

Guide to SQL Server 2012

Database Administration

by

Mukesh B

Preface _____

This book entitled "Guide to SQL Server Database Administration" for administrators; this book is loaded with unique tips, tricks and case studies for handling the most difficult SQL Server administration issues.

SQL Server 2012 represents a significant jump forward in scalability and performance. Database administrators will need to master new methods for effectively managing increasingly large and complex SQL Server environments.

This book is extensively covered each and every topic under database administration with unique tips, tricks and case studies including Solid architecture, installing, migration, managing and monitoring SQL Server, AlwaysOn automating administration, high availability, clustering, performance tuning and many more... You'll learn various tools that are available to you with the 2012 release. With a solid understanding you'll be able to confidently handle even the most difficult SQL Server admin issues.

Who this book for

This book is for beginners as well as experienced database administrators who are interested in learning the best practices for administering SQL Server 2012.

Finally, I would like to thank and express my appreciation to my family members for their patience and encouragement in bringing out this book.

Any suggestions for the improvement of the book are welcomed. Suggestions can be emailed to mssqldba2012@gmail.com

- Mukesh B

Contents

Introduction to SQL Server 2012

Overview on RDBMS and Beyond Relational	1
System databases	3
What's New in SQL Server 2012	5
DBA Check List	15

Chapter 1: SQL Server Component Overview

Components of SQL Server	17
Protocols	18
Tabular Data Stream	18
The Relational Engine	19
The Query Optimizer	19

Chapter 2: Pages & Extents

Pages	21
Extents [Uniform & Mixed]	21
Managing Extent Allocations	22
Tracking Free Space	22

Chapter 3: Files and File groups

Database Files	23
1. Primary data files	23
2. Secondary data files	23
3. Log files	23
Database File groups [Primary & User-defined]	23

Chapter 4: Thread and Task Architecture

Allocating threads to CPU	24
Affinity Mask	25
Using Light weight pooling option	26
Max Degree of Parallelism	27
Hot Add CPU	27

Chapter 5: Memory Architecture

32-bit Vs 64-bit Architecture	28
Dynamic Memory Management	29
Effects of min and max server memory	30
Using AWE	31
Buffer Management	33
The Buffer Pool and the Data Cache	33
Deference between chckpond and LazyWriter	34

Chapter 6: SQL server 2008R2/2012 Installation

Installing SQL server 2008R2	35
Planning the System/Pre-Requisites	35
Installing SQL server 2008R2	38
SQL server 2012 Installation	53
Best Practices on Installation	74
Case Study on Moving System Databases	74

Chapter 7: Upgrading to SQL server

Upgrading the server by applying service packs	82
SQL Server 2008 R2 SPI	83
Upgrading to SQL Server 2012	89
SQL Server Upgrade Advisor	92
In-Place Upgrade to SQL Server 2012	95
Best Practices to follow while upgrading	97

Chapter 8: Configuring SQL Server

Network Protocols from SQL Server configuration manager	98
Dedicated Administrator Connection [DAC]	100
Using DAC with SQL Server Management Studio	102
Configuring Database Mail	105
Central Management Services	109
Best Practices on configuration/Database settings	119
Tempdb Configuration	120
Case Study: Possible failures while shrinking tempdb in SQL Server	120

Chapter 9: Managing services

Starting and Stopping Services through	124
1. Configuration manager	125
2. Net Command	125
3. Command Prompt [sqlsvr.exe]	125
Start Up parameters	126
Starting SQL server in single user mode	127
Starting SQL server with minimal configuration	127
Case study 1: Resolving tempdb error [Error: 5123]	128
Case study 2: Rebuilding System databases	131
Case Study 3: Changing SQL Server Collation in 2008	133

Chapter 10: Migrating SQL server

Difference between in-place & Side by Side Migration/Upgradation	136
Migrating database by using Back and restore method	136
Migrating database by using Copy Database Wizard	143
Migrating database by using Attach and Detach Method	152
Migrating Logins [Fixing Orphaned Users]	154
Import & Export	156
Case study : Migrating jobs & Logins by using SSIS	166

Chapter 11: Security

Security Architecture	172
Fixed Server Roles	173
Availability Groups	174
User Defined Server Roles	174
Instance-Wide Settings	175
Database Securables	177
Creating Logins	178
Creating credentials	179
Fixed Database Roles	180
Database users	181
Permissions	182
What is contained Database?	182
How to Create Contained Database	183
Best Practices on security	186
Case study 1: Granting limited column level permissions	188
Case study 2: Recovering SA password	192
Case study 3: Grant exec permissions on single SP	192
Case study 4: Grant exec permissions on all SP	192
Case study 5: Recover access to a SQL Server instance	192

Chapter 12: Automating Administrative Tasks

About SQL server Agent	195
Agent Security	195
Creating Jobs	196
Alerts and Operators	200
Creating Maintenance Plans	203
Best Practices on Job maintenance	209

Chapter 13: Monitoring SQL Server

Tools and Techniques for Monitoring	210
Server Reports	210
Database Reports	212
Using Sp_configure	212
Error Logs	213
Activity Monitor	214
Monitoring Process in T-SQL	219
Case Study: How to create new error log or recycle error logs in SQL server	220
Best Practices on Monitoring	220

Chapter 14: Transaction Log Architecture

Transaction Log Logical Architecture	221
Transaction Log Physical Architecture	221
Checkpoint Operation	223
Write-Ahead Transaction Log	224
Transaction log truncation	225
How transaction log truncation works	225
Shrinking the log file	227

Chapter 15: Backup & Restore:

How Backup Works	228
Recovery Models	228
Types of backups	229
Recovery states	233
Case study 1. Backup & Restore for SQL Server 2012	234
Case study 2. How to restore a Suspect database	240
Case study 3: Backup and restore of the Resource database:	240
Case study 4: How to recover the database without having ldf file	241

Case study 5: Recovering Crashed SQL Server with existing data files	242
Case study 6: How to Restart an Interrupted SQL Server Database Restore	242
Case study 7: Point in Time Recovery Using New Timeline Feature	244
Best Practices on Backup & Recovery	251
Chapter 16: Log Shipping	
Log shipping overview	253
Pre-requisites/Log-Shipping Process	253
Deploying Log Shipping	258
Logs shipping Role changing [Fail-Over]	266
Frequently Raised Errors In Log-Shipping	267
Best Practices on Log-Shipping	271
Case Study 1: How to add files to a log-shipped database.....	271
Case Study 2: How to monitor Log Shipping for SQL Server Database	272
Chapter 17: Database Mirroring	
Overview of Database Mirroring	277
Operating Modes in Database Mirroring	278
Pre-Requisites for Database Mirroring	278
Deploying Database Mirroring	279
Advantages & Disadvantages of database mirroring	289
Mirroring Vs Clustering	289
Database Snapshots	291
Creating database snapshot	292
Best practices on Mirroring	293
Case study 1: Login Failures while connecting to new principal	294
Case study 2: Moving mirrored database files	294
Chapter 18: Replication	
Replication terminology	299
Replication Agents	301
Configuring distributor	302
Creating publication	307
Creating subscriptions	312
Peer to peer replication	314

Case study: Initialize from Backup for Transactional Replication	328
Best Practices on Replication	329

Chapter 19: SQL Server Clustering

Overview of Windows Server Failover Clustering	330
Topolgy and Architecture of a SQL Server Custer	332
Health Detection	334
Resource Group Properties	337
Installing the Failover Cluster Feature	
and Tools in Windows Server 2012	338
Creating a Windows Server 2012 Failover Cluster	346
Common SQL Server Failover Deployments	351
Advantages of SQL Server 2008 on Windows	
Server 2008 Clustering	354
Adding a node on a SQL Server 2008 Failover Cluster	368
Applying Patches on a SQL Server 2008 Cluster	372
Adding a Node on a SQL Server 2012 Multi-Subnet Cluster	391
Best Practices on Clustering	395
Case study 1: Changing the Network Name of a SQL Server	
Failover Cluster	386
Case study 2: Changing the Service Accounts for a SQL Server	
Failover Cluster	397
Case study 3: Changing the IP Address of a SQL Server	
Failover Cluster	397
Case study 4: How to add storage to Clustered Shared Volumnes in	
Windows Server 2012	398
Case study 5: How to start SQL Sever clustered instance in	
Single user mode	400
Case study 6: How to crate cluster logs in windows Server 2008/R2?	404
Case Study 7: Force a WSFC Cluster to Start Without a Quorum	407

Chapter 20: Always on Availability Groups

Overview	408
Terminology	408
Advantages	409
Availability Group Replicas and Roles	410
Types of Failover Supported	411
Allowing Read-only Access to Secondary Replicas	412

Pre-Requisites for Configuring the AlwaysOn	
Availability Groups	413
Availability Group Configuration	413
Always Group Dashboard	423
Add/Remove Replica	425
Add/Remove Database	426
Availability Group Failover Operation	427
Suspended an Availability Database	428
Resume an Availability Database	429
Active Secondary For Secondary Read-only	430
Backup on The Secondary Replica	431
Moniotring and Troubleshooting	432

Chapter 21: High Availability: Interoperability and Coexistence

Database Mirroring and Log Shipping	434
Setting Up Mirroring and Log Shipping Together	436
Database Mirroring and Database Snapshots	437
Database Mirroring and Failover Clustering	438
Replication and Log Shipping	440
Log Shipping with Transactional Replication	441
Replication and Database Mirroring	442
Configuring Replication with Database Mirroring	443
Failover Clustering and AlwaysOn Availability Groups	444
SQL Server Failover Cluster Instances (FCIs) and Groups	445

Chapter 22: Optimizing SQL Server

Policy based Management	447
Create a Condition	449
Create a Policy	450
Evaluate a Policy	452
Resource Governor Components	453
Putting it all together	454
Resource Governor's Catalog Views and Dynamic Management Views	458
Change Data Capture	459
Compression Techniques in SQL Server	464
Backup Compression	469
Partitioning	472
Creating Partitioned Tables	472
Partitioned Index	475
Case Study on Partitioning	477

Chapter 23: Indexing

Table & Index Architecture	483
Clustered Index Structures	485
Non Clustered Index Structures	486
Understanding Indexes	488
How Index are used by SQL Server	490
Index Fragmentation in SQL Server	493
Column Store Index Overview	495
How is Data Stored when using a Columnstore Index?	496
Creating a Columnstore Index	497
Best Practices on Indexing	499
Case Study 1: Is your indexes are being used effectively?	500
Case Study 2: Analyze and Fix Index Fragmentation in SQL Server 2012	507
Case Study 3: How to Identity Missing Indexes Using SQL Server DMVs..	514

Chapter 24: Performance Tuning

Introduction to Performance Troubleshooting Methodology	521
Factors That Impact Performance	521
CPU-Related Configuration Options	522
Memory - Related Configuration Options	524
Database Truning Advisor - DTA	524
Best Practices on Performance Tuning	528
Case Study 1: How to Collect Profiler Data for Correlation Analysis	529
Case Study 2: How to detect Deadlock in SQL Server by using profile	551
Case Study 3: SQL Server: Performance Conters - Thresholds	554
Case Study 4: MAXDOP Settings to Limit Query to Run on Specific CPU ..	569

ANNEXURE -I

Dynamic Management Views	572
---------------------------------------	-----

ANNEXURE -II

System Stored Procedures	579
---------------------------------------	-----

ANNEXURE -III

Useful Scripts	589
-----------------------------	-----

ANNEXURE -IV

Frequently Asked Questions	596
---	-----

Introduction to SQL Server 2012

What is SQL Server 2012/RDBMS?

As you most likely know, SQL Server 2012 is primarily thought of as a *Relational Database Management System (RDBMS)*. It is certainly that, but it is also much more.

SQL Server 2012 can be more accurately described as an *Enterprise Data Platform*. It offers many new features and even more enhanced or improved features from previous editions of the product. In addition to traditional RDBMS duty, SQL Server 2012 also provides rich reporting capabilities, powerful data analysis, and data mining, as well as features that support asynchronous data applications, data-driven event notification, and more.

Database Engine

The Database Engine is the primary component of SQL Server 2012. It is the Online Transaction Processing (OLTP) engine for SQL Server, and has been improved and enhanced tremendously in this version. The Database Engine is a high-performance component responsible for the efficient storage, retrieval, and manipulation of relational and Extensible Markup Language (XML) formatted data.

SQL Server 2012's Database Engine is highly optimized for transaction processing, but offers exceptional performance in complex data retrieval operations. The Database Engine is also responsible for the controlled access and modification of data through its security subsystem. SQL Server 2012's Database Engine has many major improvements to support scalability, availability, and advanced (and secure) programming objects.

Analysis Services

Analysis Services delivers Online Analytical Processing (OLAP) and Data Mining functionality for business intelligence applications. As its name suggests, Analysis Services provides a very robust environment for the detailed analysis of data. It does this through user-created, multidimensional data structures that contain de-normalized and aggregated data from diverse data sources (such as relational databases, spreadsheets, flat files, and even other multidimensional sources).

Reporting Services

Reporting Services is a Web service-based solution for designing, deploying, and managing flexible, dynamic Web-based reports, as well as traditional paper reports. These reports can contain information from virtually any data source. Because Reporting Services is implemented as a Web service, it must be installed on a server with Internet Information Services (IIS). However, IIS does not have to be installed on a SQL Server. The Reporting Services databases are hosted on SQL Server 2012, but the Web service itself can be configured on a separate server.

Integration Services

SQL Server Integration Services (SSIS) is Microsoft's new enterprise class data Extract, Transform, and Load (ETL) tool. SSIS is a completely new product built from the ashes of SQL Server 2000's Data Transformation Services (DTS). SSIS offers a much richer feature set and

the ability to create much more powerful and flexible data transformations than its predecessor. This huge improvement, however, is not without a cost. SSIS is a fairly complex tool and offers a completely different design paradigm than DTS. Database administrators adept at the former tool are very often intimidated and frustrated by the new SSIS. Their biggest mistake is in thinking that Integration Services would just be an upgrade of Data Transformation Services.

Replication Services

SQL Server 2012 Replication Services provides the ability to automate and schedule the copying and distribution of data and database objects from one database or server to another, while ensuring data integrity and consistency. Replication has been enhanced in SQL Server 2012 to include true Peer-to-Peer replication, replication over HTTP, the ability to replicate schema changes, and, very interestingly, the ability to configure an Oracle server as a replication publisher.

SQL Server 2012 Services

SQL Server runs as a service. In fact, it runs as several services if all the different features of the product are installed. It is important to know what service is responsible for what part of the application so that each service can be configured correctly, and so that unneeded services can be disabled to reduce the overhead on the server and reduce the surface area of SQL Server.

MSSQLServer (SQL Server)

The MSSQLServer service is the database engine. To connect and transact against a SQL Server 2012 database, the MSSQLServer service must be running. Most of the functionality and storage features of the database engine are controlled by this service.

The MSSQLServer service can be configured to run as the local system or as a domain user. If installed on Windows Server 2003, it can also be configured to run under the Network System account.

SQLServerAgent (SQL Server Agent)

This service is responsible for the execution of scheduled jobs such as scheduled backups, import/export jobs, and Integration Services packages. If any scheduled tasks require network or file system access, the SQLServerAgent service's credentials are typically used.

The SQLServerAgent service is dependent on the MSSQLServer service. During installation, the option is given to configure both services with the same credentials. Although this is by no means required, it is common practice. A frequent problem encountered by database administrators is that jobs that work perfectly when run manually fail when run by the agent. The reason for the failure is because the account that is used when testing the job manually is the logged-in administrator, but when the job is executed by the agent, the account the agent is running under does not have adequate permissions.

SQLBrowser (SQL Server Browser)

The SQLBrowser service is used by SQL Server for named instance name resolution and server name enumeration over TCP/IP and VIA networks.

The default instance of SQL Server is assigned the TCP port 1433 by default to support client communication. However, because more than one application cannot share a port assignment, any named instances are given a random port number when the service is started. This random port assignment makes it difficult for clients to connect to it, because the client applications don't know what port the server is listening on. To meet this need, the SQLBrowser service was created.

MSDTC (Distributed Transaction Coordinator)

The MSDTC service is used to manage transactions that span more than one instance of SQL Server or an instance of SQL Server and another transaction-based system. It utilizes a protocol known as Two-Phased Commit (2PC) to ensure that all transactions that span systems are committed on all participating systems.

SQL Server 2012 Database Objects

SQL Server 2012 database objects are defined and exist within a defined scope and hierarchy. This hierarchy enables more control over security permissions and organization of objects by similar function. SQL Server 2012 objects are defined at the Server, Database, and Schema levels.

Server

The server scope encompasses all the objects that exist on the instance of SQL Server, regardless of their respective database or namespace. The database object resides within the server scope.

We can install multiple instances of the SQL Server 2012 Data Platform application on a single computer running a Windows operating system.

Database

The database scope defines all the objects within a defined database catalog. Schemas exist in the database scope.

Schema

Each database can contain one or more schemas. A schema is a namespace for database objects. All data objects in a SQL Server 2012 database reside in a specific schema.

Object Names

Every object in a SQL Server 2012 database is identified by a four-part, fully qualified name. This fully qualified name takes the form of server.database.schema.object. However, when referring to objects, the fully qualified name can be abbreviated. By omitting the server name SQL Server will assume the instance the connection is currently connected to. Likewise, omitting the database name will cause SQL Server to assume the existing connection's database context.

SQL Server 2012 Databases

There are two types of databases in SQL Server: system databases and user databases. The *system databases* are used to store system-wide data and metadata. *User databases* are created by users who have the appropriate level of permissions to store application data.

System Databases

The system databases are comprised of Master, Model, MSDB, TempDB, and the hidden Resource database. If the server is configured to be a replication distributor, there will also be at least one system distribution database that is named during the replication configuration process.

The Master Database

The Master database is used to record all server-level objects in SQL Server 2012. The database ID of master is 1. This includes Server Logon accounts, Linked Server definitions, and EndPoints. The Master database also records information about all the other databases on the server (such as their file locations and names). Unlike its predecessors, SQL Server 2012 does not store system information in the Master database, but rather in the Resource database. However, system information is logically presented as the SYS schema in the Master database.

The Model Database

The Model database is a template database. The database ID of model is 3. Whenever a new database is created (including the system database TempDB), a copy of the Model database is created and renamed with the name of the database being created. The advantage of this behavior is that objects can be placed in the Model database prior to the creation of any new database and, when the database is created, the objects will appear in the new database.

The MSDB Database

I mostly think of the MSDB database as the SQL Server Agent's database. The database ID of msdb is 4. That's because the SQL Server Agent uses the MSDB database extensively for the storage of automated job definitions, job schedules, operator definitions, and alert definitions.

The TempDB Database

The TempDB database is used by SQL Server to store data temporarily. The database ID of temp is 2. The TempDB database is used extensively during SQL Server operations, so careful planning and evaluation of its size and placement are critical to ensure efficient SQL Server database operations.

The TempDB database is used by the Database Engine to store temporary objects (such as temporary tables, views, cursors, and table-valued variables) that are explicitly created by database programmers. In addition, the TempDB database is used by the SQL Server database engine to store work tables containing intermediate results of a query prior to a sort operation or other data manipulation.

The Resource Database

The last system database is the Resource database. The database ID of resource is 32767. The Resource database is a read-only database that contains all the system objects used by an instance of SQL Server. The Resource database is not accessible during normal database operations. It is logically presented as the SYS schema in every database. It contains no user data or metadata. Instead, it contains the structure and description of all system objects. This design enables the fast application of service packs by just replacing the existing Resource database with a new one. As an added bonus, to roll back a service pack installation, all you have to do is replace the new Resource database with the old one. This very elegant design

replaces the older method of running many scripts that progressively dropped and added new system objects.

User Databases

User databases are simply that: databases created by users. They are created to store data used by data applications and are the primary purpose of having a database server. You can create up to 32762 user databases in a single instance.

Distribution Databases

The distribution database stores metadata and transactional history to support all types of replication on a SQL Server. Typically, one distribution database is created when configuring a SQL Server as a replication Distributor. However, if needed, multiple distribution databases can be configured.

A model distribution database is installed by default and is used in the creation of a distribution database used in replication. It is installed in the same location as the rest of the system databases and is named distmdl.mdf.

SQL Server 2012 Enhancements

SQL Server 2012 is Microsoft's latest cloud-ready information platform. Organizations can use SQL Server 2012 to efficiently protect, unlock, and scale the power of their data across the desktop, mobile device, datacenter, and either a private or public cloud. Building on the success of the SQL Server 2008 R2 release, SQL Server 2012 has made a strong impact on organizations worldwide with its significant capabilities. It provides organizations with mission-critical performance and availability, as well as the potential to unlock breakthrough insights with pervasive data discovery across the organization. Finally, SQL Server 2012 delivers a variety of hybrid solutions you can choose from. For example, an organization can develop and deploy applications and database solutions on traditional nonvirtualized environments, on appliances, and in on-premises private clouds or off-premises public clouds. Moreover, these solutions can easily integrate with one another, offering a fully integrated hybrid solution.

Microsoft has made major investments in the SQL Server 2012 product as a whole; however, the new features and breakthrough capabilities that should interest database administrators (DBAs) are:

Availability Enhancements

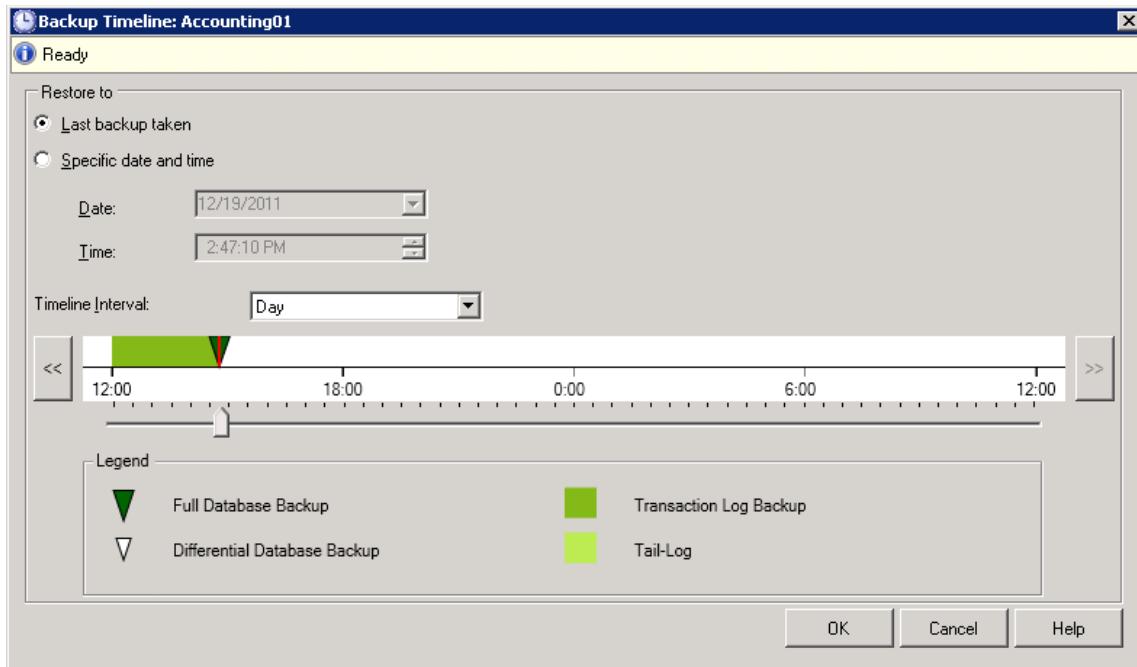
A tremendous amount of high-availability enhancements were added to SQL Server 2012, which is sure to increase both the confidence organizations have in their databases and the maximum uptime for those databases. SQL Server 2012 continues to deliver database mirroring, log shipping, and replication. However, it now also offers a new brand of technologies for achieving both high availability and disaster recovery known as AlwaysOn. Let's quickly review the new high-availability enhancement AlwaysOn:

AlwaysOn Availability Groups for DBAs, AlwaysOn Availability Groups is probably the most highly anticipated feature related to the Database Engine for DBAs. This new capability protects databases and allows for multiple databases to fail over as a single unit. Better data redundancy and protection is achieved because the solution supports up to four secondary replicas. Of these four secondary replicas, up to two secondaries can be configured as synchronous secondaries to ensure the copies are up to date. The secondary replicas can reside within a datacenter for achieving high availability within a site or across datacenters for disaster recovery. In addition, AlwaysOn Availability Groups provide a higher return on investment because hardware utilization is increased as the secondaries are active, readable, and can be leveraged to offload backups, reporting, and ad hoc queries from the primary replica. The solution is tightly integrated into SQL Server Management Studio, is straightforward to deploy, and supports either shared storage or local storage.

AlwaysOn Failover Cluster Instances (FCI) AlwaysOn Failover Cluster Instances provides superior instance-level protection using Windows Server Failover Clustering and shared storage. However, with SQL Server 2012 there are a tremendous number of enhancements to improve availability and reliability. First, FCI now provides support for multi-subnet failover clusters. These subnets, where the FCI nodes reside, can be located in the same datacenter or in geographically dispersed sites. Second, local storage can be leveraged for the TempDB database. Third, faster startup and recovery times are achieved after a failover transpires. Finally, improved cluster health-detection policies can be leveraged, offering a stronger and more flexible failover.

Support for Windows Server Core Installing SQL Server 2012 on Windows Server Core is now supported. Windows Server Core is a scaled-down edition of the Windows operating system and requires approximately 50 to 60 percent fewer reboots when patching servers. This translates to greater SQL Server uptime and increased security. Server Core deployment options using Windows Server 2008 R2 SP1 and higher are required. Chapter 2, "High-Availability and Disaster-Recovery Options," discusses deploying SQL Server 2012 on Server Core, including the features supported.

