC) feature. The SQL Server name is SQL1. How should you type the server name in the connection dialog? Select the best answer.

   - ❍ A. ADMIN:SQL1
   - ❍ B. DAC:SQL1
   - ❍ C. SQL1:ADMIN
   - ❍ D. SQL1:DAC

4. You have connected to a SQL Server 2008 server with the DAC for troubleshooting purposes. You want to run a query as a different user who is a standard user. How can you change context to the different user who is a standard user and not an administrator while in the DAC? Select the best answer.

- ❍ A. Use the EXECUTE AS statement.
- ❍ B. You cannot change context to a non-administrative user while in the DAC.
- ❍ C. Use the SU command.
- ❍ D. Execute the REVERT statement.

Chapter 6

1. You manage a SQL Server 2008 Enterprise Edition database server running on Windows Server 2008 Enterprise Edition. A database named Sales exists on this server and the server name is Sales-SQL. You want to compress the table named Tracking and all indexes related to this table. The table has a clustered index and two non-clustered indexes. You execute the following code:ALTER TABLE TrackingREBUILD WITH (DATA_COMPRESSION = PAGE); What else should you do in order to achieve your desired goal? Select the best answer.

- ❍ A. Compress the non-clustered indexes.
- ❍ B. Compress the clustered index.
- ❍ C. Drop and recreate all indexes.
- ❍ D. Run the RECONFIGURE command.

2. You are creating a condition for use by Policy-Based Management in SQL Server 2008. You want to check for Names that begin with the string 'Tra'. What operator should you use? Select the best answer.

- ❍ A. =
- ❍ B. LIKE
- ❍ C. <
- ❍ D. >

3. You are planning the strategy and design of a Policy-Based Management implementation. You must plan for the backup of the policies, conditions, and other policy-related objects. What system database should be backed up to backup the policies? Select the best answer.

- ❍ A. Msdb
- ❍ B. Master
- ❍ C. Resources
- ❍ D. Policies

4. As part of your SQL Server 2008 management automation strategy, you plan to use PowerShell scripts. You will use the sqlps utility to launch the PowerShell with the SQL Server 2008 snap-ins pre-registered. Which of the following can you NOT do by default in the PowerShell instance launched by sqlps? Select the best answer.

- ❍ A. Execute cmdlets.
- ❍ B. Run scripts.
- ❍ C. Interactively run PowerShell commands.
- ❍ D. Use SQL Server provider paths.

Chapter 7

1. Sources, transformations, and destinations make up which component of an Integration Services package? Select the best answer.

- ❍ A. Control flow
- ❍ B. Connection managers
- ❍ C. Data flow
- ❍ D. Event handlers

2. You want to use a linked server to return data for client requests. The clients will connect to SQL1 and three other servers will be available as linked server objects on SQL1. The three servers are named SQL2, SQL3, and SQL4. Where must the OLE DB providers be installed in order for the queries to run through SQL1? Select the best answer.

- ❍ A. SQL1
- ❍ B. SQL2
- ❍ C. SQL3
- ❍ D. SQL4

3. You are planning for replication in a SQL Server 2008 environment. What are the three key types of replication servers? Select the three correct answers.

- ❍ A. Witness
- ❍ B. Publisher
- ❍ C. Subscriber
- ❍ D. Distributor

4. You are implementing merge replication. SQL Server 2008 Enterprise Edition is used on all servers involved in the replication topology. Two tables are involved in the determination of changes needing to be synchronized between publishers and subscribers. What are these two tables? Select the best TWO answers.

- ❍ A. MSmerge_tune
- ❍ B. MSmerge_genhistory
- ❍ C. MSmerge_contents
- ❍ D. MSmerge_rules

Answers & Explanations

Chapter 1

1. Answer: A

Explanation A. Correct. Since no specific performance objectives are mentioned in the question, it is best to place the transaction log on a separate drive from the data files for maximum recoverability.

Explanation B. Incorrect. This configuration would provide neither optimal performance nor recoverability.

Explanation C. Incorrect. While this is possible, recoverability would be reduced because the transaction log is on the same drive as a portion of the data files.

Explanation D. Incorrect. While this is possible, recoverability would be reduced because the transaction log is on the same drive as a portion of the data files.

2. Answer: B

Explanation A. Incorrect. The question indicates that most databases require a dedicated instance.

Explanation B. Correct. By installing multiple named instances, you can consolidate four to six of the SQL Server 2000 machines into one of the new machines.

Explanation C. Incorrect. Hyper-V is a virtualization solution.

Explanation D. Incorrect. While you could configure such a scenario, the only instance of SQL Server that would be available is the one that is currently booted.

Chapter 2

1. Answer: B

Explanation A. Incorrect. Windows authentication mode can only authenticate Windows clients.

Explanation B. Correct. When supporting Linux or Mac clients, it is common to use mixed mode so that the Linux and Mac clients can use SQL Logins instead of Windows accounts.

Explanation C. Incorrect. If mixed mode is selected, Linux clients can authenticate using SQL Logins.

Explanation D. Incorrect. Windows authentication is more secure that SQL Logins, but you may be required to support SQL Logins if non-Microsoft clients are used.

2. Answer: C

Explanation A. Incorrect. The ownership chain is broken at the Sales table and Susan will be denied access because of this.

Explanation B. Incorrect. The ownership chain is broken and the entire query will fail.

Explanation C. Correct. Because the ownership chain is broken at the Sales table, she will be denied access for the entire query.

Explanation D. Incorrect. The ownership chain is broken and the entire query will fail.

3. Answers: B, D

Explanation A. Incorrect. No such function exists in SQL Server 2008 by default.

Explanation B. Correct. Both CURRENT_USER and USER_NAME will return the current user context.

Explanation C. Incorrect. No such function exists by default in SQL Server 2008.

Explanation D. Correct. Both CURRENT_USER and USER_NAME will return the current user context.

4. Answers: A, B

Explanation A. Correct. If you detach the database from the original server and attach it to another instance, you will have to provide the certificate used to encrypt the database encryption key.

Explanation B. Correct. The backup is also encrypted so you will need the certificate to decrypt the encryption key during recovery.

Explanation C. Incorrect. TDE is transparent in that you do not need to be aware of the encryption/decryption processes during normal database operations.

Explanation D. Incorrect. TDE is transparent in that you do not need to be aware of the encryption/decryption processes during normal database operations.

Chapter 3

1. Answer: C

Explanation A. Incorrect. The mirror server instance cannot be the witness.

Explanation B. Incorrect. SQL2 is already heavily utilized. SQL5 should be the witness server in this scenario.

Explanation C. Correct. SQL5 has the extra resources to handle the minimal additional requirements of being a witness server. A witness server is required in order to implement automatic promotion of the mirror to primary.

Explanation D. Incorrect. SQL4 cannot promote itself without authorization from a witness server.

2. Answers: A, C

Explanation A. Correct. The failover time is the amount of time it takes for the failover to occur. Failover time consists mainly of the time that the former mirror server requires to roll forward any log remaining in its redo queue. Enterprise Editions of SQL Server 2008 may use multiple threads to perform the redo if more than 5 processors exist in the server.

Explanation B. Incorrect. No unnecessary delay time is added to the failover process.

Explanation C. Correct. The error detection time is the time required for the mirror server to recognize that the principal server instance has failed. The time for the system to notice an error depends on the type of error; for example, a network error is noticed almost instantly, while noticing a server hang by default takes 10 seconds, which is the default timeout period.

Explanation D. Incorrect. No such time interval is defined for the calculation of out-of-service time.

Chapter 4

1. Answer: B

Explanation A. Incorrect. The actual command is RESTORE VERIFYONLY.

Explanation B. Correct. With the RESTORE VERIFYONLY command, you can verify a backup without restoring it.

Explanation C. Incorrect. DBCC is a separate command used to test the integrity of databases and tables.

Explanation D. Incorrect. No such statement exists. The proper command is RESTORE VERIFYONLY.

2. Answers: A, B, D

Explanation A. Correct. The checksum is usually present on backup media and, if it is, RESTORE VERIFYONLY will check this.

Explanation B. Correct. If the backup set is incomplete (not all files or media are available), you cannot restore the database. RESTORE VERIFYONLY does ensure that all backup set volumes are available.

Explanation C. Incorrect. No such defined item is included in SQL Server backups or checked by RESTORE VERIFYONLY.

Explanation D. Correct. The RESTORE VERIFYONLY process does check header fields in the database pages to ensure readability.

3. Answer: A

Explanation A. Correct. The proper command would look something like this: BACKUP DATABASE Sales FILE = 'SGrp1Fi2', FILE = 'SGrp2Fi2' TO DISK = 'G:\SQL Server Backups\Sales\SalesGroup1.bck'.

Explanation B. Incorrect. No such requirement exists.

Explanation C. Incorrect. No such keyword exists for the BACKUP statement.

Explanation D. Incorrect. The WITH COMPRESSION clause is a valid option, but no such requirement exists.

4. Answer: D

Explanation A. Incorrect. SQL Server must run Enterprise edition in order to use mirrored media sets.

Explanation B. Incorrect. While SQL Server 2008 supports mirrored media sets that are hard drive-based, it must run SQL Server 2008 Enterprise edition and this scenario calls for Standard edition.

Explanation C. Incorrect. Mirrored media sets are supported in SQL Server 2005 Enterprise edition and higher.

Explanation D. Correct. Only Enterprise edition supports mirrored media sets.

5. Answers: A, B, D

Explanation A. Correct. The MIRROR TO clause is used to backup a database to more than one place in duplicate.

Explanation B. Correct. You can simply create more than one backup within the same job one right after the other.

Explanation C. Incorrect. Copy only backups do not result in more than one backup copy.