Recovery Advisor A new visual timeline has been introduced in SQL Server Management Studio to simplify the database restore process. As illustrated in Figure, the scroll bar beneath the timeline can be used to specify backups to restore a database to a point in time.



Scalability and Performance Enhancements

The SQL Server product group has made sizable investments in improving scalability and Performance associated with the SQL Server Database Engine. Some of the main enhancements that allow organizations to improve their SQL Server workloads include the following:

Columnstore Indexes More and more organizations have a requirement to deliver breakthrough and predictable performance on large data sets to stay competitive. SQL Server 2012 introduces a new in-memory, column store index built directly in the relational engine. Together with advanced query-processing enhancements, these technologies provide blazing-fast performance and improve queries associated with data warehouse workloads by 10 to 100 times. In some cases, customers have experienced a 400 percent improvement in performance.

Partition Support Increased To dramatically boost scalability and performance associated with large tables and data warehouses, SQL Server 2012 now supports up to 15,000 partitions per table by default. This is a significant increase from the previous version of SQL Server, which was limited to 1000 partitions by default. This new expanded support also helps enable large sliding-window scenarios for data warehouse maintenance.

Online Index Create, Rebuild, and Drop Many organizations running mission-critical workloads use online indexing to ensure their business environment does not experience downtime during routine index maintenance. With SQL Server 2012, indexes containing varchar(max), nvarchar(max), and varbinary(max) columns can now be created, rebuilt, and

dropped as an online operation. This is vital for organizations that require maximum uptime and concurrent user activity during index operations.

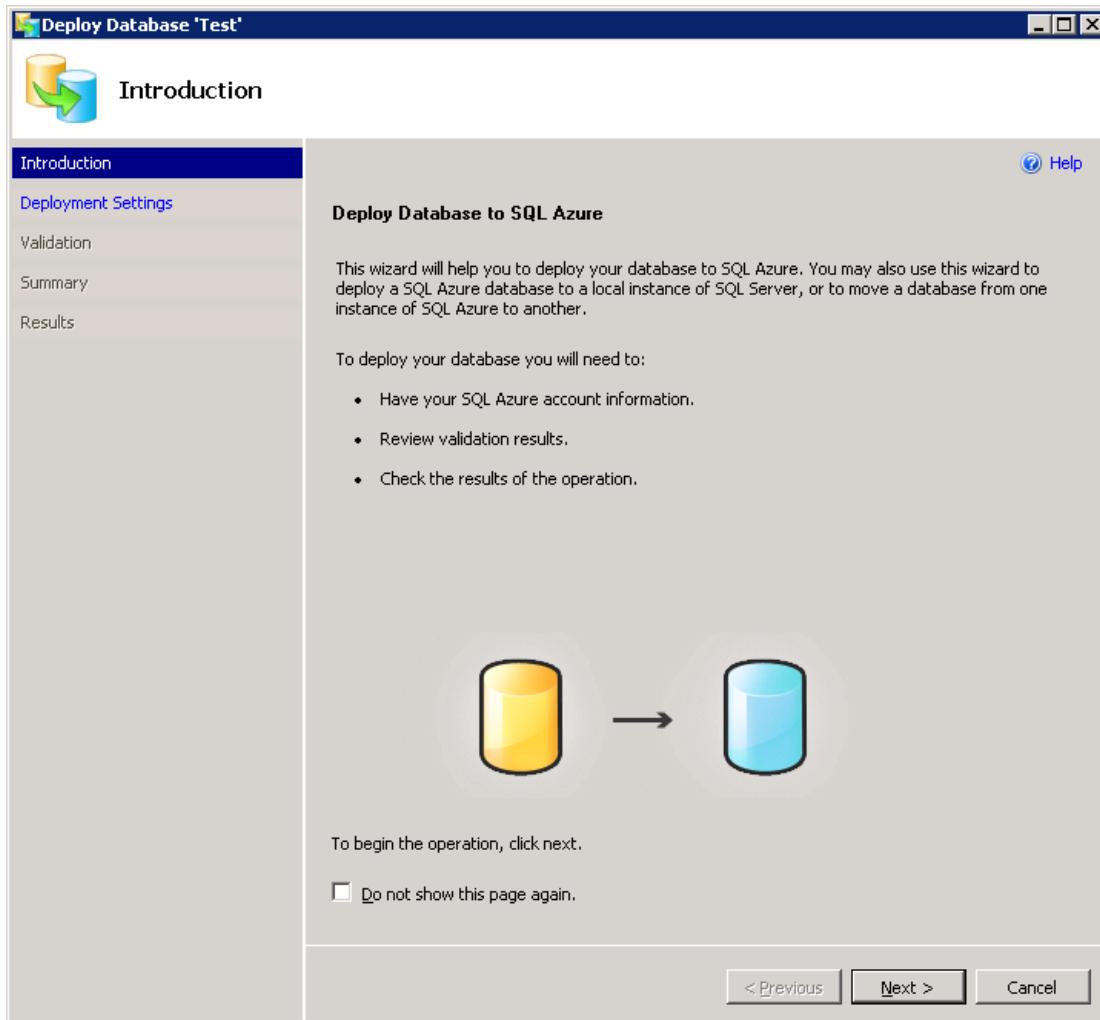
Achieve Maximum Scalability with Windows Server 2008 R2 Windows Server 2008 R2 is built to achieve unprecedented workload size, dynamic scalability, and across-the-board availability and reliability. As a result, SQL Server 2012 can achieve maximum scalability when running on Windows Server 2008 R2 because it supports up to 256 logical processors and 2 terabytes of memory in a single operating system instance.

Resource Governor Enhancements Many organizations currently leverage Resource Governor to gain predictable performance and improve their management of SQL Server workloads and resources by implementing limits on resource consumption based on incoming requests. In the past few years, customers have also been requesting additional improvements to the Resource Governor feature. Customers wanted to increase the maximum number of resource pools and support large-scale, multitenant database solutions with a higher level of isolation between workloads. They also wanted predictable chargeback and vertical isolation of machine resources.

The SQL Server product group responsible for the Resource Governor feature introduced new capabilities to address the requests of its customers and the SQL Server community. To begin, support for larger scale multitenancy can now be achieved on a single instance of SQL Server because the number of resource pools Resource Governor supports increased from 20 to 64. In addition, a maximum cap for CPU usage has been introduced to enable predictable chargeback and isolation on the CPU. Finally, resource pools can be affinitized to an individual schedule or a group of schedules for vertical isolation of machine resources. A new Dynamic Management View (DMV) called `sys.dm_resource_governor_resource_pool_affinity` improves database administrators' success in tracking resource pool affinity.

Contained Databases Authentication associated with database portability was a challenge in the previous versions of SQL Server. This was the result of users in a database being associated with logins on the source instance of SQL Server. If the database ever moved to another instance of SQL Server, the risk was that the login might not exist. With the introduction of contained databases in SQL Server 2012, users are authenticated directly into a user database without the dependency of logins in the Database Engine. This feature facilitates better portability of user databases among servers because contained databases have no external dependencies.

Tight Integration with SQL Azure A new Deploy Database To SQL Azure wizard, pictured in Figure, is integrated in the SQL Server Database Engine to help organizations deploy an on-premise database to SQL Azure. Furthermore, new scenarios can be enabled with SQL Azure Data Sync, which is a cloud service that provides bidirectional data synchronization between databases across the datacenter and cloud.



Startup Options Relocated within SQL Server Configuration Manager, a new Startup Parameters tab was introduced for better manageability of the parameters required for startup. A DBA can now easily specify startup parameters compared to previous versions of SQL Server, which at times was a tedious task. The Startup Parameters tab can be invoked by right-clicking a SQL Server instance name in SQL Server Configuration Manager and then selecting Properties.

Data-Tier Application (DAC) Enhancements SQL Server 2008 R2 introduced the concept of data-tier applications. A data-tier application is a single unit of deployment containing all of the database's schema, dependent objects, and deployment requirements used by an application. SQL Server 2012 introduces a few enhancements to DAC. With the new SQL Server, DAC upgrades are performed in an in-place fashion compared to the previous side-by-side upgrade process we've all grown accustomed to over the years. Moreover, DACs can be deployed, imported and exported more easily across premises and public cloud environments, such as

SQL Azure. Finally, data-tier applications now support many more objects compared to the previous SQL Server release.

Security Enhancements

Here is a snapshot of some of the new enterprise-ready security capabilities and controls that enable organizations to meet strict compliance policies and regulations:

- User-defined server roles for easier separation of duties
- Audit enhancements to improve compliance and resiliency
- Simplified security management, with a default schema for groups
- Contained Database Authentication, which provides database authentication that uses self-contained access information without the need for server logins
- SharePoint and Active Directory security models for higher data security in end-user reports

SQL Server 2012 Editions

There are three major editions of SQL Server 2012. There are additional, smaller editions typically not recommended for production environments so they are not covered here.

Enterprise Edition contains all the new SQL Server 2012 capabilities, including high availability and performance updates that make this a mission critical-ready database. This edition contains all the BI functionality as well.

Business Intelligence Edition is new in SQL Server 2012. The BI Server Edition offers the full suite of powerful BI capabilities in SQL Server 2012, including PowerPivot and Power View. One major focus of this edition is empowering end users with BI functionality. This is ideal for projects that need advanced BI capabilities but don't require the full OLTP performance and scalability of Enterprise Edition. The new BI Edition is inclusive of the standard edition and still contains the basic OLTP offerings.

Standard Edition remains as it is today, designed for departmental use with limited scale. It has basic database functionality and basic BI functionality. New features are now included in Standard such as compression.

Enterprise Edition

(Inclusive of Business Intelligence edition)

- Mission critical and Tier 1 applications
- Data Warehousing
- Private cloud and highly virtualized environments
- Large, centralized or external facing BI



Business Intelligence Edition

(Inclusive of Standard edition)

- Corporate and scalable reporting & analytics
- Power View and Power Pivot enable self-service analytics



Standard edition

- Departmental databases
- Limited business intelligence projects



Datacenter Edition is being retired — all capabilities now available in Enterprise Edition.

Workgroup & Small Business Editions are being retired.

Web Edition will only be available via SPLA.

SQL Server Capabilities	SQL Server Editions		
	Standard	Business Intelligence	Enterprise
Maximum Number of Cores	16 cores	16 cores – DB OS Max – BI	OS Max
Basic OLTP	●	●	●
Basic Reporting & Analytics	●	●	●
Programmability & Developer Tools	●	●	●
Manageability (Management studio, policy based management)	●	●	●
Enterprise Data Management (Data quality, master data services)		●	●
Self-service Business Intelligence (Power View, PowerPivot for SPS)		●	●
Corporate Business Intelligence (Semantic model, advanced analytics)		●	●
Advanced Security (Advanced auditing, transparent data encryption)			●
Data Warehousing (Column store index, compression, partitioning)			●
Maximum Scalability and Performance			●
High Availability	Limited	Basic	●
<u>AlwaysOn</u>			●
Virtualization Licensing	1 VM	1 VM	Unlimited w/ SA

Specialized Editions

Developer: The Developer edition includes all of the features and functionality found in the Enterprise edition; however, it is meant strictly for the purpose of development, testing, and demonstration. Note that you can transition a SQL Server Developer installation directly into production by upgrading it to SQL Server 2012 Enterprise without reinstallation.

Web: Available at a much more affordable price than the Enterprise and Standard editions, SQL Server 2012 Web is focused on service providers hosting Internet-facing web services environments. Unlike the Express edition, this edition doesn't have database size restrictions, it supports four processors, and supports up to 64 GB of memory. SQL Server 2012 Web does not offer the same premium features found in Enterprise and Standard editions, but it still remains the ideal platform for hosting websites and web applications.

Express: This free edition is the best entry-level alternative for independent software vendors, nonprofessional developers, and hobbyists building client applications. Individuals learning about databases or learning how to build client applications will find that this edition meets all their needs. This edition, in a nutshell, is limited to one processor and 1 GB of memory, and it can have a maximum database size of 10 GB. Also, Express is integrated with Microsoft Visual Studio.

Top 12 Features of SQL Server 2012

Microsoft has introduced SQL Server 2012 to the world and it's time for IT professionals to start to come to speed on what's new in this highly anticipated version of SQL Server.

1. AlwaysOn Availability Groups -- This feature takes database mirroring to a whole new level. With AlwaysOn, users will be able to fail over multiple databases in groups instead of individually. Also, secondary copies will be readable, and can be used for database backups. The big win is that your DR environment no longer needs to sit idle.

2. Windows Server Core Support -- If you don't know what Windows Server Core is, you may want to come up to speed before Windows 8 (MS is making a push back to the command line for server products). Core is the GUI-less version of Windows that uses DOS and PowerShell for user interaction. It has a much lower footprint (50% less memory and disk space utilization), requires fewer patches, and is more secure than the full install. Starting with SQL 2012, it is supported for SQL Server.

3. Columnstore Indexes -- This a cool new feature that is completely unique to SQL Server. They are special type of read-only index designed to be use with Data Warehouse queries. Basically, data is grouped and stored in a flat, compressed column index, greatly reducing I/O and memory utilization on large queries.

4. User-Defined Server Roles -- DBAs have always had the ability to create custom database role, but never server wide. For example, if the DBA wanted to give a development team read/write access to every database on a shared server, traditionally the only ways to do it

were either manually, or using undocumented procedures. Neither of which were good solutions. Now, the DBA can create a role, which has read/write access on every DB on the server, or any other custom server wide role.

5. Enhanced Auditing Features -- Audit is now available in all editions of SQL Server. Additionally, users can define custom audit specifications to write custom events into the audit log. New filtering features give greater flexibility in choosing which events to write to the log.

6. BI Semantic Model -- This is replacing the Analysis Services Unified Dimensional Model (or cubes most people referred to them). It's a hybrid model that allows one data model will support all BI experiences in SQL Server. Additionally, this will allow for some really neat text infographics

7. Sequence Objects -- For those folks who have worked with Oracle, this has been a long requested feature. A sequence is just an object that is a counter -- a good example of it's use would be to increment values in a table, based a trigger. SQL has always had similar functionality with identity columns, but now this is a discrete object.

8. Enhanced PowerShell Support -- Windows and SQL Server admins should definitely start brushing up on their PowerShell scripting skills. Microsoft is driving a lot of development effort into instrumenting all of their server-based products with PowerShell. SQL 2008 gave DBAs some exposure to it, but there are many more in cmdlets in SQL 2012.

9. Distributed Replay -- Once again this is answer to a feature that Oracle released (Real Application Testing). However, and in my opinion where the real value proposition of SQL Server is, in Oracle it is a (very expensive) cost option to Enterprise Edition. With SQL, when you buy your licenses for Enterprise Edition, you get everything. Distributed replay allows you to capture a workload on a production server, and replay it on another machine. This way changes in underlying schemas, support packs, or hardware changes can be tested under production conditions.

10. PowerView -- You may have heard of this under the name "Project Crescent" it is a fairly powerful self-service BI toolkit that allows users to create mash ups of BI reports from all over the Enterprise.

11. SQL Azure Enhancements -- These don't really go directly with the release of SQL 2012, but Microsoft is making some key enhancements to SQL Azure. Reporting Services for Azure will be available, along with backup to the Windows Azure data store, which is a huge enhancement. The maximum size of an Azure database is now up to 150G. Also Azure data sync allows a better hybrid model of cloud and on-premise solutions

12. Big Data Support -- I saved the biggest for last, introduced at the PASS (Professional Association for SQL Server) conference last year, Microsoft announced a partnership with

Hadoop provider Cloudera. One part of this involves MS releasing a ODBC driver for SQL Server that will run on a Linux platform. Additionally, Microsoft is building connectors for Hadoop, which is an extremely popular NoSQL platform. With this announcement, Microsoft has made a clear move into this very rapidly growing space.

SQL 2012 is a big step forward for Microsoft -- the company is positioning itself to be a leader in availability and in the growing area of big data. As a database professional, I look forward to using SQL 2012 to bring new solutions to my clients.

DBA Daily Activity Checklist

Database Administrators can sometimes have one of the most stressful jobs in the company. If you have been a DBA for long, you know the scenario. You have just sat in your chair with your cup of coffee, and your phone starts ringing off the hook. The voice on the other end states that they can't pull up their data or they are getting timeouts, or the system is running slow. Okay, time to dig in; it's going to be one of those days! Is it Friday yet?

In this case study, I will present ways to minimize those stressful days by having a pre-defined DBA checklist. A DBA checklist is a document of pre-defined administrative checks that are performed every morning to ensure that your server is at optimal performance. By having a standard list of items to check, you are more likely to catch and fix issues before there is a real problem.

The end result of the DBA checklist should have three sections. Section one contains the list of items that need checked. Section one should include checks from the following categories: performance, job failures, disk space, backups, connectivity, and anything specific to your environment, such as replication, mirroring, clustering, etc. Section two contains a place to write down issues and how they were resolved. The third section is a confirmation section where it is signed and dated. The third section is very important. Without this section, it is difficult to enforce and guarantee that these checks were performed.

The first step to create an effective checklist is to meet with all the DBAs and ask them these questions:

1. what do you check in the morning?
2. How do you check it?
3. What do you do when there is a problem?
4. Is there anyone you notify in the event of a failure?

In my experience, every DBA has his own mental checklist and different ways that he / she fix issues. It is important to get a list of the items written down in a document. By combining the ideas of every DBA, you will come up with a more thorough checklist, a standardized way to fix issues, and problems are less likely to fall through the cracks.

After the DBA checklist is created, completed checklists should be archived in a notebook to ensure that each check was performed every day. This also serves as a history of fixes for past issues, and an audit trail for the DBA.

Since every database environment is different, and every IS shop has its own tools, every DBA's checklist will be different. The end goal is to create a checklist that is customized to your environment, in which issues can be found and fixed quickly, so that you can avoid having one of those difficult days.

With this in mind, listed below is a sample checklist. Your checklist should be unique to your environment and should help find and fix issues as quickly as possible.

DBA Checklist

Backups

- Verify that the Network Backups are good by checking the backup emails. If a backup did not complete, contact _____ in the networking group, and send an email to the DBA group.
- Check the SQL Server backups. If a backup failed, research the cause of the failure and ensure that it is scheduled to run tonight.
- Check the database backup run duration of all production servers. Verify that the average time is within the normal range. Any significant increases in backup duration times need to be emailed to the networking group, requesting an explanation. The reason for this is that networking starts placing databases backups to tape at certain times, and if they put it to tape before the DBAs are done backing up, the tape copy will be bad.
- Verify that all databases were backed up. If any new databases were not backed up, create a backup maintenance plan for them and check the current schedule to determine a backup time.

Disk Space

Verify the free space on each drive of the servers. If there is significant variance in free space from the day before, research the cause of the free space fluctuation and resolve if necessary. Often times, log files will grow because of monthly jobs.

Job Failures

Check for failed jobs, by connecting to each SQL Server, selecting "job activity" and filtering on failed jobs. If a job failed, resolve the issue by contacting the owner of the job if necessary.

System Checks

- Check SQL logs on each server. In the event of a critical error, notify the DBA group and come to an agreement on how to resolve the problem. [SQL Server Error Logs]
- Check Application log on each server. In the event of a critical or unusual error, notify the DBA group and the networking group to determine what needs to be done to fix the error. [Event Viewer]

Performance

- Checking the server activities by using activity monitor and resolve any blockings occurred

- Checking the CPU as well memory utilization
- Checking the database consistency by using DBCC commands

Connectivity

- Log into the Customer application and verify that it can connect to the database and pull up data. Verify that it is performing at an acceptable speed. In the event of a failure, email the Customer Support Group, DBA group, and the DBA manager, before proceeding to resolve the issue.
- Log into the Billing application and verify that it can connect to the database and pull up data. Verify that it is performing at an acceptable speed. In the event of a failure, email the Billing Support Group, DBA group, and the DBA manager, before proceeding to resolve the issue.

Chapter-1

SQL Server 2012 Architecture

Components of the SQL Server Engine

Figure 1-1 shows the general architecture of SQL Server, which has four major components (three of whose subcomponents are listed): protocols, the relational engine (also called the Query Processor), the storage engine, and the SQLOS. Every batch submitted to SQL Server for execution, from any client application, must interact with these four components. (For simplicity, I've made some minor omissions and simplifications and ignored certain "helper" modules among the subcomponents.)

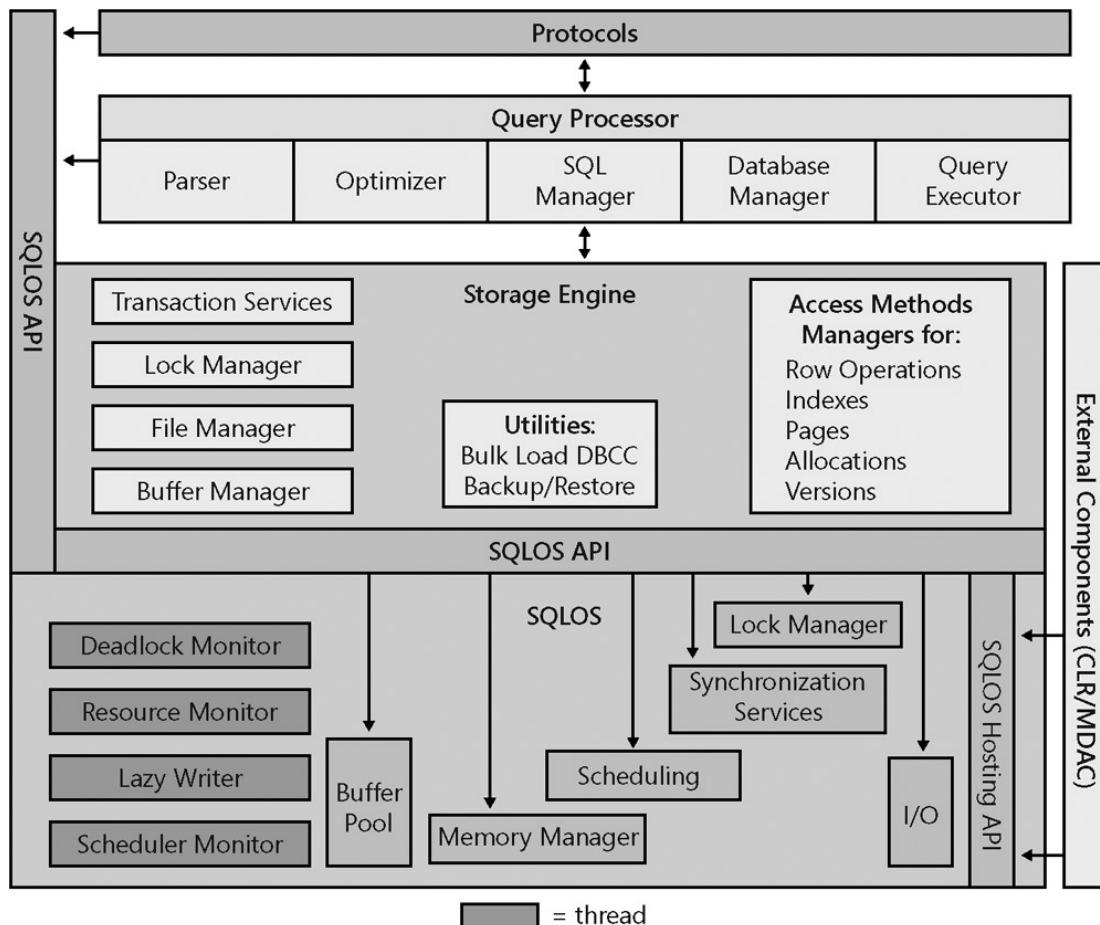


Figure 1-1: The major components of the SQL Server database engine

The protocol layer receives the request and translates it into a form that the relational engine can work with, and it also takes the final results of any queries, status messages, or error messages and translates them into a form the client can understand before sending them back

to the client. The relational engine layer accepts SQL batches and determines what to do with them. For Transact-SQL queries and programming constructs, it parses, compiles, and optimizes the request and oversees the process of executing the batch. As the batch is executed, if data is needed, a request for that data is passed to the storage engine. The storage engine manages all data access, both through transaction-based commands and bulk operations such as backup, bulk insert, and certain DBCC (Database Consistency Checker) commands. The SQLOS layer handles activities that are normally considered to be operating system responsibilities, such as thread management (scheduling), synchronization primitives, deadlock detection, and memory management, including the buffer pool.

Protocols

When an application communicates with the SQL Server Database Engine, the application programming interfaces (APIs) exposed by the protocol layer formats the communication using a Microsoft-defined format called a *tabular data stream (TDS) packet*. There are Net-Libraries on both the server and client computers that encapsulate the TDS packet inside a standard communication protocol, such as TCP/IP or Named Pipes. On the server side of the communication, the Net-Libraries are part of the Database Engine, and that protocol layer is illustrated in Figure 1-1. On the client side, the Net-Libraries are part of the SQL Native Client. The configuration of the client and the instance of SQL Server determine which protocol is used.

SQL Server can be configured to support multiple protocols simultaneously, coming from different clients. Each client connects to SQL Server with a single protocol. If the client program does not know which protocols SQL Server is listening on, you can configure the client to attempt multiple protocols sequentially. The following protocols are available:

- **Shared Memory** The simplest protocol to use, with no configurable settings. Clients using the Shared Memory protocol can connect only to a SQL Server instance running on the same computer, so this protocol is not useful for most database activity. Use this protocol for troubleshooting when you suspect that the other protocols are configured incorrectly. Clients using MDAC 2.8 or earlier cannot use the Shared Memory protocol. If such a connection is attempted, the client is switched to the Named Pipes protocol.
- **Named Pipes** A protocol developed for local area networks (LANs). A portion of memory is used by one process to pass information to another process, so that the output of one is the input of the other. The second process can be local (on the same computer as the first) or remote (on a networked computer).
- **TCP/IP** The most widely used protocol over the Internet. TCP/IP can communicate across interconnected networks of computers with diverse hardware architectures and operating systems. It includes standards for routing network traffic and offers advanced security features. Enabling SQL Server to use TCP/IP requires the most configuration effort, but most networked computers are already properly configured.

Tabular Data Stream Endpoints

From SQL Server 2005 onwards a new concept for defining SQL Server connections: the connection is represented on the server end by a TDS endpoint. During setup, SQL Server creates an endpoint for each of the four Net-Library protocols supported by SQL Server, and if the protocol is enabled, all users have access to it. For disabled protocols, the endpoint still

exists but cannot be used. An additional endpoint is created for the dedicated administrator connection (DAC), which can be used only by members of the sysadmin fixed server role. (I'll discuss the DAC in more detail in configuration chapter.)

The Relational Engine

As mentioned earlier, the relational engine is also called the query processor. It includes the components of SQL Server that determine exactly what your query needs to do and the best way to do it. By far the most complex component of the query processor, and maybe even of the entire SQL Server product, is the query optimizer, which determines the best execution plan for the queries in the batch.

The relational engine also manages the execution of queries as it requests data from the storage engine and processes the results returned. Communication between the relational engine and the storage engine is generally in terms of OLE DB row sets. (*Row set* is the OLE DB term for a *result set*.) The storage engine comprises the components needed to actually access and modify data on disk.

The Command Parser

The command parser handles Transact-SQL language events sent to SQL Server. It checks for proper syntax and translates Transact-SQL commands into an internal format that can be operated on. This internal format is known as a *query tree*. If the parser doesn't recognize the syntax, a syntax error is immediately raised that identifies where the error occurred. However, non-syntax error messages cannot be explicit about the exact source line that caused the error. Because only the command parser can access the source of the statement, the statement is no longer available in source format when the command is actually executed.

The Query Optimizer

The query optimizer takes the query tree from the command parser and prepares it for execution. Statements that can't be optimized, such as flow-of-control and DDL commands, are compiled into an internal form. The statements that are optimizable are marked as such and then passed to the optimizer. The optimizer is mainly concerned with the DML statement *SELECT, INSERT, UPDATE, and DELETE*, which can be processed in more than one way, and it is the optimizer's job to determine which of the many possible ways is the best. It compiles an entire command batch, optimizes queries that are optimizable, and checks security. The query optimization and compilation result in an execution plan.

The first step in producing such a plan is to *normalize* each query, which potentially breaks down a single query into multiple, fine-grained queries. After the optimizer normalizes a query, it *optimizes* it, which means it determines a plan for executing that query. Query optimization is cost based; the optimizer chooses the plan that it determines would cost the least based on internal metrics that include estimated memory requirements, CPU utilization, and number of required I/Os. The optimizer considers the type of statement requested, checks the amount of data in the various tables affected, looks at the indexes available for each table, and then looks at a sampling of the data values kept for each index or column referenced in the query. The sampling of the data values is called *distribution statistics*. Based on the available information, the optimizer considers the various access methods and processing strategies it could use to resolve a query and chooses the most cost-effective plan.

The Database Manager

The database manager handles access to the metadata needed for query compilation and optimization, making it clear that none of these separate modules can be run completely separately from the others. The metadata is stored as data and is managed by the storage engine, but metadata elements such as the data types of columns and the available indexes on a table must be available during the query compilation and optimization phase, before actual query execution starts.

The Query Executor

The query executor runs the execution plan that the optimizer produced, acting as a dispatcher for all the commands in the execution plan. This module steps through each command of the execution plan until the batch is complete. Most of the commands require interaction with the storage engine to modify or retrieve data and to manage transactions and locking.

The Storage Engine

The SQL Server storage engine has traditionally been considered to include all the components involved with the actual processing of data in your database. SQL Server 2005 separates out some of these components into a module called the SQLOS. In fact, the SQL Server storage engine team at Microsoft actually encompasses three areas: access methods, transaction management, and the SQLOS.

Transaction Services

A core feature of SQL Server is its ability to ensure that transactions are *atomic*—that is, all or nothing. In addition, transactions must be durable, which means that if a transaction has been committed, it must be recoverable by SQL Server no matter what—even if a total system failure occurs 1 millisecond after the commit was acknowledged. There are actually four properties that transactions must adhere to, called the ACID properties: atomicity, consistency, isolation, and durability.

Locking Operations Locking is a crucial function of a multi-user database system such as SQL Server, even if you are operating primarily in the snapshot isolation level with optimistic concurrency. SQL Server lets you manage multiple users simultaneously and ensures that the transactions observe the properties of the chosen isolation level. Even though readers will not block writers and writers will not block readers in snapshot isolation, writers do acquire locks and can still block other writers, and if two writers try to change the same data concurrently, a conflict will occur that must be resolved. The locking code acquires and releases various types of locks, such as share locks for reading, exclusive locks for writing, intent locks taken at a higher granularity to signal a potential “plan” to perform some operation, and extent locks for space allocation. It manages compatibility between the lock types, resolves deadlocks, and escalates locks if needed. The locking code controls table, page, and row locks as well as system data locks.

Chapter – 2

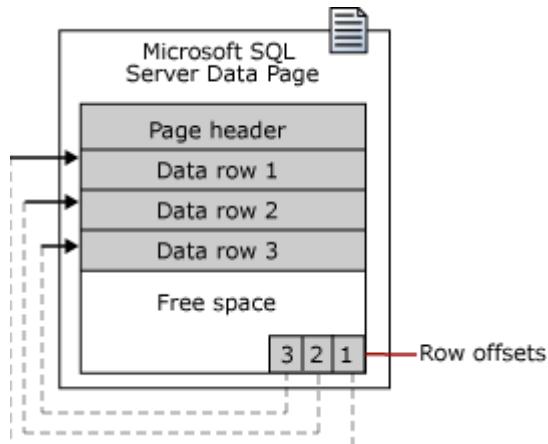
Pages and Extents

Pages

The fundamental unit of data storage in SQL Server is the page. The disk space allocated to a data file (.mdf or .ndf) in a database is logically divided into pages numbered contiguously from 0 to n .

Extents are a collection of eight physically contiguous pages and are used to efficiently manage the pages. All pages are stored in extents. In SQL Server, the page size is 8 KB. Each page begins with a 96-byte header that is used to store system information about the page. This information includes the page number, page type, the amount of free space on the page, and the allocation unit ID of the object that owns the page.

Data rows are put on the page serially, starting immediately after the header. A row offset table starts at the end of the page, and each row offset table contains one entry for each row on the page. Each entry records how far the first byte of the row is from the start of the page. The entries in the row offset table are in reverse sequence from the sequence of the rows on the page.



The maximum amount of data and overhead that is contained in a single row on a page is 8,060 bytes (8 KB).

Extents

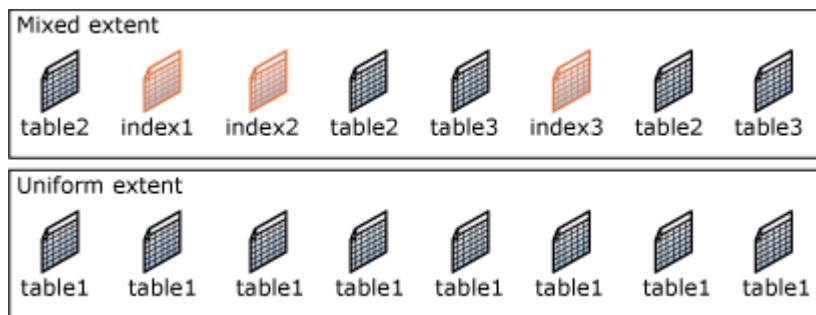
Extents are the basic unit in which space is managed. An extent is eight physically contiguous pages, or 64 KB. This means SQL Server databases have 16 extents per megabyte.

SQL Server has two types of extents:

- Uniform extents are owned by a single object; all eight pages in the extent can only be used by the owning object.

- Mixed extents are shared by up to eight objects. Each of the eight pages in the extent can be owned by a different object.

A new table or index is generally allocated pages from mixed extents. When the table or index grows to the point that it has eight pages, it then switches to use uniform extents for subsequent allocations. If you create an index on an existing table that has enough rows to generate eight pages in the index, all allocations to the index are in uniform extents.



Managing Extent Allocations

SQL Server uses two types of allocation maps to record the allocation of extents:

- Global Allocation Map (GAM)

GAM pages record what extents have been allocated. Each GAM covers 64,000 extents, or almost 4 GB of data. The GAM has one bit for each extent in the interval it covers. If the bit is 1, the extent is free; if the bit is 0, the extent is allocated.

- Shared Global Allocation Map (SGAM)

SGAM pages record which extents are currently being used as mixed extents and also have at least one unused page. Each SGAM covers 64,000 extents, or almost 4 GB of data. The SGAM has one bit for each extent in the interval it covers. If the bit is 1, the extent is being used as a mixed extent and has a free page. If the bit is 0, the extent is not used as a mixed extent, or it is a mixed extent and all its pages are being used.

Each extent has the following bit patterns set in the GAM and SGAM, based on its current use.

Current use of extent	GAM bit setting	SGAM bit setting
Free, not being used	1	1
Uniform extent, or full mixed extent	0	0
Mixed extent with free pages	0	1

Tracking Free Space

Page Free Space (PFS) pages record the allocation status of each page, whether an individual page has been allocated, and the amount of free space on each page. The PFS has one byte for each page, recording whether the page is allocated, and if so, whether it is empty, 1 to 50 percent full, 51 to 80 percent full, 81 to 95 percent full, or 96 to 100 percent full.

Chapter -3

Files and File groups

SQL Server maps a database over a set of operating-system files. Data and log information are never mixed in the same file, and individual files are used only by one database. File groups are named collections of files and are used to help with data placement and administrative tasks such as backup and restore operations

Database Files

SQL Server databases have three types of files:

- Primary data files

The primary data file is the starting point of the database and points to the other files in the database. Every database has one primary data file. The recommended file name extension for primary data files is .mdf.

- Secondary data files

Secondary data files make up all the data files, other than the primary data file. Some databases may not have any secondary data files, while others have several secondary data files. The recommended file name extension for secondary data files is .ndf.

- Log files

Log files hold all the log information that is used to recover the database. There must be at least one log file for each database, although there can be more than one. The recommended file name extension for log files is .ldf.

SQL Server does not enforce the .mdf, .ndf, and .ldf file name extensions, but these extensions help you identify the different kinds of files and their use.

Database File groups

Database objects and files can be grouped together in file groups for allocation and administration purposes. There are two types of file groups:

Primary

The primary file group contains the primary data file and any other files not specifically assigned to another file group. All pages for the system tables are allocated in the primary file group.

User-defined

User-defined file groups are any file groups that are specified by using the FILEGROUP keyword in a CREATE DATABASE or ALTER DATABASE statement.

Log files are never part of a file group. Log space is managed separately from data space.

Chapter-4

Thread and Task Architecture

Overview

Threads are an operating system feature that lets application logic be separated into several concurrent execution paths. This feature is useful when complex applications have many tasks that can be performed at the same time.

When an operating system executes an instance of an application, it creates a unit called a process to manage the instance. The process has a thread of execution. This is the series of programming instructions performed by the application code. For example, if a simple application has a single set of instructions that can be performed serially; there is just one execution path or thread through the application. More complex applications may have several tasks that can be performed in tandem, instead of serially. The application can do this by starting separate processes for each task. However, starting a process is a resource-intensive operation. Instead, an application can start separate threads. These are relatively less resource-intensive. Additionally, each thread can be scheduled for execution independently from the other threads associated with a process.

Threads allow complex applications to make more effective use of a CPU, even on computers that have a single CPU. With one CPU, only one thread can execute at a time. If one thread executes a long-running operation that does not use the CPU, such as a disk read or write, another one of the threads can execute until the first operation is completed. By being able to execute threads while other threads are waiting for an operation to be completed, an application can maximize its use of the CPU. This is especially true for multi-user, disk I/O intensive applications such as a database server. Computers that have multiple microprocessors or CPUs can execute one thread per CPU at the same time. For example, if a computer has eight CPUs, it can execute eight threads at the same time.

Allocating Threads to a CPU

By default, each instance of SQL Server starts each thread. The operating system then assigns each thread to a specific CPU. The operating system distributes threads from instances of SQL Server evenly among the microprocessors, or CPUs on a computer. Sometimes, the operating system can also move a thread from one CPU with heavy usage to another CPU.

SQL Server administrators can use the affinity mask configuration option to exclude one or more CPUs from being eligible to run threads from a specific instance of SQL Server. The affinity mask value specifies a bit pattern that indicates the CPUs that are used to run threads from that instance of SQL Server. For example, the affinity mask value 13 represents the bit pattern 1101. On a computer that has four CPUs, this indicates that threads from that instance of SQL Server can be scheduled on CPUs 0, 2, and 3, but not on CPU 1. If affinity mask is specified, the instance of SQL Server allocates threads evenly among the CPUs that have not been masked off. Another effect of affinity mask is that the operating system does not move threads from one CPU to another. However, affinity mask is rarely used. Most systems obtain optimal performance by letting the operating system schedule the threads among the available CPUs.

Affinity Mask

The affinity mask configuration option restricts a SQL Server instance to running on a subset of the processors. If SQL Server 2012 runs on a dedicated server, allowing SQL Server to use all processors can ensure best performance. In a server consolidation or multiple-instance environment, for more predictable performance, SQL Server may be configured on dedicated hardware resources that affinize processors by SQL Server instance.

SQL Server Processor Affinity Mask

SQL Server's mechanism for scheduling work requests is handled through a data structure concept called a *scheduler*. The scheduler is created for each processor assigned to SQL Server through the affinity mask configuration setting at startup. *Worker threads* (a subset of the max worker threads configuration setting) are dynamically created during a batch request and are evenly distributed between each CPU node and load-balanced across its schedulers.

Affinity I/O Mask

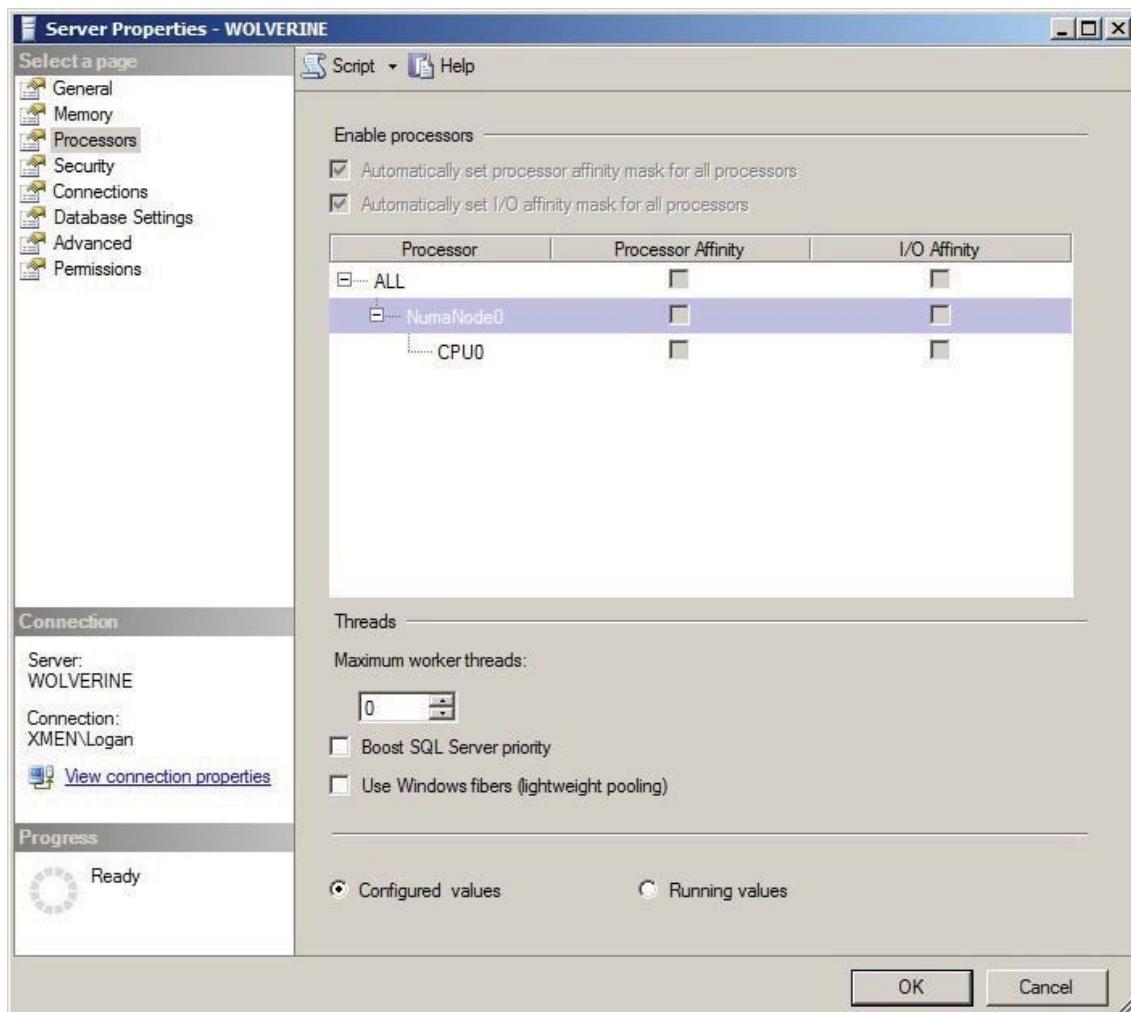
The affinity I/O mask feature, as shown in Figure 11-11, was introduced with SP1 of SQL Server 2000. This option defines the specific processors on which SQL Server I/O threads can execute. The affinity I/O mask option has a default setting of 0, indicating that SQL Server threads are allowed to execute on all processors. The performance gains associated with enabling the affinity I/O mask feature are achieved by grouping the SQL threads that perform all I/O tasks (no-data cache retrievals — specifically, physical I/O requests) on dedicated resources. This keeps I/O processing and related data in the same cache systems, maximizing data locality and minimizing unnecessary bus traffic.

When using affinity masks to assign processor resources to the operating system, either SQL Server processes (non-I/O) or SQL Server I/O processes, you must be careful not to assign multiple functions to any individual processor.

You should consider SQL I/O affinity when there is high privileged time on the processors that is not affinized to SQL Server. For example, consider a 32-processor system running under load with 30 of the 32 processors affinized to SQL Server, leaving two processors to the Windows OS and other non-SQL activities. If the Processor: % Privileged time is high (greater than 15 percent), SQL I/O affinity can be configured to help reduce the privileged-time overhead in the processors assigned to SQL Server.

The following steps outline the SQL I/O affinity procedure for determining what values to set:

1. Add I/O capability until there is no I/O bottleneck (disk queue length has been eliminated) and all unnecessary processes have been stopped.
2. Add a processor designated to SQL I/O affinity.
3. Measure the CPU utilization of these processors under heavy load.
4. Increase the number of processors until the designated processors are no longer peaked. For a non-SMP system, select processors that are in the same cell node.



SQL Server Lightweight Pooling

Context switching can often become problematic. In most environments, context switching should be less than 1,000 per second per processor. The SQL Server lightweight pooling option provides relief for this by enabling tasks to use NT "fibers," rather than threads, as workers.

A *fiber* is an executable unit that is lighter than a thread and operates in the context of user mode. When lightweight pooling is selected, each scheduler uses a single thread to control the scheduling of work requests by multiple fibers. A fiber can be viewed as a "lightweight thread," which under certain circumstances takes less overhead than standard worker threads to context switch. The number of fibers is controlled by the Max Worker Threads configuration setting.

Max Degree of Parallelism (MAXDOP)

By default, the MAXDOP value is set to 0, which enables SQL Server to consider all processors when creating an execution plan. In most systems, a MAXDOP setting equivalent to the number of cores (to a maximum of 8) is recommended. This limits the overhead introduced by parallelization. In some systems, based on application-workload profiles, it is even recommended that you set this value to 1 (including for SAP and Siebel). This can prevent the query optimizer from choosing parallel query plans. Using multiple processors to run a single query is not always desirable in an OLTP environment, although it is desirable in a data warehousing environment.

Hot Add CPU

Hot add CPU is the ability to dynamically add CPUs to a running system. Adding CPUs can occur physically by adding new hardware, logically by online hardware partitioning, or virtually through a virtualization layer. Starting with SQL Server 2008, SQL Server supports hot add CPU.

Requirements for hot add CPU:

- Requires hardware that supports hot add CPU.
- Requires the 64-bit edition of Windows Server 2008 Datacenter or the Windows Server 2008 Enterprise Edition for Itanium-Based Systems operating system.
- Requires SQL Server Enterprise.

SQL Server does not automatically start to use CPUs after they are added. This prevents SQL Server from using CPUs that might be added for some other purpose. After adding CPUs, execute the RECONFIGURE statement, so that SQL Server will recognize the new CPUs as available resources.

Chapter-5

Memory Architecture

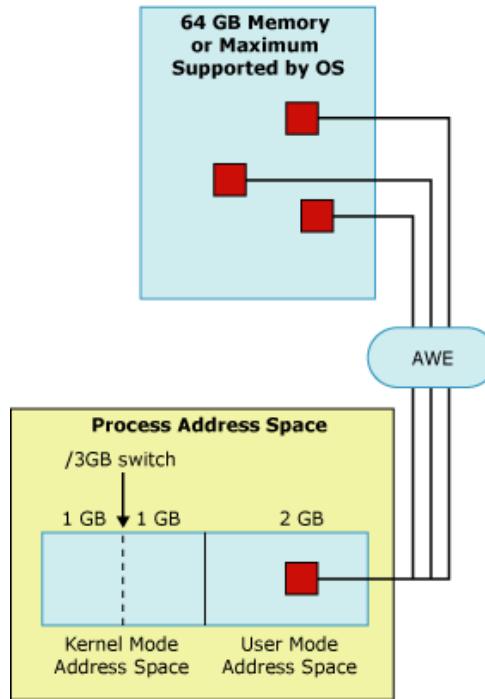
32-bit Vs 64-bit Architecture

A 32-bit machine can directly address only 4 GB of memory, and by default, Windows itself reserves the top 2 GB of address space for its own use, which leaves only 2 GB as the maximum size of the VAS for any application, such as SQL Server. You can increase this by enabling a */3GB* flag in the system's Boot.ini file, which allows applications to have a VAS of up to 3 GB. If your system has more than 3GB of RAM, the only way a 32-bit machine can get to it is by enabling AWE. One benefit in SQL Server 2005 of using AWE, is that memory pages allocated through the AWE mechanism are considered locked pages and can never be swapped out.

On a 64-bit platform, the AWE Enabled configuration option is present, but its setting is ignored. However, the Windows policy Lock Pages in Memory option is available, although it is disabled by default. This policy determines which accounts can make use of a Windows feature to keep data in physical memory, preventing the system from paging the data to virtual memory on disk. It is recommended that you enable this policy on a 64-bit system.

On 32-bit operating systems, you will have to enable Lock Pages in Memory policy when using AWE. It is recommended that you don't enable the Lock Pages in Memory policy if you are not using AWE. Although SQL Server will ignore this option when AWE is not enabled, other processes on the system may be impacted.

All 32-bit applications have a 4-gigabyte (GB) process address space (32-bit addresses can map a maximum of 4 GB of memory). Microsoft Windows operating systems provide applications with access to 2 GB of process address space, specifically known as user mode virtual address space. All threads owned by an application share the same user mode virtual address space. The remaining 2 GB are reserved for the operating system (also known as kernel mode address space). All operating system editions starting with Windows 2000 Server, including Windows Server 2003, have a boot.ini switch that can provide applications with access to 3 GB of process address space, limiting the kernel mode address space to 1 GB.



Address Windowing Extensions (AWE) extend the capabilities of 32-bit applications by allowing access to as much physical memory as the operating system supports. AWE accomplishes this by mapping a subset of up to 64 GB into the user address space. Mapping between the application buffer pool and AWE-mapped memory is handled through manipulation of the Windows virtual memory tables.

Dynamic Memory Management

The default memory management behavior of the Microsoft SQL Server Database Engine is to acquire as much memory as it needs without creating a memory shortage on the system. The Database Engine does this by using the Memory Notification APIs in Microsoft Windows.

Virtual address space of SQL Server can be divided into two distinct regions: space occupied by the buffer pool and the rest. If AWE mechanism is enabled, the buffer pool may reside in AWE mapped memory, providing additional space for database pages.

The buffer pool serves as a primary memory allocation source of SQL Server. External components that reside inside SQL Server process, such as COM objects, and not aware of the SQL Server memory management facilities, use memory outside of the virtual address space occupied by the buffer pool.

When SQL Server starts, it computes the size of virtual address space for the buffer pool based on a number of parameters such as amount of physical memory on the system, number of server threads and various startup parameters. SQL Server reserves the computed amount of its process virtual address space for the buffer pool, but it acquires (commits) only the required amount of physical memory for the current load.

The instance then continues to acquire memory as needed to support the workload. As more users connect and run queries, SQL Server acquires the additional physical memory on demand. A SQL Server instance continues to acquire physical memory until it either reaches its max server memory allocation target or Windows indicates there is no longer an excess of free memory; it frees memory when it has more than the min server memory setting, and Windows indicates that there is a shortage of free memory.

As other applications are started on a computer running an instance of SQL Server, they consume memory and the amount of free physical memory drops below the SQL Server target. The instance of SQL Server adjusts its memory consumption. If another application is stopped and more memory becomes available, the instance of SQL Server increases the size of its memory allocation. SQL Server can free and acquire several megabytes of memory each second, allowing it to quickly adjust to memory allocation changes.