Explanation D. Correct. If you use RAID1, mirroring, the backup is automatically stored in two or more locations.

Chapter 5

1. Answers: A, B, D

Explanation A. Correct. You can create WSH scripts that use VBScript to read WMI information.

Explanation B. Correct. PowerShell can read WMI information.

Explanation C. Incorrect. System Monitor can read performance counters within performance objects, but it does not work with WMI information related to SQL Server 2008.

Explanation D. Correct. WMIC is a Windows command line tool for working with WMI information.

2. Answer: A

Explanation A. Correct. The add counter button looks like a plus sign. You can also press CTRL+I as a shortcut.

Explanation B. Incorrect. CTRL+M is a shortcut to add a new snap-in in an MMC interface. The shortcut to add a new counter is CTRL+I.

Explanation C. Incorrect. Being an MMC snap-in, the System Monitor does not have functions on the File menu.

Explanation D. Incorrect. No Insert menu exists in the MMC.

3. Answer: A

Explanation A. Correct. By prefacing the server name with ADMIN:, you are indicating the desire for a DAC connection.

Explanation B. Incorrect. The proper syntax is ADMIN:server_name.

Explanation C. Incorrect. The proper syntax is ADMIN:server_name. The key is to place ADMIN: before the server name.

Explanation D. Incorrect. The proper syntax is ADMIN:server_name. The key is to place ADMIN: (not DAC:) before the server name.

4. Answer: B

Explanation A. Incorrect. You cannot change context to another user while in the DAC unless that user is an administrator.

Explanation B. Correct. To prevent security problems, context switching to non-administrative users is not supported while in the DAC.

Explanation C. Incorrect. No such command exists in SQL Server.

Explanation D. Incorrect. The REVERT statement is used to switch back to the original user context after the EXECUTE AS statement has been utilized.

Chapter 6

1. Answer: A

Explanation A. Correct. The non-clustered indexes are not automatically compressed. You must compress each one individually.

Explanation B. Incorrect. The clustered index is the table and it is compressed through the command issued.

Explanation C. Incorrect. The clustered index is already compressed. You must modify the non-clustered indexes to compress them.

Explanation D. Incorrect. The RECONFIGURE command is needed after running an sp_configure change.

2. Answer: B
Explanation A. Incorrect. The = operator returns exact matches and does not support wildcards.

Explanation B. Correct. The LIKE operator works just like the SQL LIKE operator in WHERE clauses.

Explanation C. Incorrect. The < operator is used to evaluate numerics.

Explanation D. Incorrect. The > operator is used to evaluate numerics.

3. Answer: A
Explanation A. Correct. The policy objects are stored in the msdb database.

Explanation B. Incorrect. The policy objects are not stored in the master database, but they are stored in the msdb database.

Explanation C. Incorrect. The policy objects are stored in the msdb database and not the resources database.

Explanation D. Incorrect. No such system database exists.

4. Answer: B
Explanation A. Incorrect. Cmdlets can be executed in the sqlps PowerShell instance.

Explanation B. Correct. By default, the sqlps PowerShell instance does not support the running of scripts. You can use the Set-ExecutionPolicy cmdlet to enable running signed scripts, or any scripts.

Explanation C. Incorrect. Interactive capabilities are enabled by default.

Explanation D. Incorrect. You can browse SQL Server like a directory tree using the SQL Server provided paths by default.

Chapter 7
1. Answer: C
Explanation A. Incorrect. The control flow component consists of tasks and containers.

Explanation B. Incorrect. The connection managers connect to different types of data sources to extract and load data.

Explanation C. Correct. The data flow component does consist of these items.

Explanation D. Incorrect. The event handlers run in response to the run-time events that packages, tasks, and containers raise.

2. Answer: A

Explanation A. Correct. The OLE DB providers must be installed locally on the access server in order to query the linked servers.

Explanation B. Incorrect. They must be installed on SQL1 in this case.

Explanation C. Incorrect. They must be installed on SQL1 in this case.

Explanation D. Incorrect. They must be installed on SQL1 in this case.

3. Answers: B, C, D

Explanation A. Incorrect. The witness server is used with database mirroring.

Explanation B. Correct. The publisher is the source of the replication data publication.

Explanation C. Correct. The subscriber receives data (and may merge data) from the publisher.

Explanation D. Correct. The distributor acts as a mediator between the publisher and the subscriber. The distributor role and the publisher role may be on the same server.

4. Answers: B, C

Explanation A. Incorrect. No such table exists.

Explanation B. Correct. The MSmerge_genhistory table contains one row for each generation that a Subscriber knows about (within the retention period). It is used to avoid sending common generations during exchanges and to resynchronize Subscribers that are restored from backups. This table is stored in the publication and subscription databases.

Explanation C. Correct. The MSmerge_contents table contains one row for each row modified in the current database since it was published. This table is used by the merge process to determine the rows that have changed. This table is stored in the publication and subscription databases.

Explanation D. Incorrect. No such table exists.