Max Server Memory

When max server memory is kept at the default dynamic setting, SQL Server acquires and frees memory in response to internal and external pressure. SQL Server uses all available memory if left unchecked, so the max server memory setting is strongly recommended. The Windows OS needs some memory to function, so a value of 8GB to 16GB less than the total system memory should be configured. Please refer to the following table for configuration guidelines:

TOTAL SYSTEM MEMORY (GB)	OS RESERVED MEMORY (GB)	MAX SQL SERVER MEMORY (GB)
16	4	12
32	6	26
64	8	56
128	16	112
256	16	240

Effects of min and max server memory

The min server memory and max server memory configuration options establish upper and lower limits to the amount of memory used by the buffer pool of the Microsoft SQL Server Database Engine. The buffer pool does not immediately acquire the amount of memory specified in min server memory. The buffer pool starts with only the memory required to initialize. As the Database Engine workload increases, it keeps acquiring the memory required to support the workload. The buffer pool does not free any of the acquired memory until it reaches the amount specified in min server memory. Once min server memory is reached, the buffer pool then uses the standard algorithm to acquire and free memory as needed. The only difference is that the buffer pool never drops its memory allocation below the level specified in min server memory, and never acquires more memory than the level specified in max server memory.

The amount of memory acquired by the Database Engine is entirely dependent on the workload placed on the instance. A SQL Server instance that is not processing many requests may never reach min server memory.

If the same value is specified for both min server memory and max server memory, then once the memory allocated to the Database Engine reaches that value, the Database Engine stops dynamically freeing and acquiring memory for the buffer pool.

If an instance of SQL Server is running on a computer where other applications are frequently stopped or started, the allocation and deallocation of memory by the instance of SQL Server may slow the startup times of other applications. Also, if SQL Server is one of several server applications running on a single computer, the system administrators may need to control the amount of memory allocated to SQL Server. In these cases, you can use the min server memory and max server memory options to control how much memory SQL Server can use.

SQL Server supports Address Windowing Extensions (AWE) allowing use of physical memory over 4 gigabytes (GB) on 32-bit versions of Microsoft Windows operating systems. Up to 64 GB of physical memory is supported. Instances of SQL Server that are running on Microsoft Windows 2000 use static AWE memory allocation, and instances that are running on Microsoft Windows Server 2003 use dynamic AWE memory allocation.

Index Creation Memory Option

The index creation memory setting, as shown in Figure 11-16, determines how much memory can be used by SQL Server for sort operations during the index-creation process. The default value of 0 enables SQL Server to automatically determine the ideal value. In conditions in which index creation is performed on large tables, pre-allocating space in memory enables a faster index creation process.

Using AWE

Microsoft SQL Server uses the Microsoft Windows Address Windowing Extensions (AWE) API to support very large amounts of physical memory. SQL Server can access up to 64 gigabytes (GB) of memory on Microsoft Windows 2000 Server and Microsoft Windows Server 2003.

AWE is a set of extensions to the memory management functions of Windows that allow applications to address more memory than the 2-3 GB that is available through standard 32-bit addressing. AWE lets applications acquire physical memory, and then dynamically map views of the nonpaged memory to the 32-bit address space. Although the 32-bit address space is limited to 4 GB, the nonpaged memory can be much larger. This enables memory-intensive applications, such as large database systems, to address more memory than can be supported in a 32-bit address space.

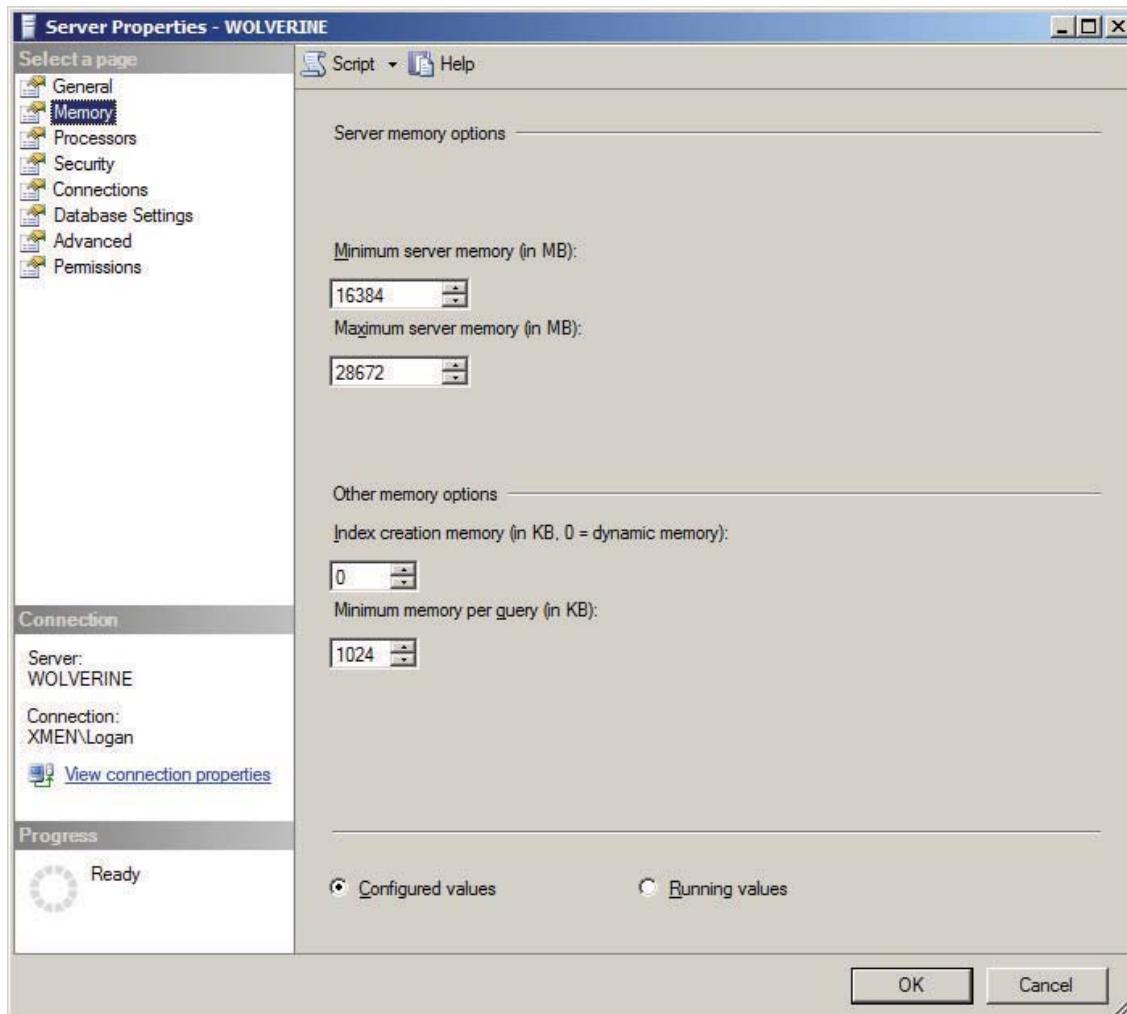
Before you configure the operating system for AWE, consider the following:

- AWE allows allocating physical memory over 4 GB on 32-bit architecture. AWE should be used only when available physical memory is greater than user-mode virtual address space.
- To support more than 4 GB of physical memory on 32-bit operating systems, you must add the /pae parameter to the Boot.ini file and reboot the computer. For more information, see your Windows documentation.

Memory management is a huge topic, and to cover every detail would require a whole volume in itself. My goal in this section is twofold: first, to provide enough information about

how SQL Server uses its memory resources so you can determine whether memory is being managed well on your system; and second, to describe the aspects of memory management that you have control over so you can understand when to exert that control.

By default, SQL Server 2008 manages its memory resources almost completely dynamically. When allocating memory, SQL Server must communicate constantly with the operating system, which is one of the reasons the SQLOS layer of the engine is so important.



Minimum Memory per Query

You can use the Minimum Memory per Query option to improve the performance of queries that use hashing or sorting operations. SQL Server automatically allocates the minimum amount of memory set in this configuration setting. The default Minimum Memory per Query option setting is equal to 1024KB (refer to Figure). It is important to ensure that the SQL Server

environment has the minimum amount of query memory available. However, in an environment with high query-execution concurrency, if this setting is configured too high, SQL Server waits for a memory allocation to meet the minimum memory level before executing a query.

MEMORY CONSIDERATIONS AND ENHANCEMENTS

Because memory is fast relative to disk I/O, using this system resource effectively can have a large impact on the system's overall ability to scale and perform well. SQL Server 2012 memory architecture and capabilities vary greatly from those of previous versions of SQL Servers. Changes include the ability to consume and release memory-based, internal-server conditions dynamically using the AWE mechanism. In SQL Server 2000, all memory allocations above 4GB were static. Additional memory enhancements include the introduction of hierarchical memory architecture to maximize data locality and improve scalability by removing a centralized memory manager. SQL Server 2012 has resource monitoring, dynamic management views (DMVs), and a common caching framework. All these concepts are discussed throughout this section.

Buffer Management

A buffer is an 8-KB page in memory, the same size as a data or index page. Thus, the buffer cache is divided into 8-KB pages. The buffer manager manages the functions for reading data or index pages from the database disk files into the buffer cache and writing modified pages back to disk. A page remains in the buffer cache until the buffer manager needs the buffer area to read in more data. Data is written back to disk only if it is modified. Data in the buffer cache can be modified multiple times before being written back to disk.

The Buffer Pool and the Data Cache

The main memory component in SQL Server is the buffer pool. All memory not used by another memory component remains in the buffer pool to be used as a data cache for pages read in from the database files on disk. The buffer manager manages disk I/O functions for bringing data and index pages into the data cache so data can be shared among users. When other components require memory, they can request a buffer from the buffer pool. A buffer is a page in memory that's the same size as a data or index page. You can think of it as a page frame that can hold one page from a database. Most of the buffers taken from the buffer pool for other memory components go to other kinds of memory caches, the largest of which is typically the cache for procedure and query plans, which are usually called the *procedure cache*.

Occasionally, SQL Server must request contiguous memory in larger blocks than the 8-KB pages that the buffer pool can provide so memory must be allocated from outside the buffer pool. Use of large memory blocks is typically kept to minimum, so direct calls to the operating system account for a small fraction of SQL Server memory usage.

Difference between Checkpoint and LazyWriter

CheckPoint	Lazy Writer
1. Flush dirty pages to Disk	1. Flush dirty pages to disk.
2. Flush only Data pages to disk	2. Check for available memory and removed Buffer pool (execution plan/compile plan/ Data pages /Memory objects)
3. Default, Occurs approximately every 1 minute	3. Occurs depending upon memory pressure and resource availability
4. Can be managed with sp_configure -recovery interval option	4. It is lazy, Sql server manages by its own.
5. Does not check the memory pressure	5. Monitor the memory pressure and try maintain the available free memory.
6. crash recovery process will be fast to read log as data file is updated.	6. No role in recovery
7. Occurs for any DDL statement	7. Occurs per requirement
8. Occurs before Backup/Detach command	8. Occurs per requirement
9. Depends upon the configuration setting, we can control.	9. Works on Least recent used pages and removed unused plans first, no user control.
10. for simple recovery it flush the tlog file after 70% full.	10. No effect on recovery model.
11. can manually /Forcefully run command "Checkpoint"	11.No command for Lazy Writer
12. Very Less performance impact	12. No performance impact

Chapter -6

Installing SQL Server

The installation process of SQL Server 2012 can be as easy as executing the Setup Wizard from the installation media and following the prompts in each of the setup screens. The Setup Wizard makes several important decisions for you during this process. Throughout this chapter you learn about those decisions and the appropriate configurations for a secure, stable, and scalable installation.

Installation Procedure for New SQL Server 2008 R2 Instances on Windows 2008 R2

The following are some recommendations before installing SQL Server 2008 R2:

- Use NTFS file system
- Don't try to install SQL Server 2008 R2 on a compressed drive, because setup will block the installation
- Do not install SQL Server on a Domain Controller
- Configure your firewall to allow SQL Server access
- The user account that is running SQL Server Setup must have administrative privileges on the computer
- Verify Windows Management Instrumentation service (Control Panel -> Administrative Tools -> Services) is running

Run SQL Server 2008 R2 setup. First, setup checks to determine software requirements are installed. If not, you will be prompted to install.



As mentioned above, SQL Server 2008 R2 requires Windows Installer 4.5 and .Net Framework 3.5 SP1 to be installed. Click OK button to install .Net Framework.

Setup



Setup is loading installation components.
This may take a minute or two.



Microsoft .NET Framework 3.5 SP1 Setup

Welcome to Setup Microsoft .net Framework

Be sure to carefully read and understand all the rights and restrictions described in the license terms. You must accept the license terms before you can install the software.

MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS

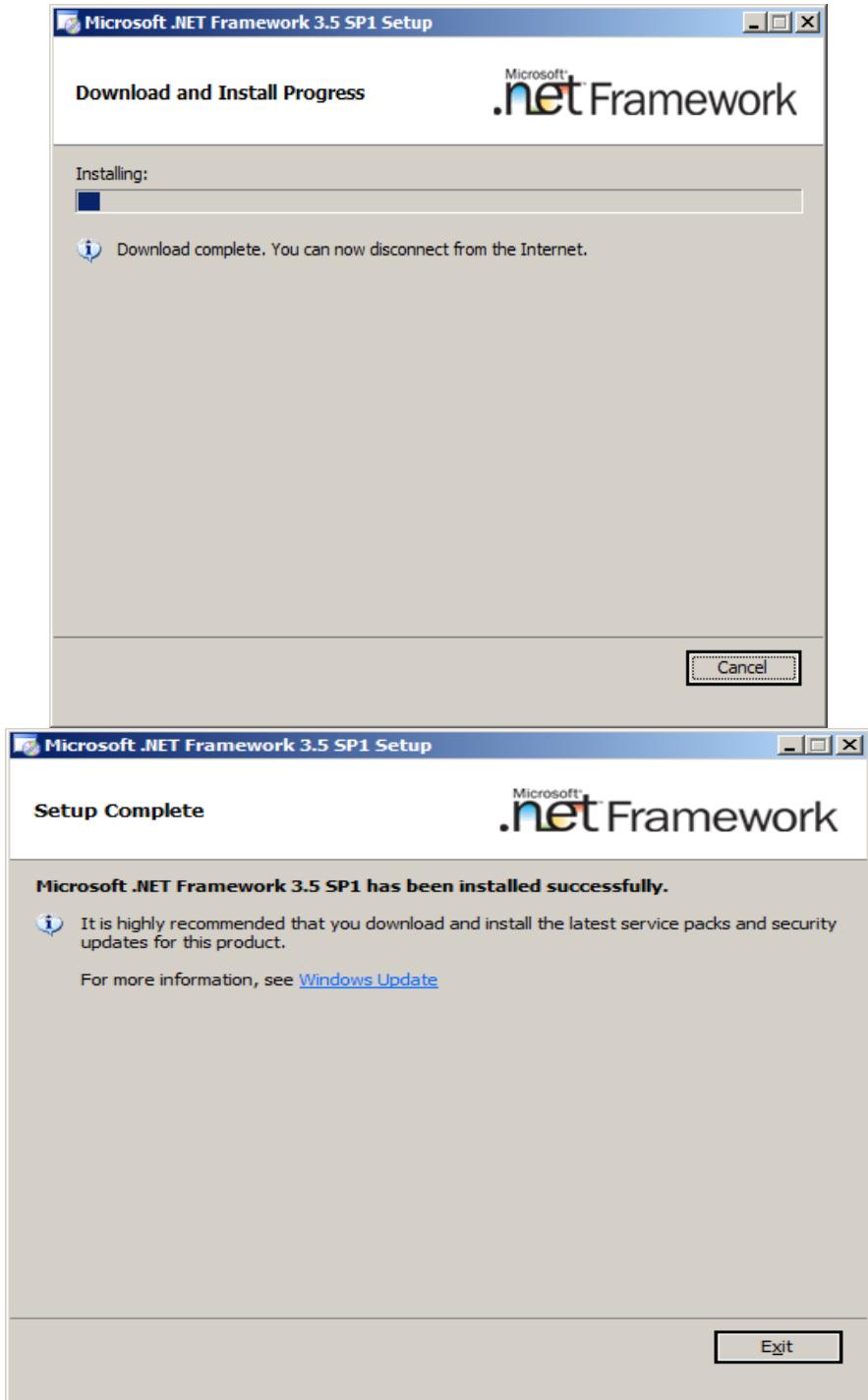
Press the Page Down key to see more text. [Print](#)

I have read and [ACCEPT](#) the terms of the License Agreement
 I DO NOT [ACCEPT](#) the terms of the License Agreement

[Send information about my setup experiences to Microsoft Corporation.](#)
Details regarding the [data collection policy](#)

Download File Size: 39 MB
Download Time Estimate: 1 hr 35 min (56 kbps)
10 min (512 kbps)

[Install >](#) [Cancel](#)

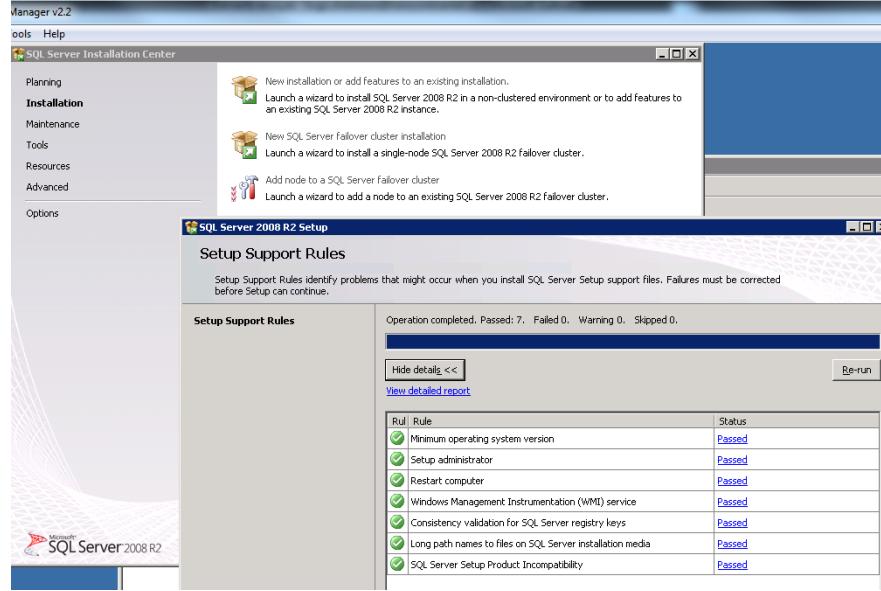


1. Click on Installation, then 'New SQL Server stand-alone installation...' (top)



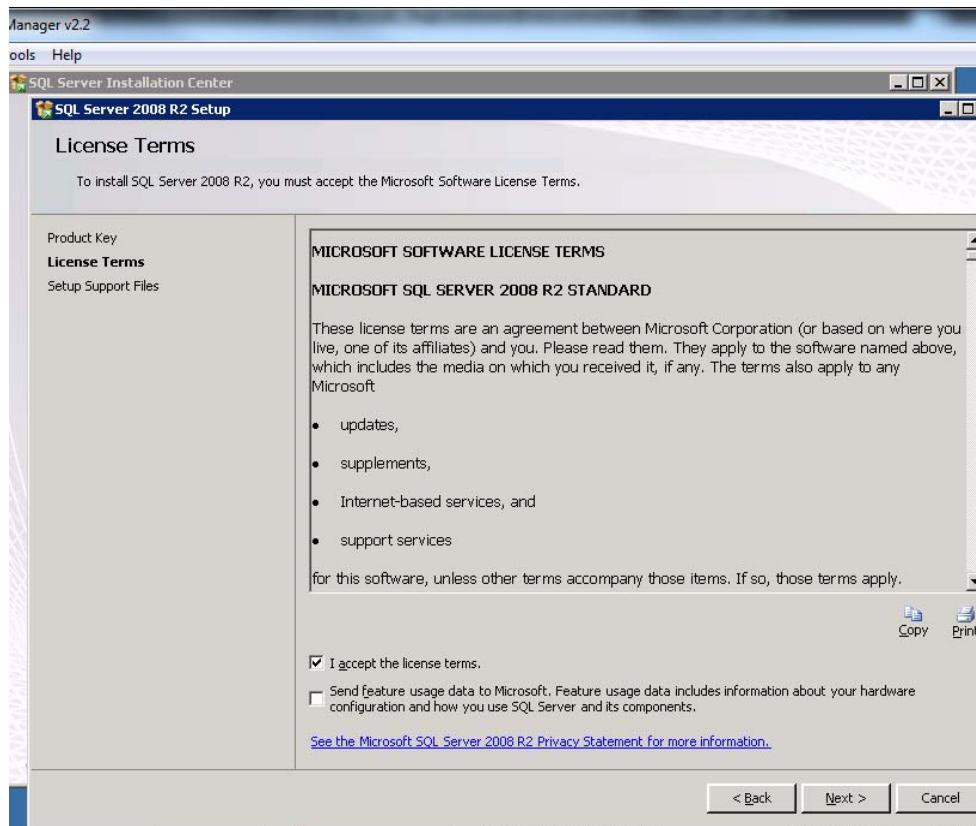
If you are adding features to an existing one, click on the top link also.

2. Let the setup rules check run:

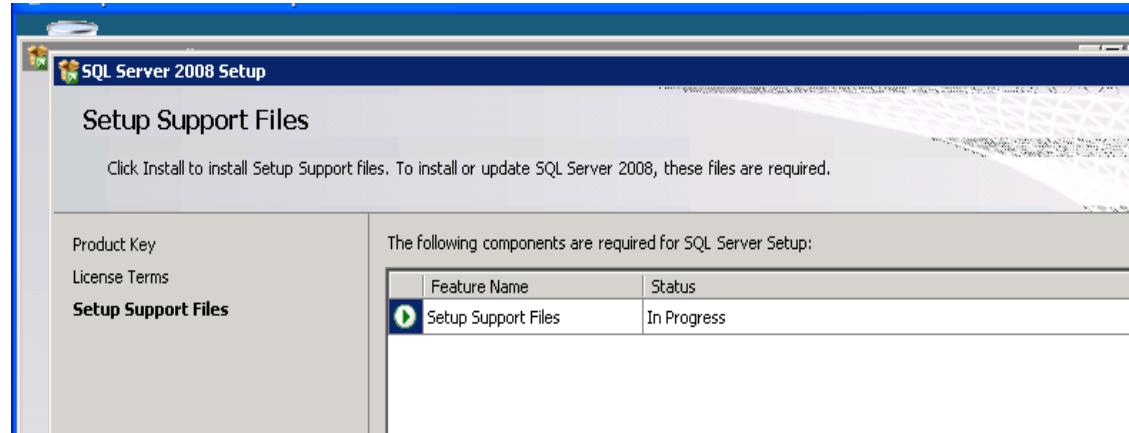


Enter key or make sure there is key automatically taken from the ISO, or enter it manually.

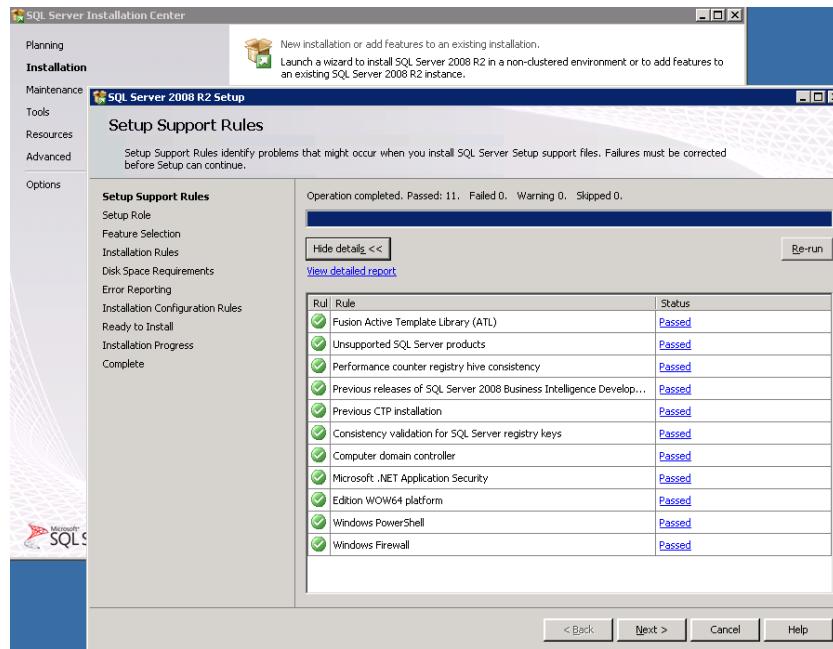
Accept the terms and conditions



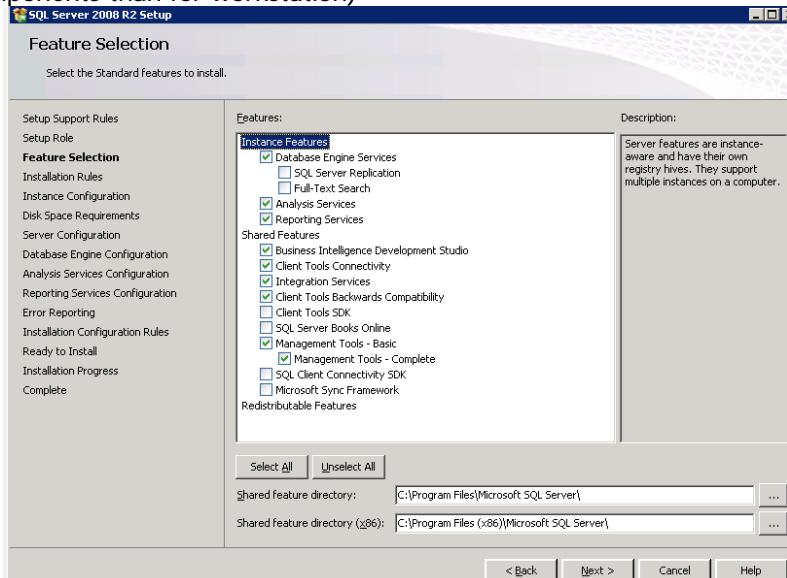
3. Click install



4. Make sure the pre-install checklist provides a Passed Status for each rule.
then click Next.

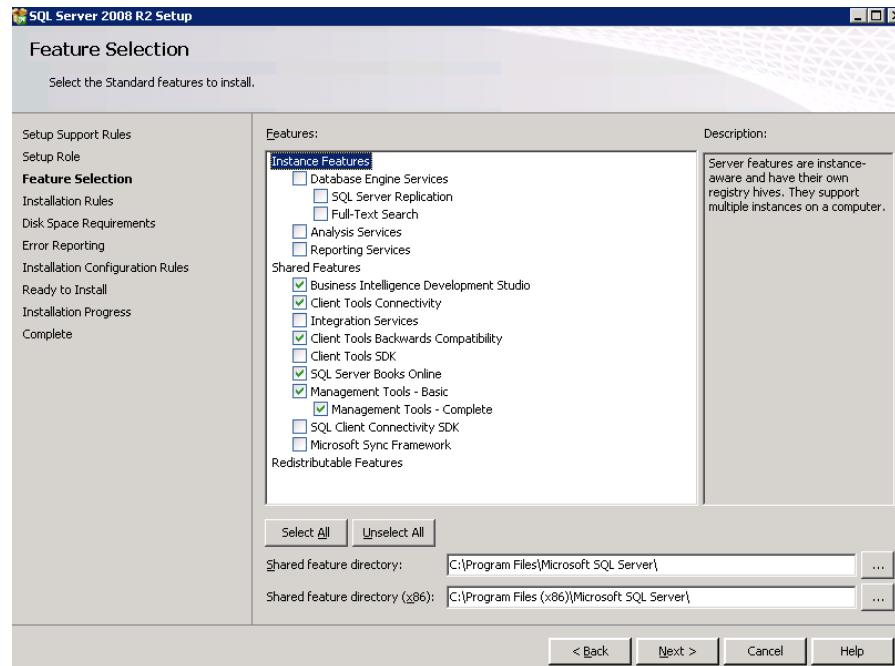


5. **DB Server** - Do not choose default, continue with Feature Selection for a **Server** install (more components than for workstation)

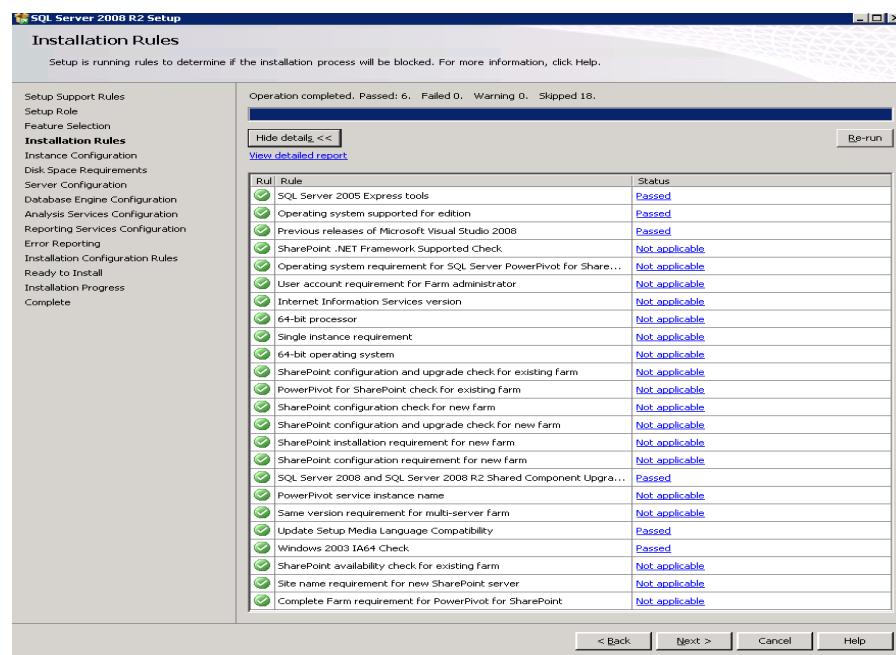


In the case of the server above, there is a 40GB C drive, so the Shared Features Directories are placed on C drive. If there is a Apps Drive available (e.g. D) and C is small, choose the drive allocated for program files (because the system databases will be placed on this drive by default).

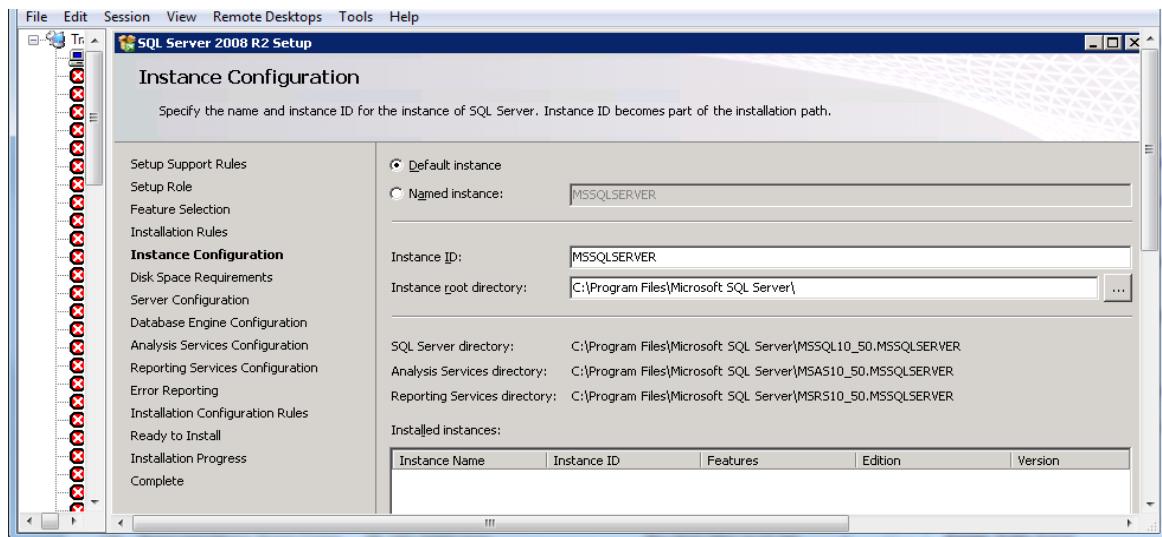
Note that if this is for a client installation, do not select ANY of the INSTANCE Features, just Shared Features should be selected, as follows (or choose more if you need, I recommend this as a minimum):



6. A new installation step added in 2008 R2 is this Installation Rule Confirmation:



Choose Default instance (so that the machine is accessed directly at ServerName), or specify instance name (as in ServerName\InstanceName) if it is required and several are on the same machine.

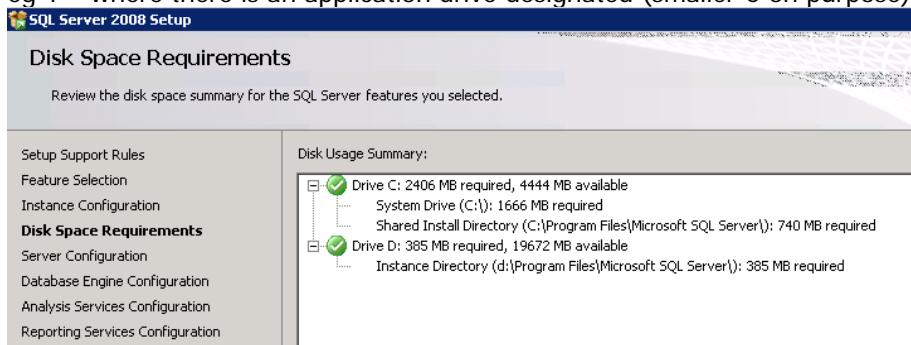


Click Next (make sure everything is run on C drive in our case, or D if there is a dedicated Application drive)

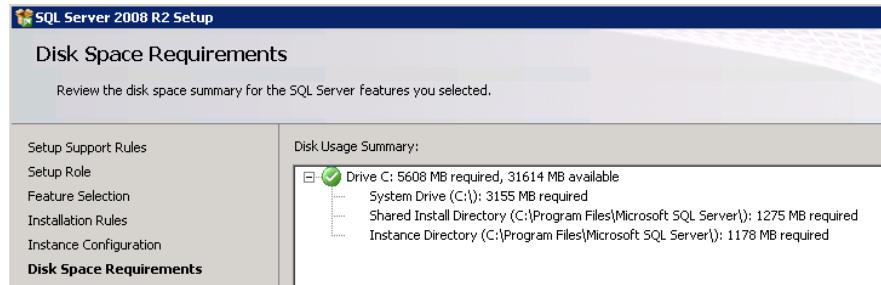
Here is where, as mentioned above, if you need to create a specific instance\alias name, you would specify it by selecting **named instance**.

10) Validate Disk Space Requirements and click next

eg 1 – where there is an application drive designated (smaller C on purpose)

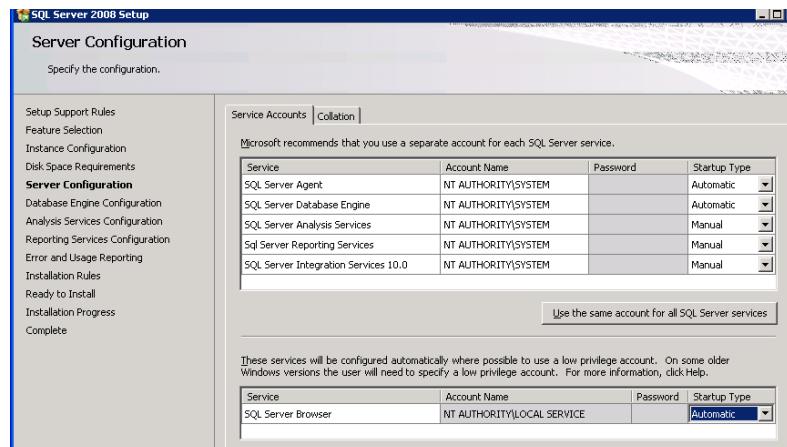


g 2 where the C drive is significant in size to hold system databases also.

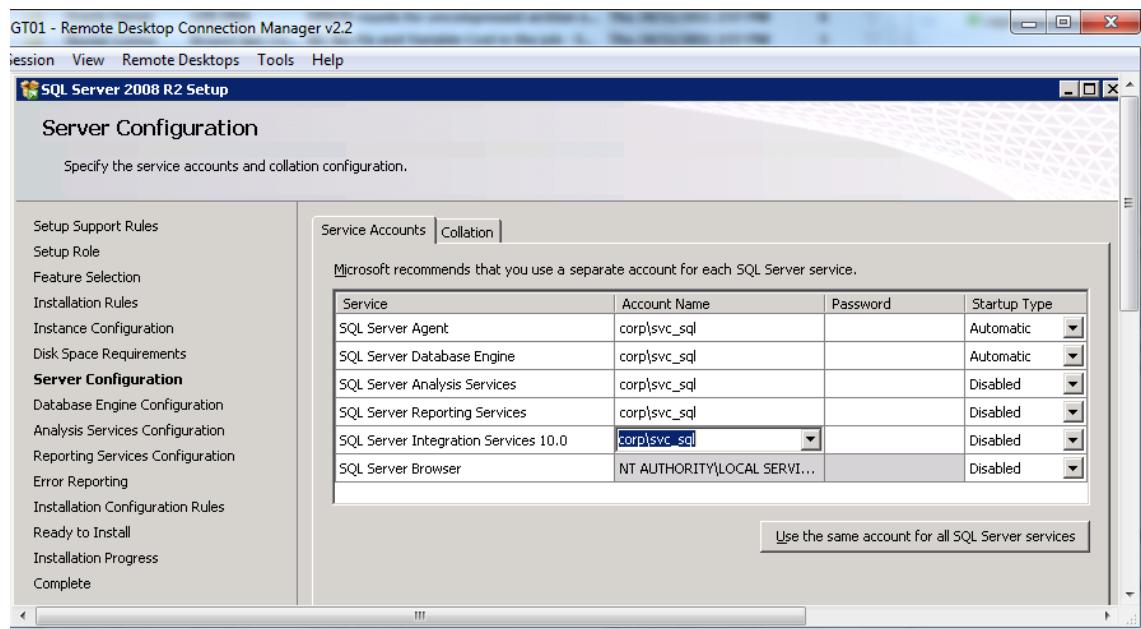


11) Server Configuration

Click on Use the Same account for all SQL Server services
eg1 for a development server just use the system account.



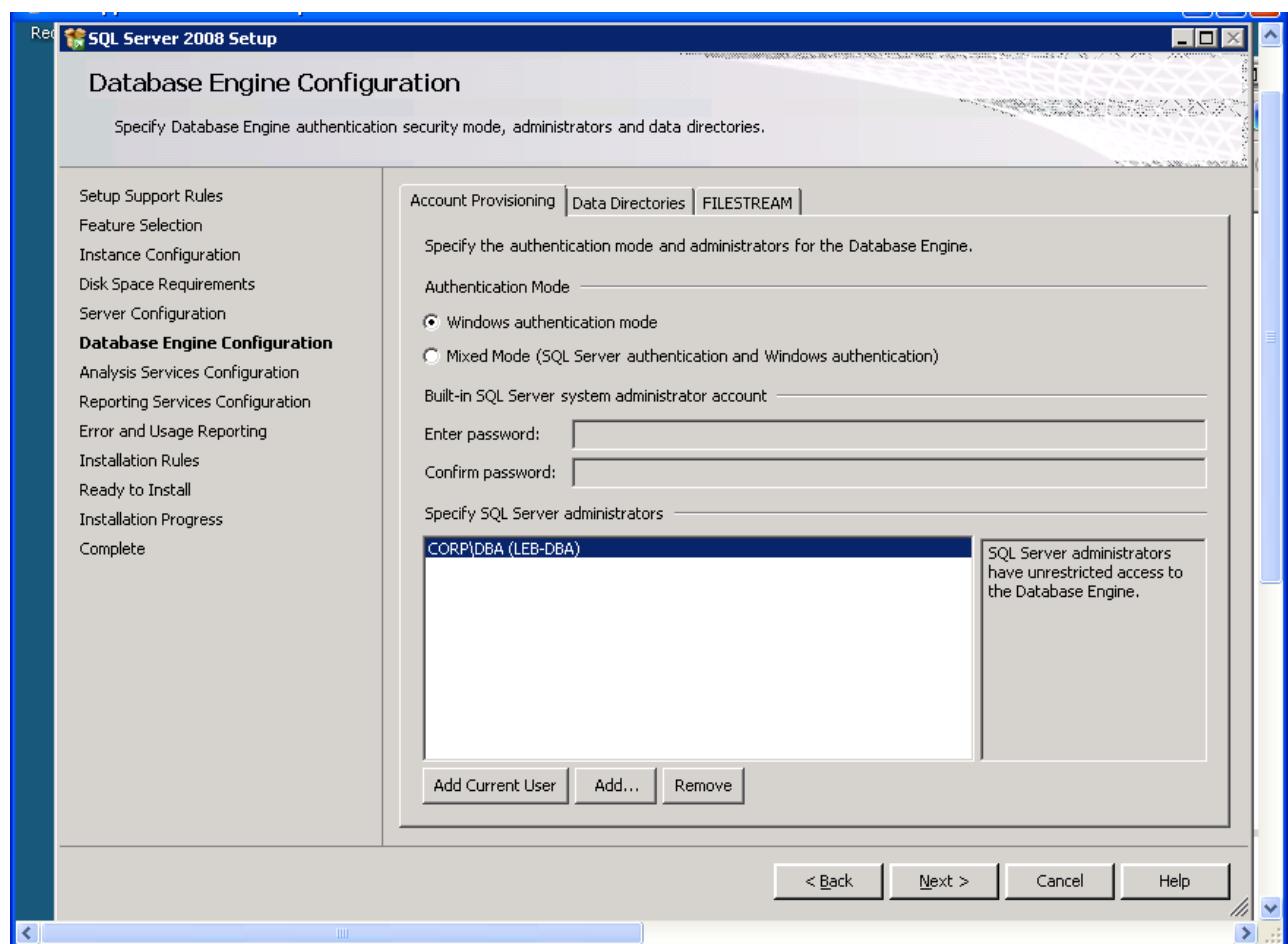
12) For a production SQL Server chose Domain\SQLDomainServiceAccount



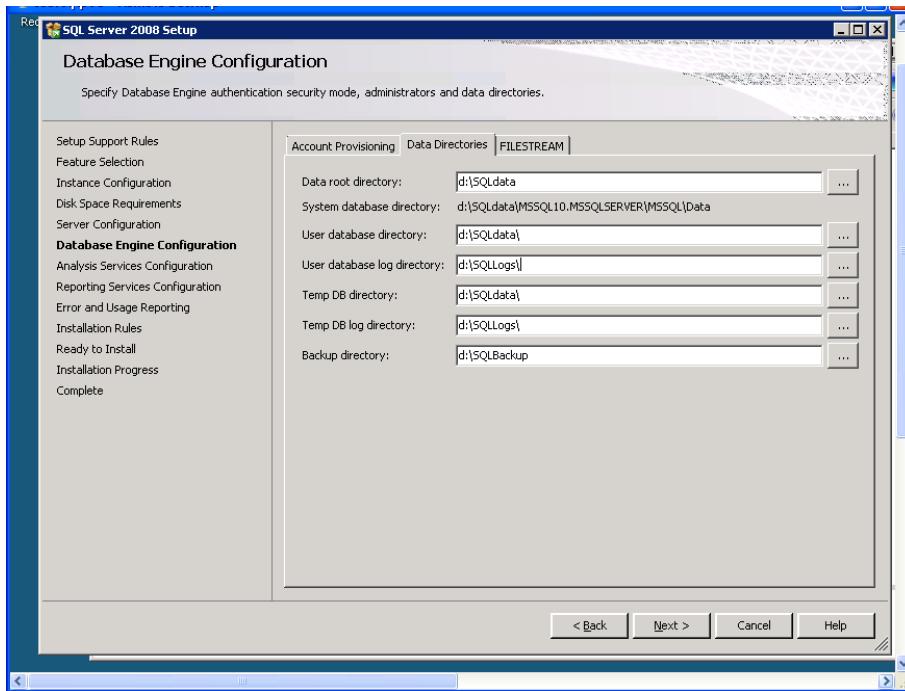
If you have the password handy (**NEED SYSADMIN help, DBAs do not know this password for SODuties reasons**), click User the same account for all SQL Server services:
Add **Domain\SQLDomainServiceAccount** - startup is auto for the Engine and Agent, leave the rest disabled until needed, if they are at all.
Leave the Collation tab as is, SQL_Latin1_General_CI_AS is the default and is Accent Sensitive and Case Insensitive.

13) Select Windows Authentication mode, and add the Domain\DBA group (or the specific members as per your specific organisation).

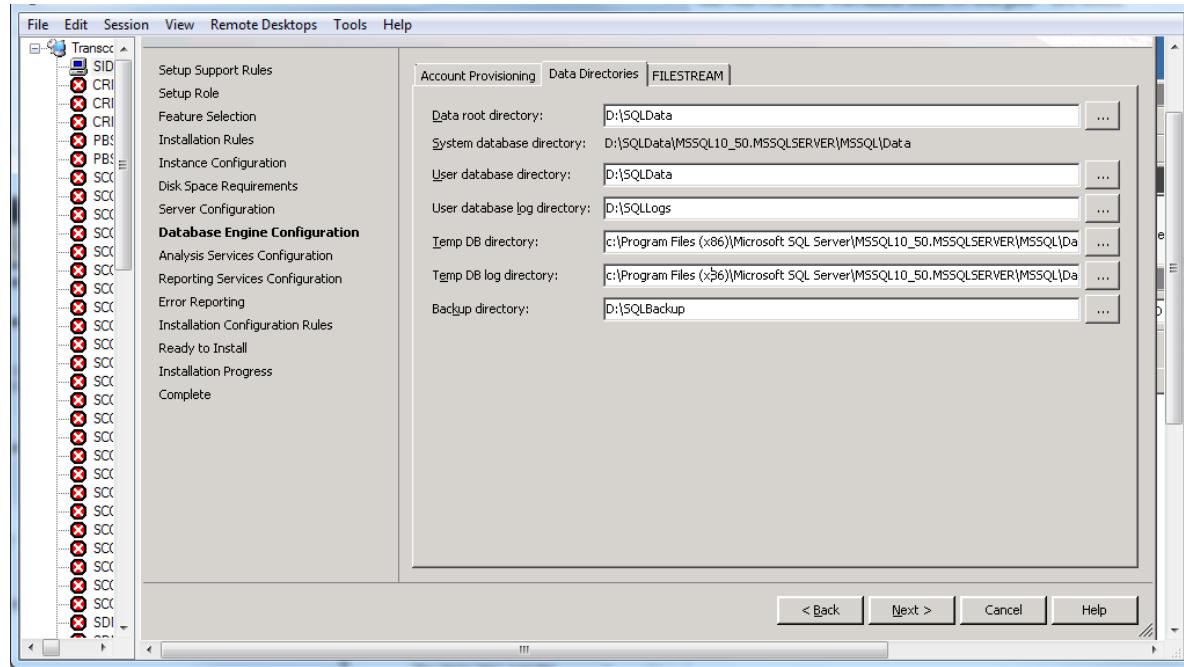
If you are installing the **Client**, this step of the installation process will not exist, move forward to the matching confirmation screen in this document. In the case of a local development installation, add the specific domain user, so that they may operate their own instance. I usually switch to Mixed Mode AFTER the installation, and add alternative accounts for the specific DBA group members with SA privileges.



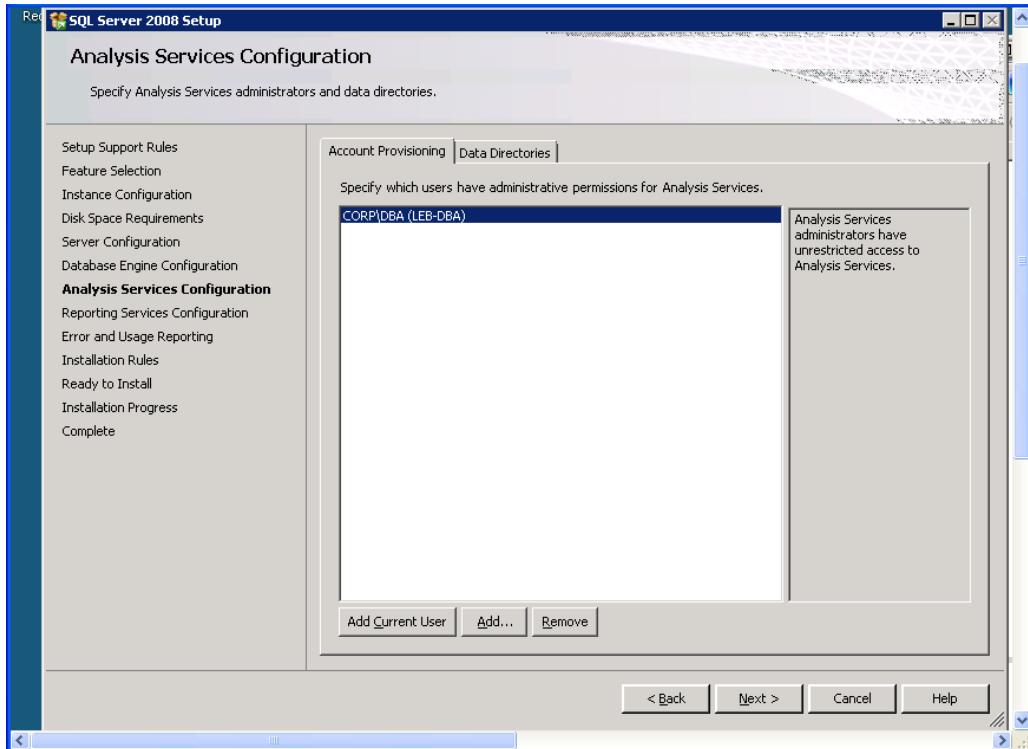
13) Root directories for Data, System DBs., User DBs., Temp and Backups.
Eg 1 – dedicated D drive for SQL Server and no space on C drive, nor a dedicated Temp Drive. For this installation I did not have other disks, but I prefer to have a drive for Temp dedicated, as well as one for Logs. Make sure to choose those drives here!



e.g. if your C drive is large enough, you can leave the Temp DB there (and ideally, if performance becomes an issue, ask the VM admins to dedicate a 10-15GB drive just for the temp database on a separate dedicated disk).

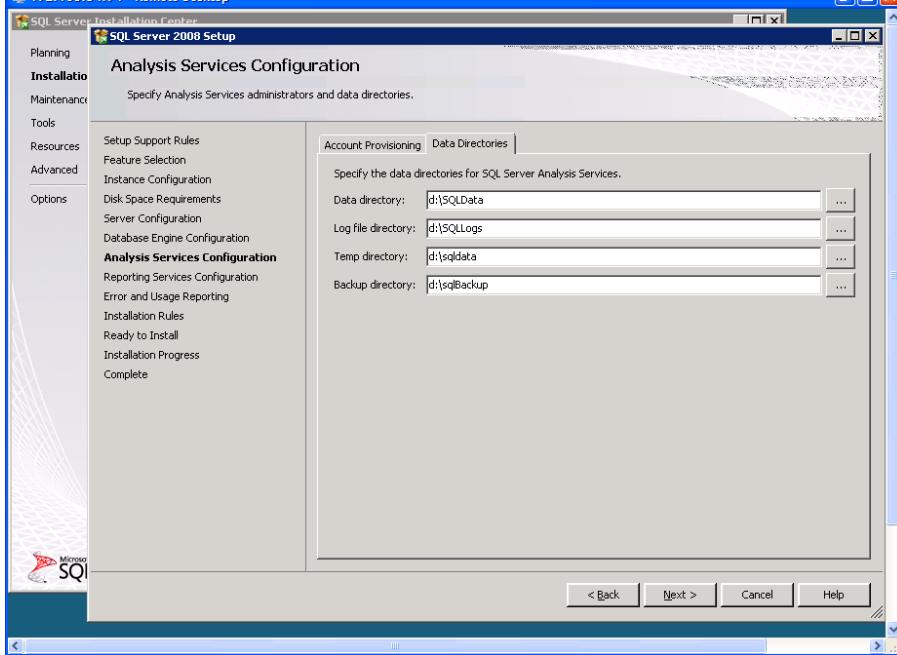


- 14) Add your DBA group or in the case of a local development installation, the domain user, so that they may operate their own instance.

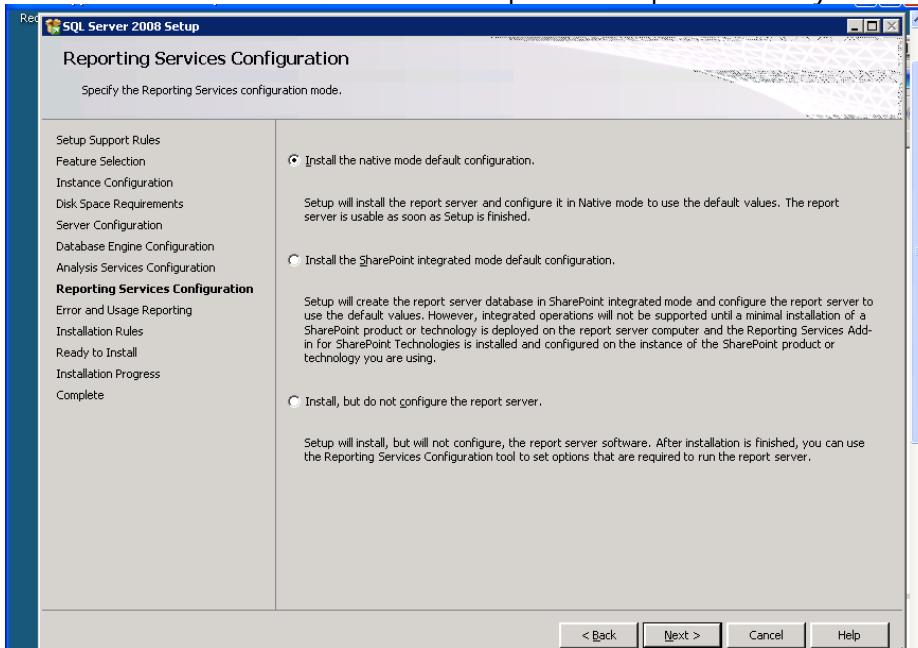


I put the data and log files at the root (d:\SQLdata, d:\SQLBackups, d:\SQLLogs), so they are easier to find, but this all depends on how you set up your drives – ideally a situation like: d:\sqldata, e:\sqllogs, f:\sqlbackups and g:\tempDB

- 15) Choose the same Data, Log, Temp folders as for the Database Engine Configuration – unless you have a dedicated drives for the Analysis Services instance (OLAP / DW)
- 16)



- 17) Click next – and if you have installed Reporting Services, use this:
 18) Please read below and chose the best setup for the requirements of your server.



If this server is to be used for SharePoint, select the second radio button for SharePoint integration.

For more regarding SQL Server 2008 R2 and SharePoint, please read the following:

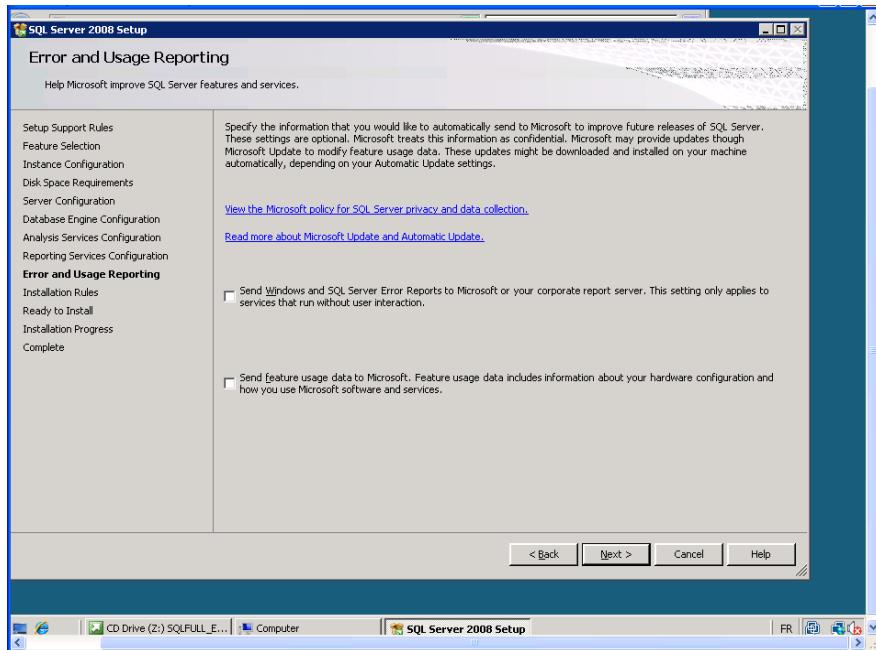
SQL Server 2008 R2 Reporting Services continues to improve integration with SharePoint. In this release, you find better options for configuring SharePoint 2010 for use with Reporting Services, working with scripts to automate administrative tasks, using SharePoint lists as data sources, and integrating Reporting Services log events with the SharePoint Unified Logging Service.

Improved Installation and Configuration

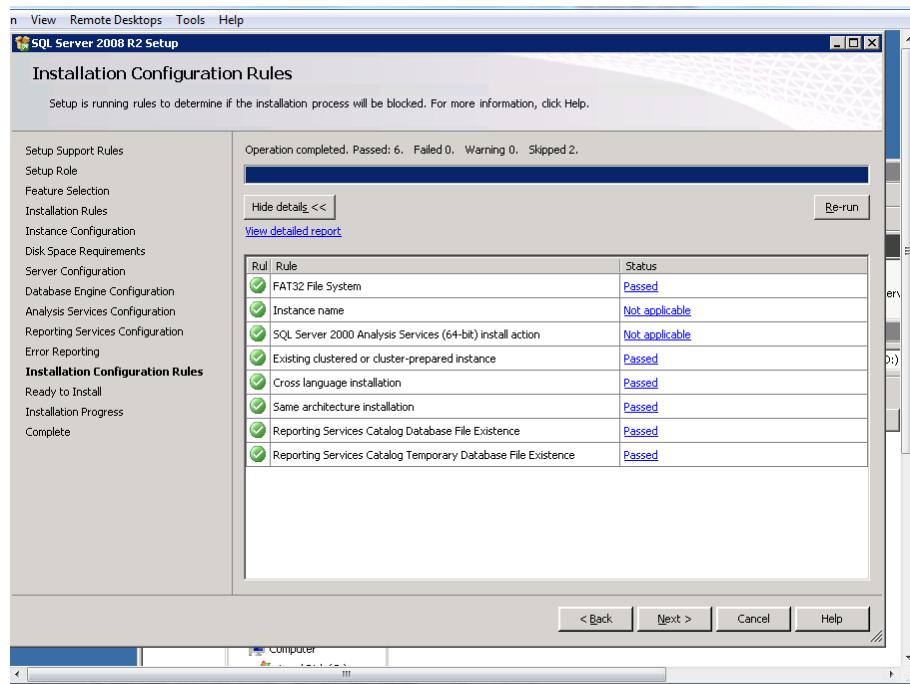
The first improvement affects the initial installation of Reporting Services in SharePoint integrated mode. Earlier versions of Reporting Services and SharePoint require you to obtain the Microsoft SQL Server Reporting Services Add-in for SharePoint as a separate download for installation. Although the add-in remains available as a separate download, the prerequisite installation options for SharePoint 2010 include the ability to download the add-in and install it automatically with the other prerequisites. After you have all components installed and configured on both the report server and the SharePoint server, you need to use SharePoint 2010 Central Administration to configure the General Application settings for Reporting Services. As part of this process, you can choose to apply settings to all site collections or to specific sites, which is a much more streamlined approach to enabling Reporting Services integration than was possible in earlier versions. Another important improvement is the addition of support for alternate access mappings with Reporting Services. Alternate access mappings allow users from multiple zones, such as the Internet and an intranet, to access the same report items by using different URLs. You can configure up to five different URLs to access a single Web application that provides access to Reporting Services content, with each URL using a different authentication provider. This functionality is important when you want to use Windows authentication for intranet users and Forms authentication for Internet users.

16) I would select the Error Report transfer if this if the first time you are using a SQL 2008 install, so that Microsoft Support can easily help you in case of issues.

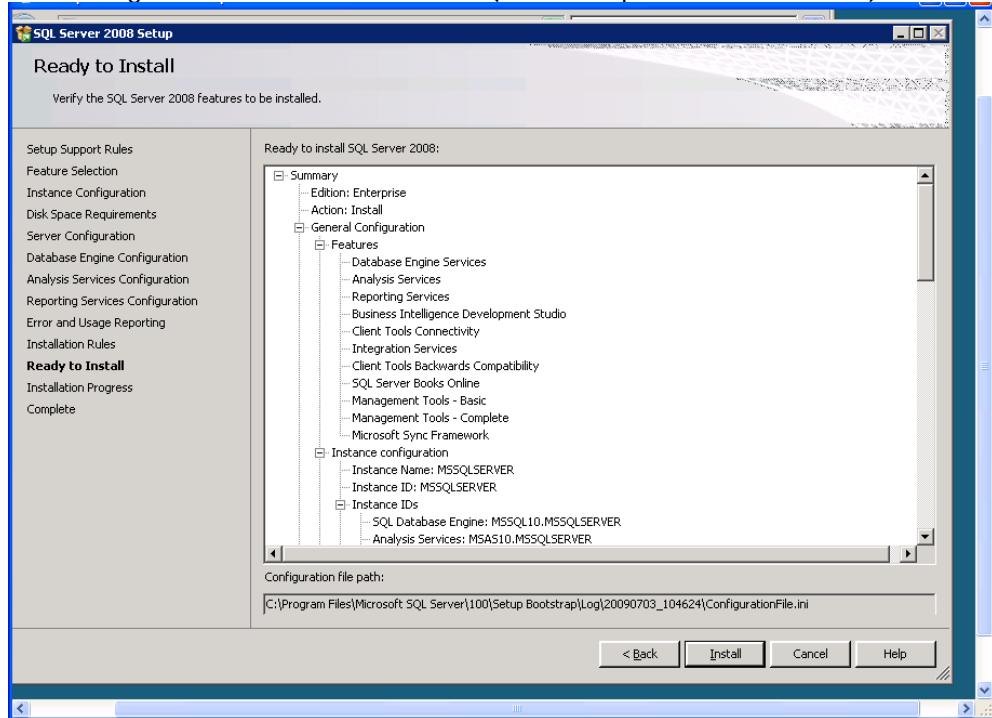
i.e. If this is a server with applications you have never worked with, check the first box (only one in R2) to help with MSFT diagnostics of problems with the server.



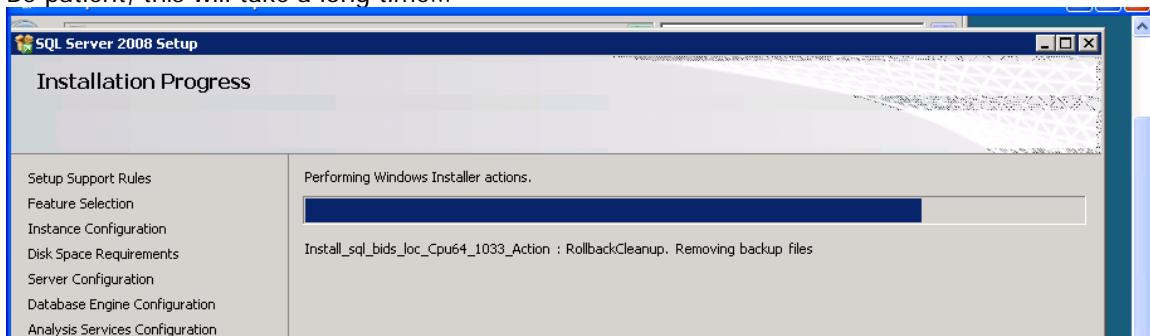
17) Click Next, now that all the 'Installation configuration rules' are taken care of.



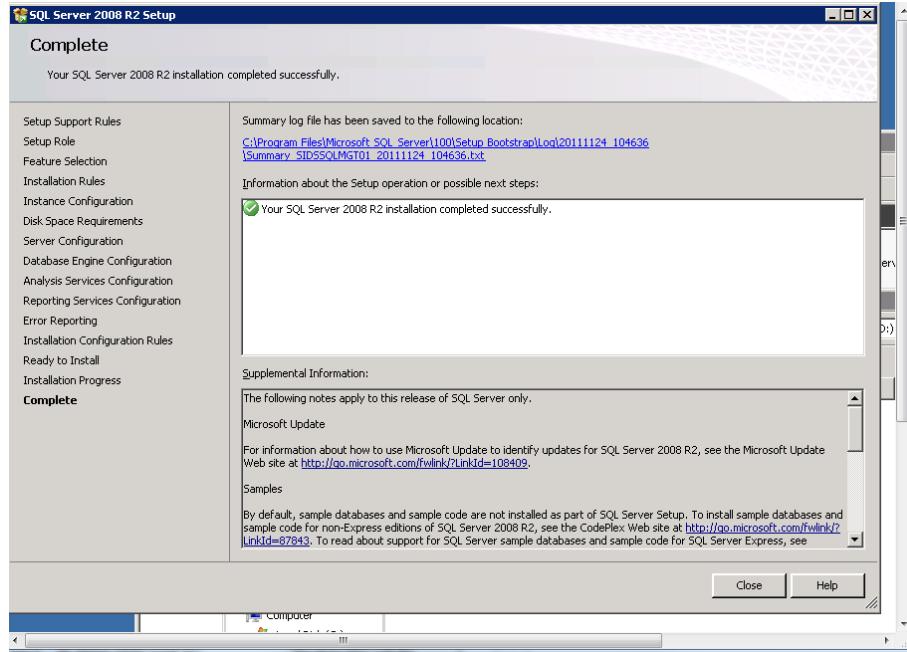
18) The following is to validate the installation (this is the point of no return ☺):



The install will take a while (20-25 minutes depending on the strength of the machine). Be patient, this will take a long time...



19) Next, and then close next page upon confirmation of successful install.



SQL Server 2012 Installation:

The installation process is simple, straight forward and is very similar to SQL Server 2008 / 2008 R2 setup. The procedure described is for a non-clustered server, and can be applied either to a default instance or a named instance. It is intended to be read by Database Administrators and anybody interested in the technical aspects of carrying out this task.

Before beginning, you should check that minimum hardware and software requirements to install and run SQL Server 2012 have been met..

Preparing for Installation

Identify drives to which the databases and/or logs will be backed up, ensuring that there is enough disk space to accommodate the backups for the retention period that you choose.

Identify drive that will be used for data or log files. These will usually be on SAN storage and hence on a different drive from the operating system and SQL Server installations. Decide which folder will be the primary location for data files: you will need to specify this during the installation. If you are installing Analysis Services, identify drives that will host the data and log files for the cube(s).

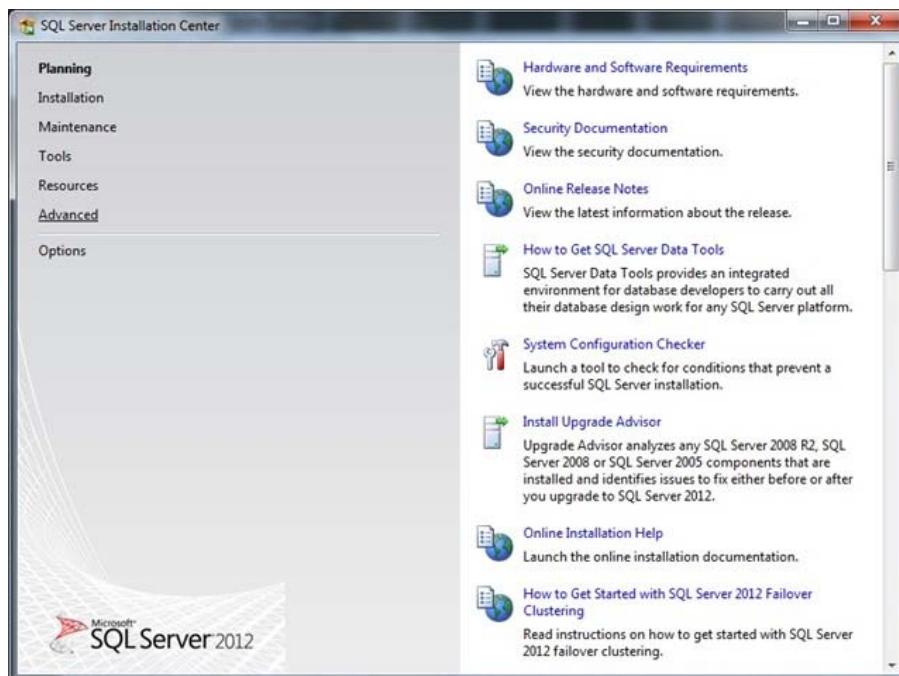
Create dedicated service account for each of the service to be installed. Ensure that these accounts are not member of Local Administrator because it will give unnecessary rights to these accounts. Note: for the purpose of this tutorial I'm using local system accounts as a service start-up accounts.

Assign **Deny logon locally** right to these service accounts.

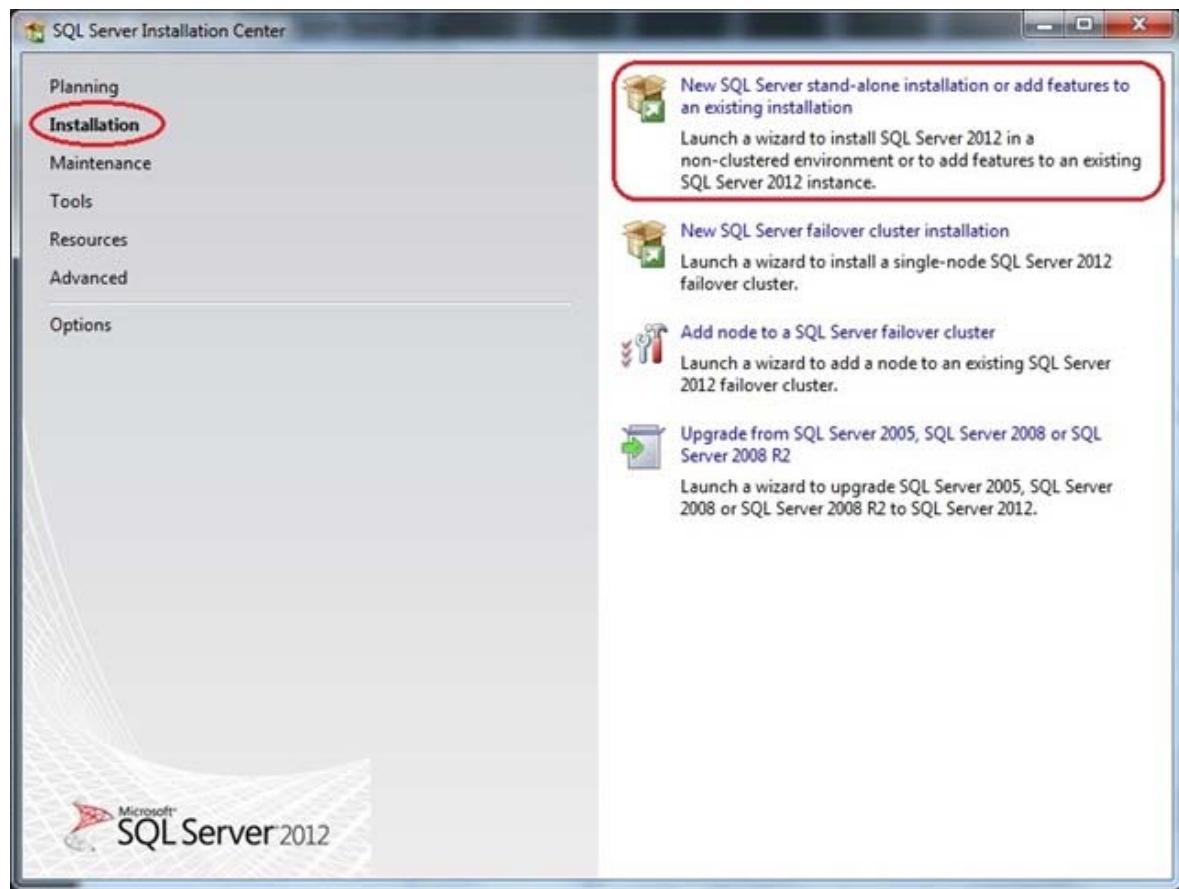
Choose the port number for your SQL Server installation because common TCP\IP ports 1433/1434 ports are well known and are common target for hackers. Therefore it is recommended to change default ports associated with the SQL Server installation.

Installing SQL Server 2012

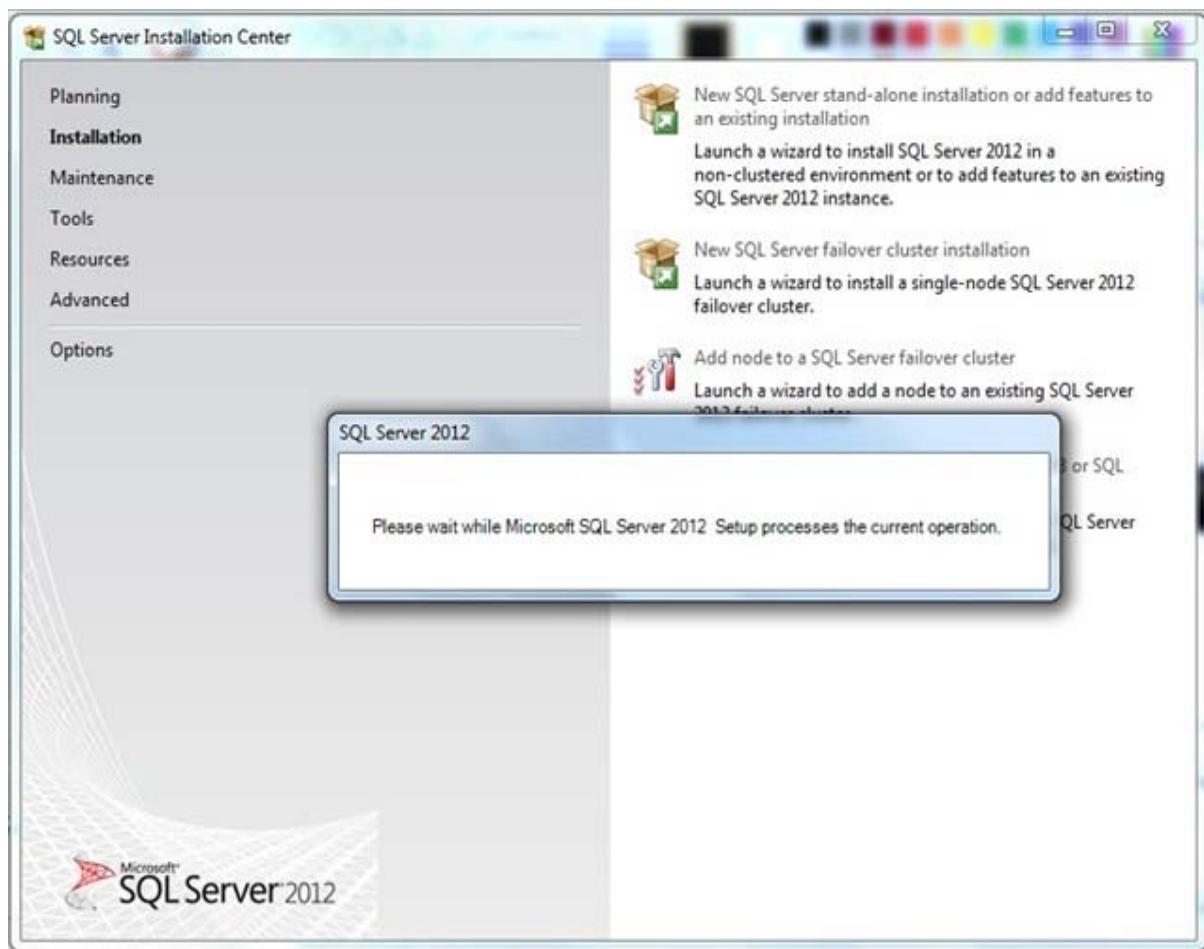
Insert the SQL Server installation media. From the root folder, double-click **Setup.exe**. To install from a network share, locate the root folder on the share, and then double-click **Setup.exe**:



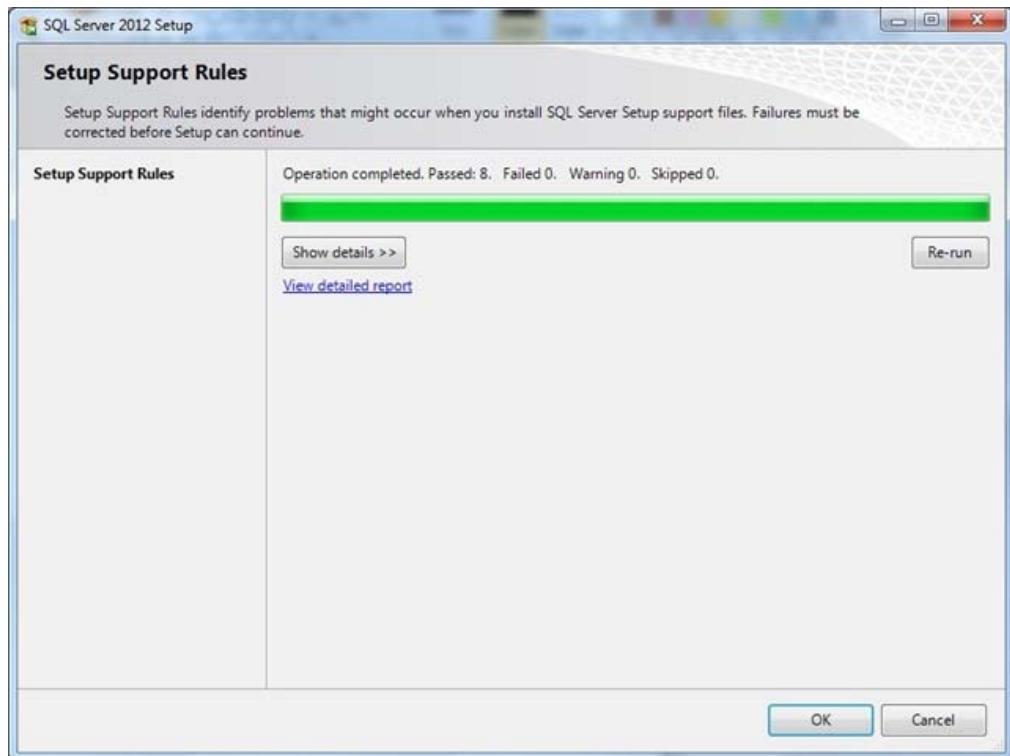
The Installation Wizard runs the SQL Server Installation Center. To create a new installation of SQL Server, select the Installation option on the left side, and then click **New SQL Server stand-alone installation or add features to an existing installation**:



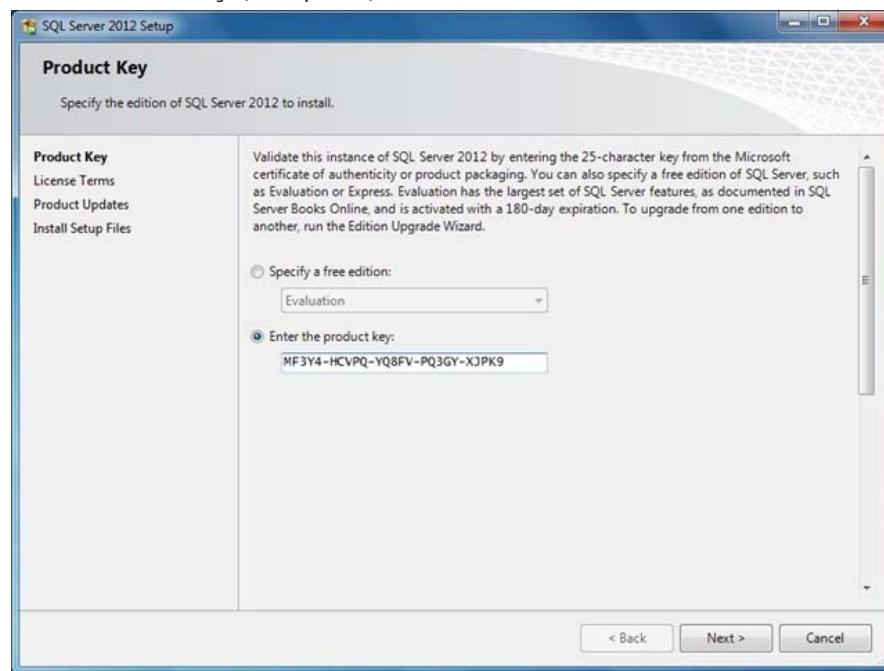
Setup is now preparing to launch Setup Support Rules window:



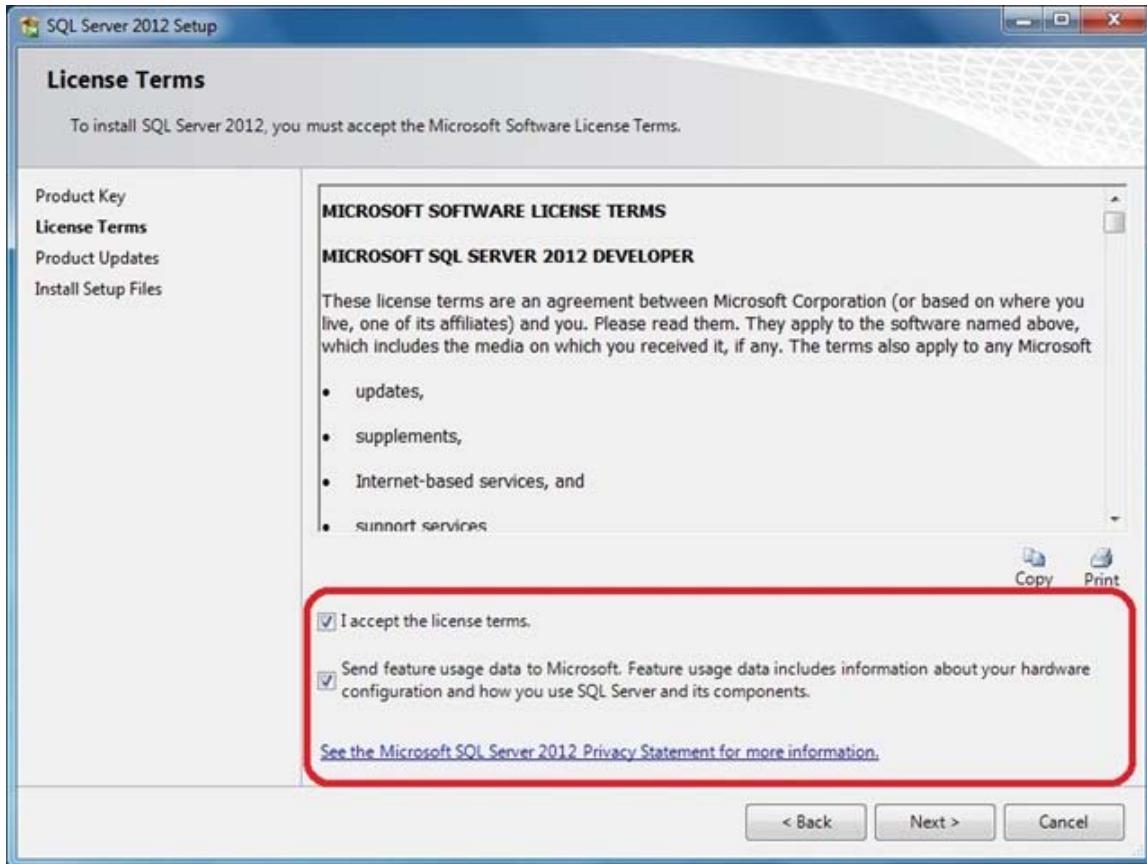
Now Setup Support Rules will run to identify problems that may occur during the Setup Support Files installation:



Once this step finishes click **OK** to proceed to Product Key window, In the Product Key window, enter the Product license key (if required), and click **Next** to continue:

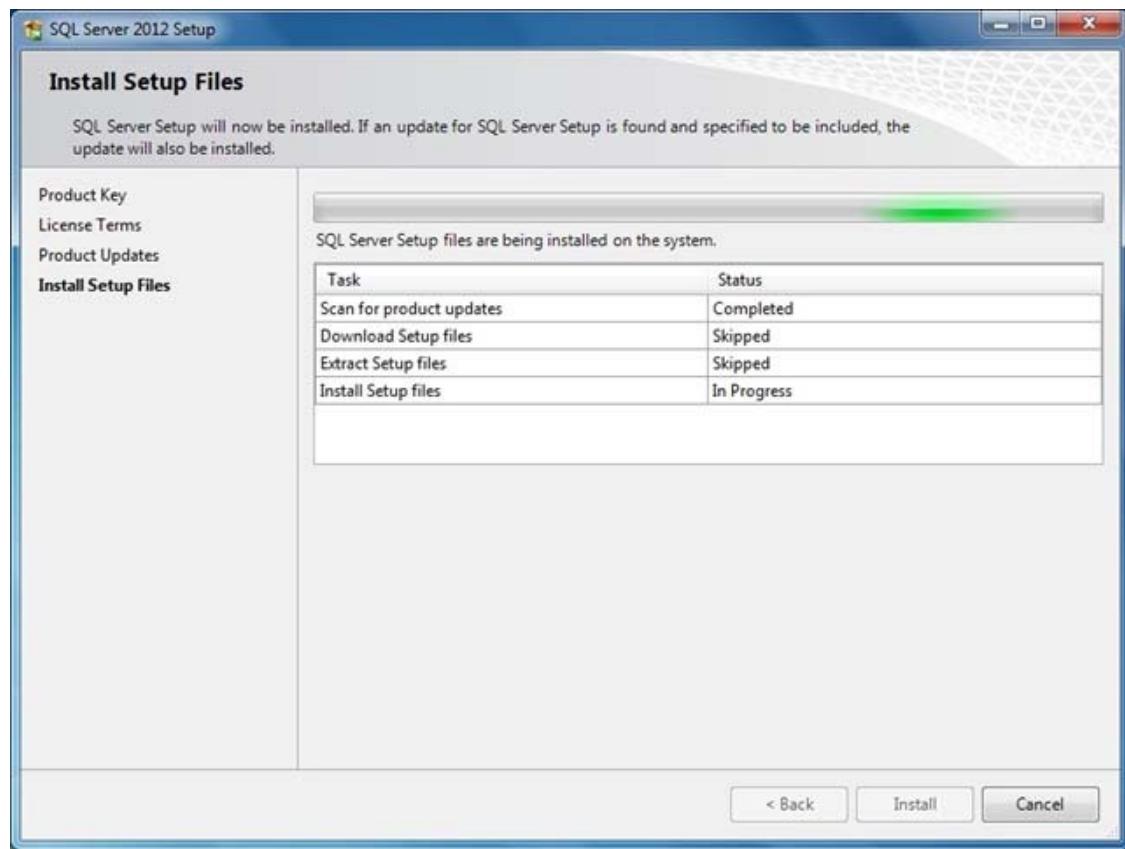


In the License Terms window, tick the box I accept the license terms and then click **Next** to continue:

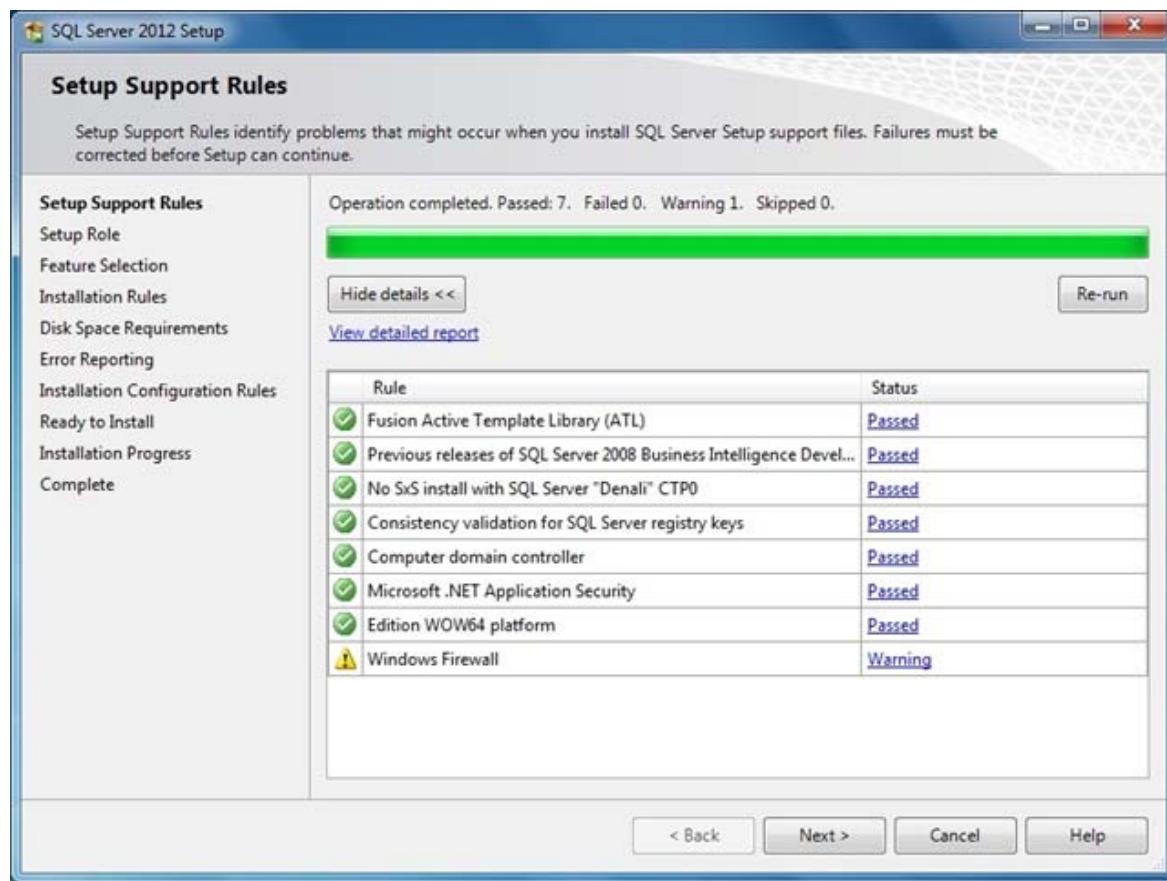


Note: You must accept the license agreement before you can continue the installation of SQL Server 2012. Send feature usage data to Microsoft option is optional.

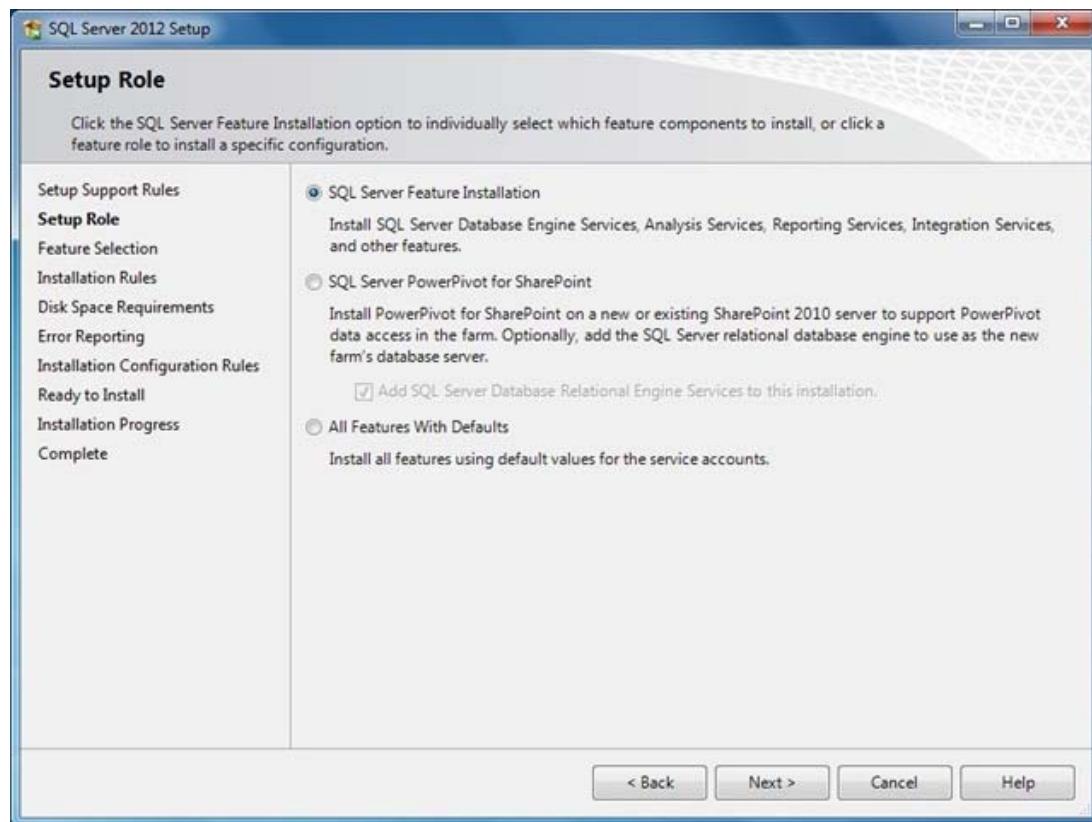
Click **Next** and then click **Install** to **Install Setup Files**:



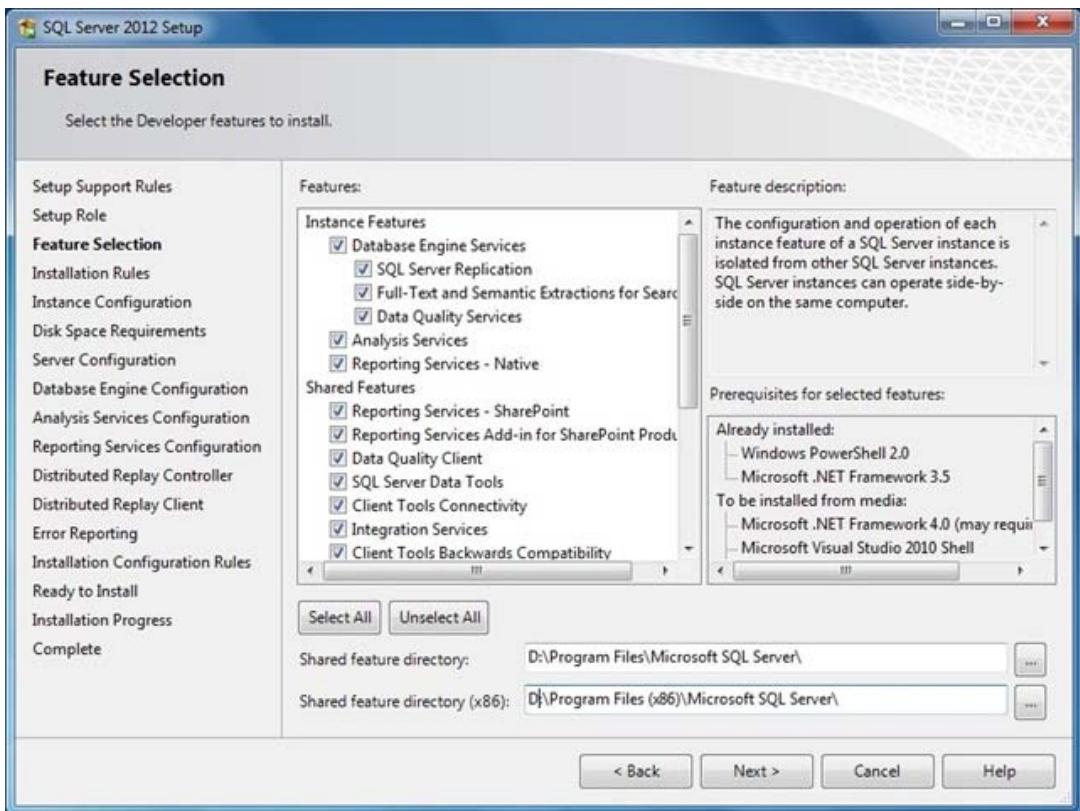
These files are necessary to perform the actual installation. Following the installation of the setup support files, you will be presented with another compatibility check. Following dialog appears once you successfully pass these checks. You can click the **Show Details** button under the green progress bar if you want to see the individual checks listed:



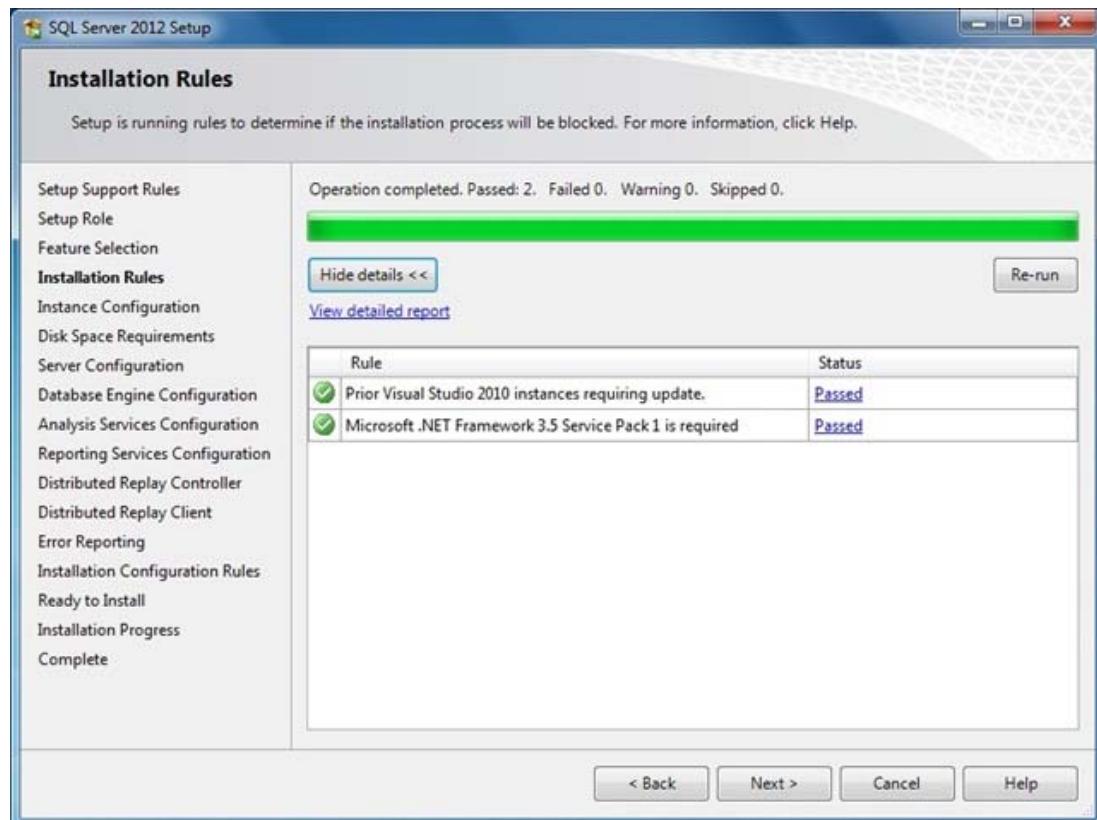
Click **Next** to continue to the Setup Role page:



Select **SQL Server Feature Installation** option and then **Next** to continue to Feature Selection page:

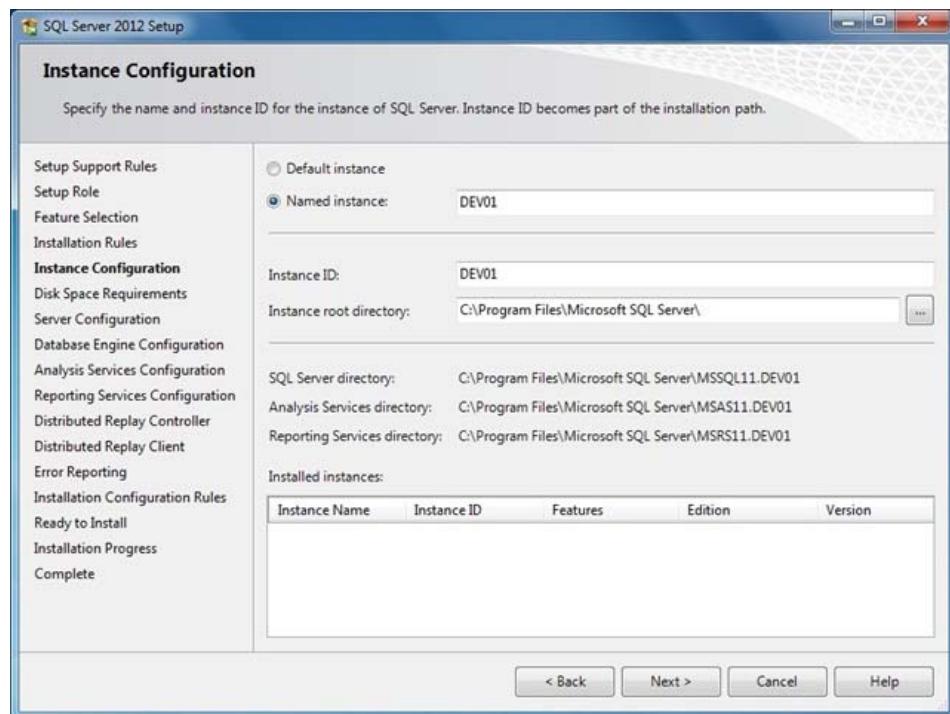


I'm going to select all features so that we can walk through a complete installation process. This will install the Database Engine Services, Analysis Services, Reporting Services, and a number of shared features including SQL Server Books Online. You can also specify the shared feature directory where share features components will be installed. Click **Next** to continue to the Installation Rules page. Setup verifies the system state of your computer before Setup continues:



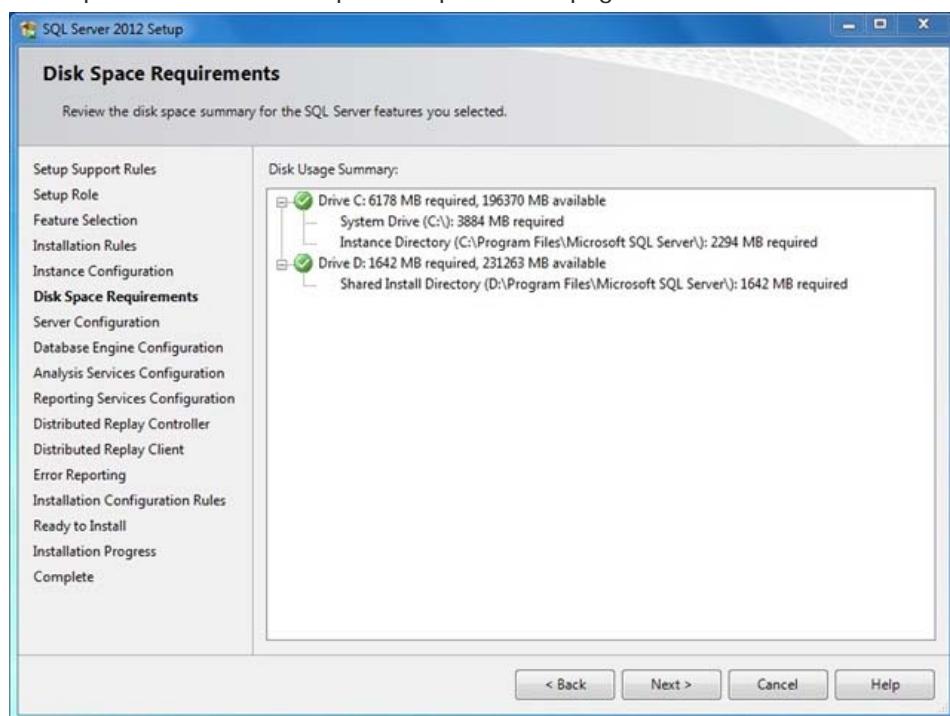
Click **Next** to continue to the Instance Configuration page. Each server machine can host one default instance of SQL Server, which resolves to the server name, and multiple named instances, which resolves to the pattern ServerName\InstanceName.

In this sample installation, I will install a named instance of SQL Server 2012 called DEV01:

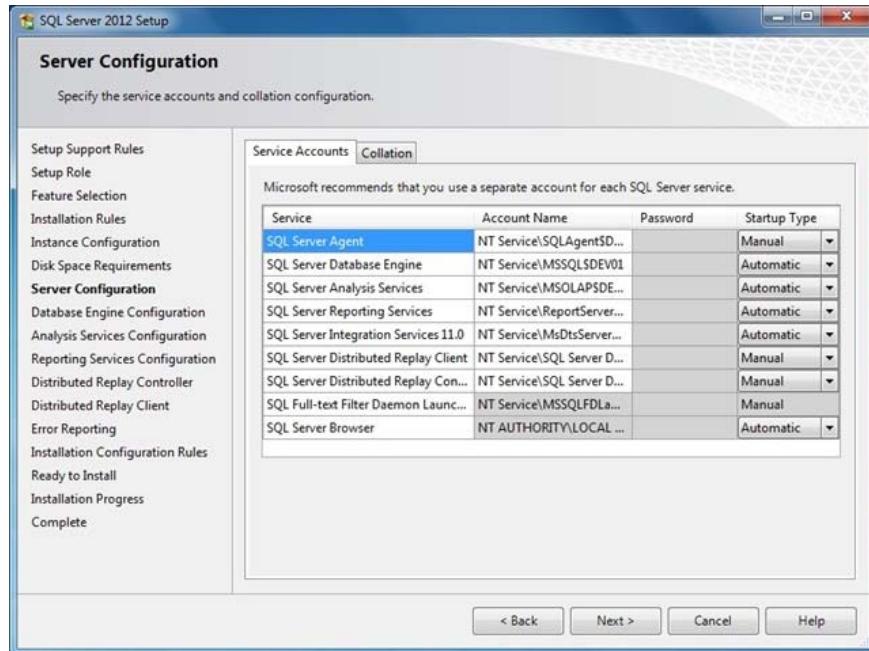


Note: This screen also reports on other instances installed on this machine.

Click **Next** to proceed to the Disk Space Requirements page:



This is just a information page that does not require you to make any choices. Click **Next** to go to Server Configuration page:



Here you specify service startup and authentication. Microsoft recommends that each service account have separate user accounts as a security best practice as shown in the following figure:

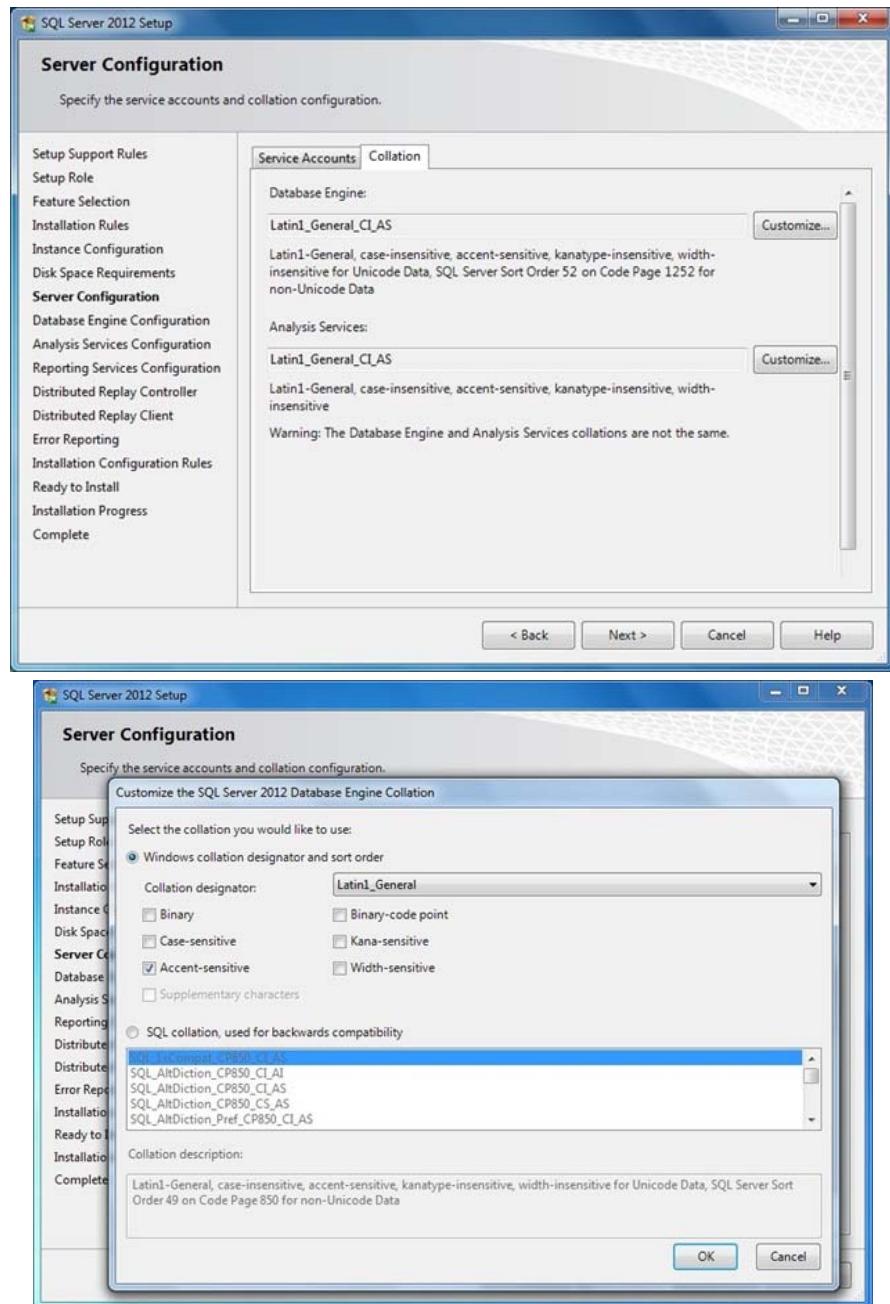
svc_SQL	User	SQL Server service account
svc_SQLBrowser	User	SQL Browser service account
svc_SQLReportServer	User	SQL Report Server service account
svc_SQLServerAgent	User	SQL Server Agent service account
svc_SQLServerOLAP	User	SQL Server OLAP service account

The SQL Server 2012 Books Online makes the following security recommendations:

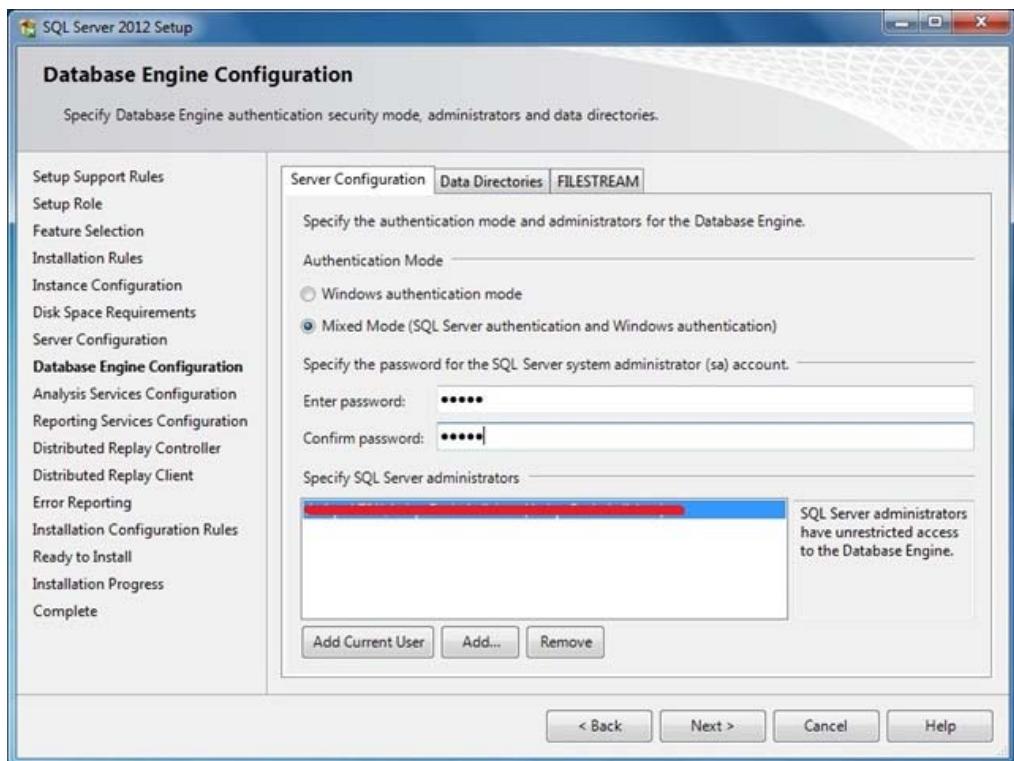
- Run separate SQL Server services under separate Windows accounts.
- Run SQL Server services with the lowest possible privileges.
- Associate SQL Server services with Windows accounts.
- Require Windows Authentication for connections to the SQL Server.

For the purposes of this article, we will authenticate all services using Local System account.

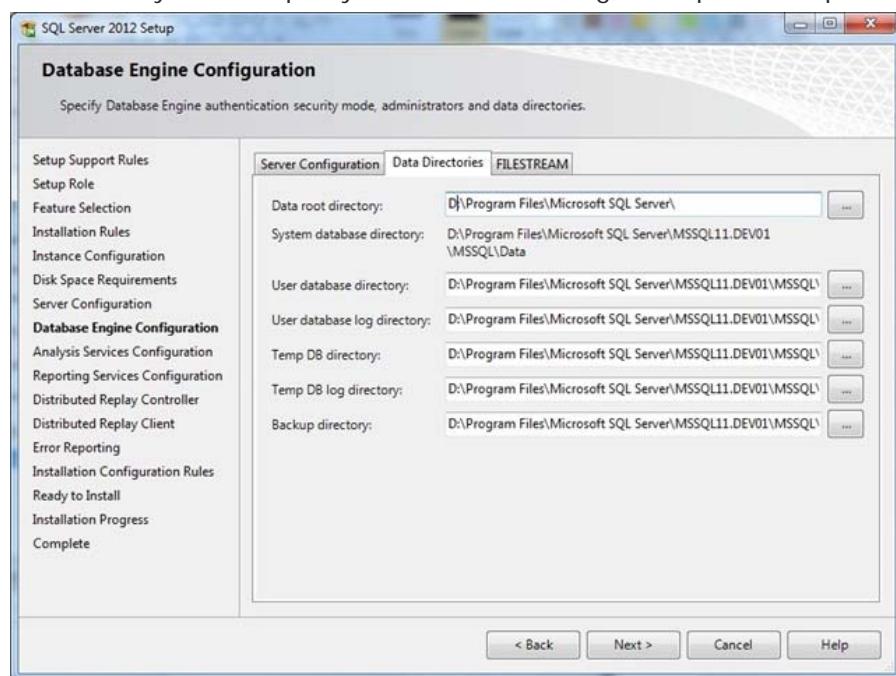
You may have noticed the Collation tab here in this page. Click **Collation** tab and then click Customize button to specify the collation for your Database Engine and Analysis Services instance that best matches your application need:



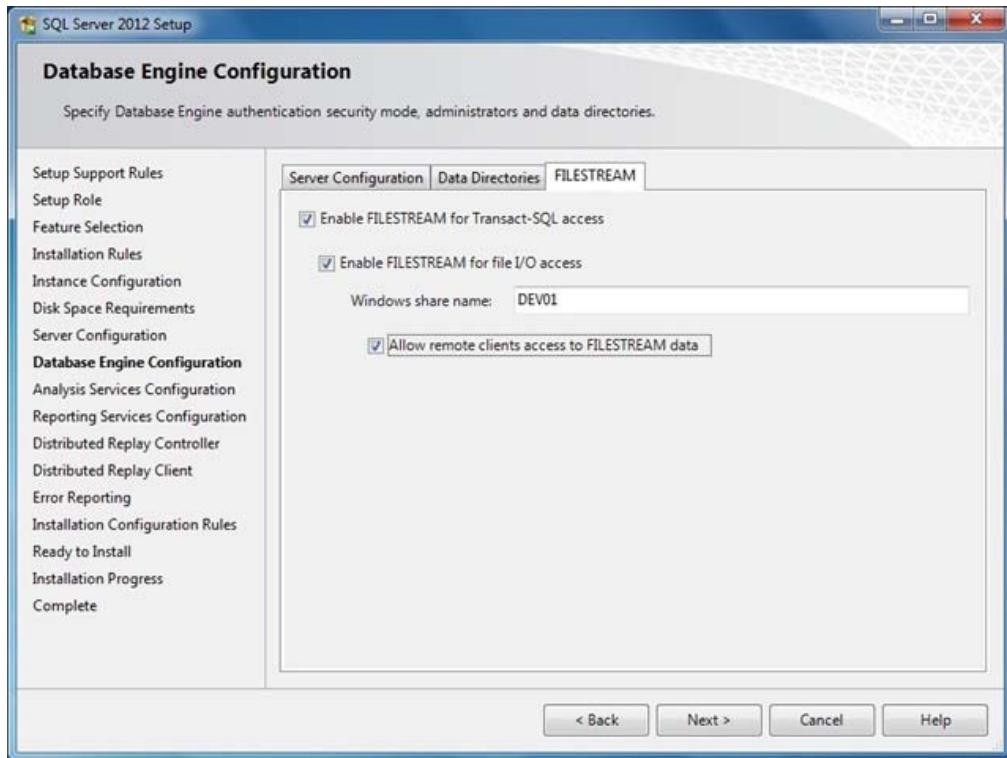
Click **Next** to continue to Database Engine Configuration page. The SQL Server 2012 authentication mode was configured for "Windows Authentication Mode" per Microsoft security best practice. For the purposes of this article, I will be choosing "Mixed Mode Authentication". You will also need to specify the SQL Server administrators to be used; in this example I will use the current logged in user by clicking **Add Current User** button:



Click Data Dictionary tab and specify default database, log, backup, and tempdb locations:



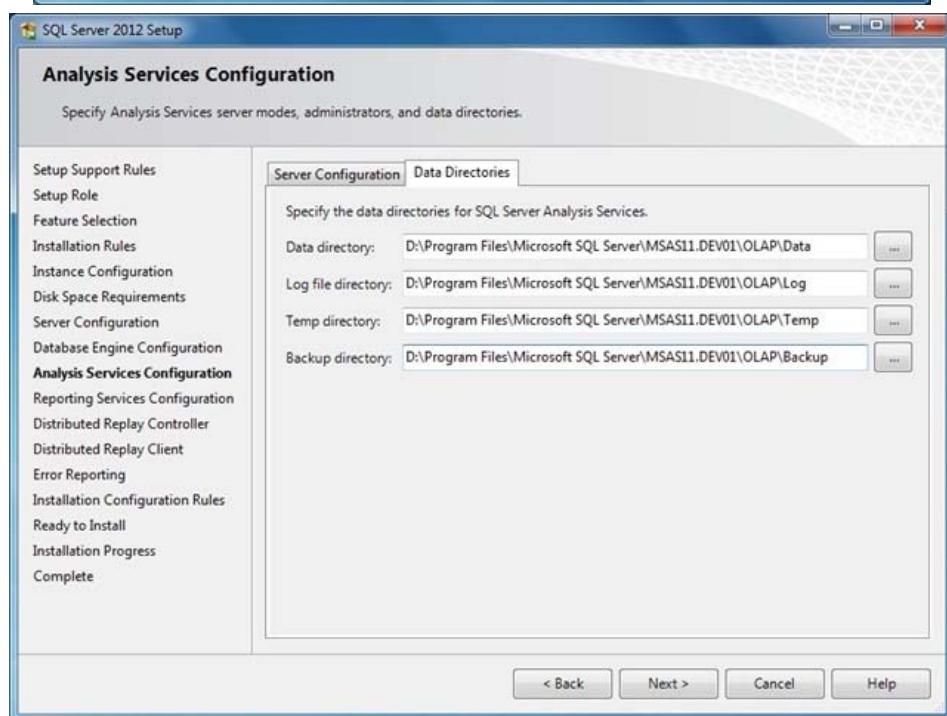
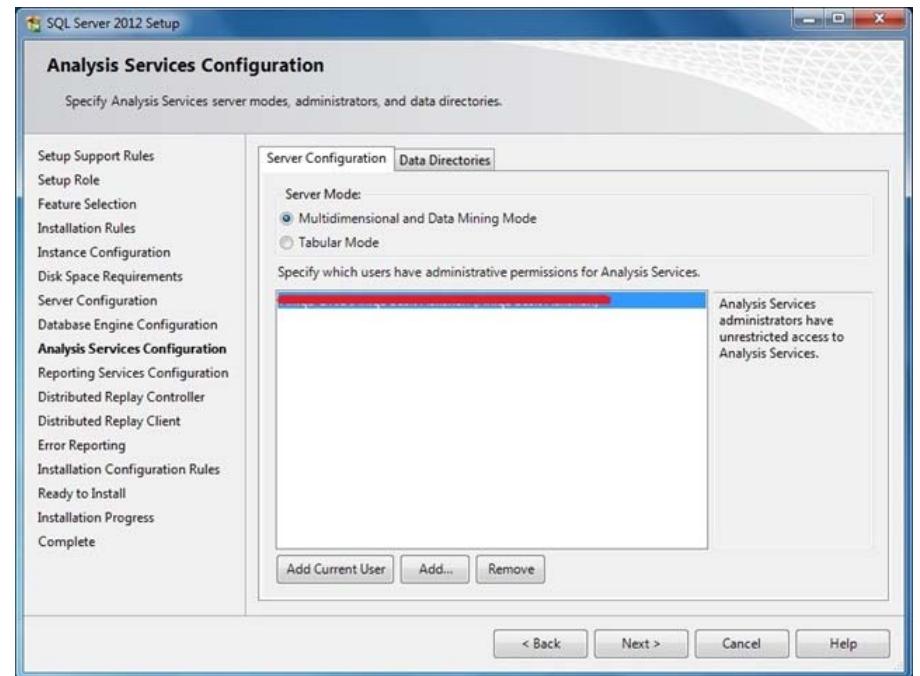
Click FILESTREAM tab and enable this feature as below:



FILESTREAM feature is quite useful when binaries or other data that does not fit neatly in a table structure.

Click **Next** to proceed to Analysis Services Configuration page, this is similar to Database Engine Configuration page.

Specify users or accounts that will have administrator permissions for Analysis Services in Account Provisioning page and specify non-default installation directories in Data Directories page:



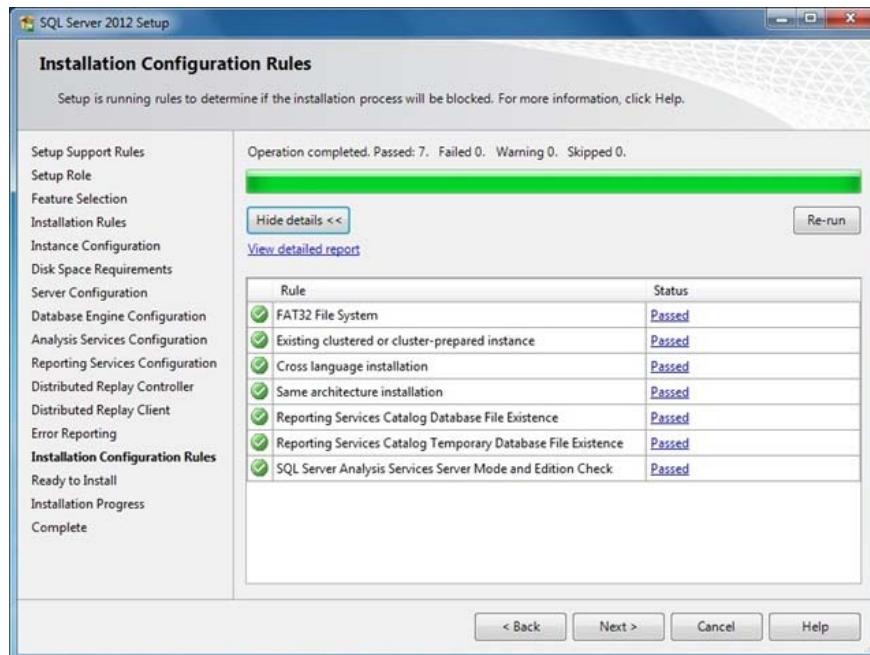
Click Next to continue to Reporting Services Configuration page, specify the kind of Reporting Services installation to create. For the purpose of this article, I will use Install and Configure option:



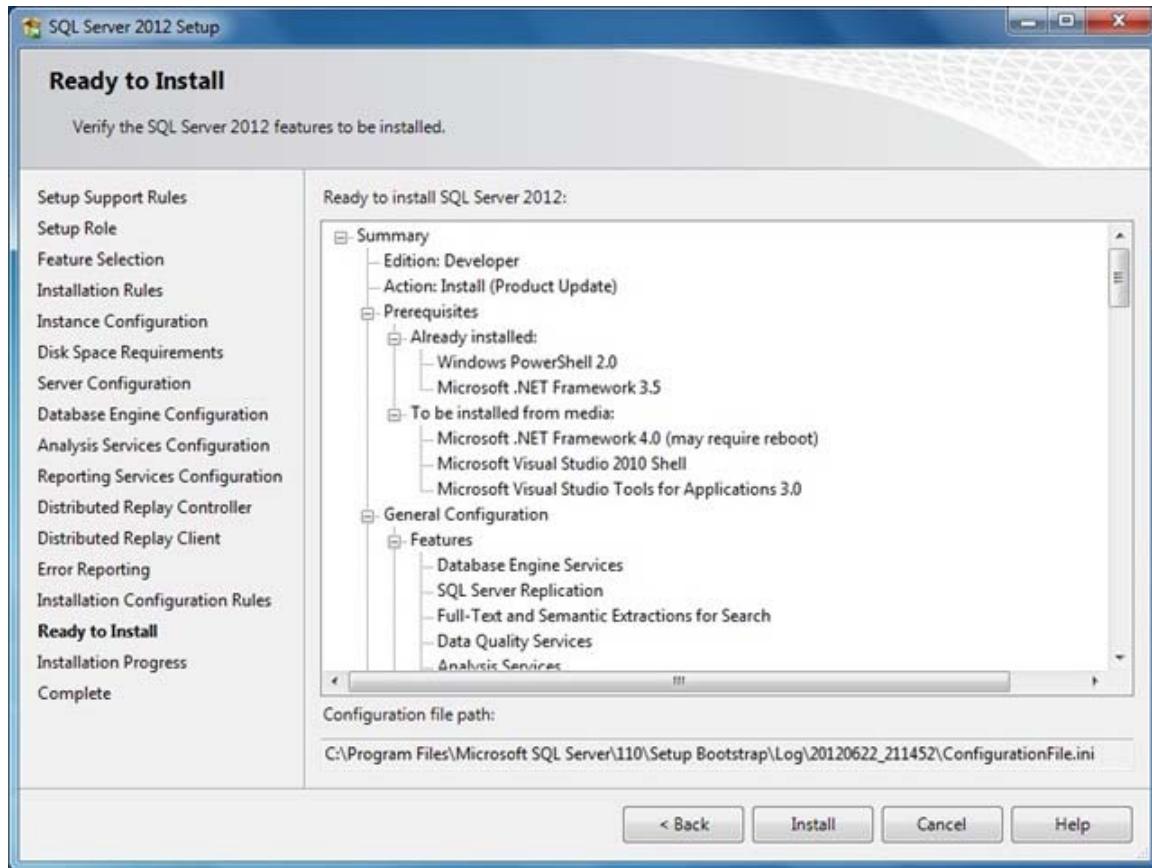
Click **Next** to advance to Error Reporting page, tick the check box if you want to send Windows and SQL Server error reports to Microsoft:



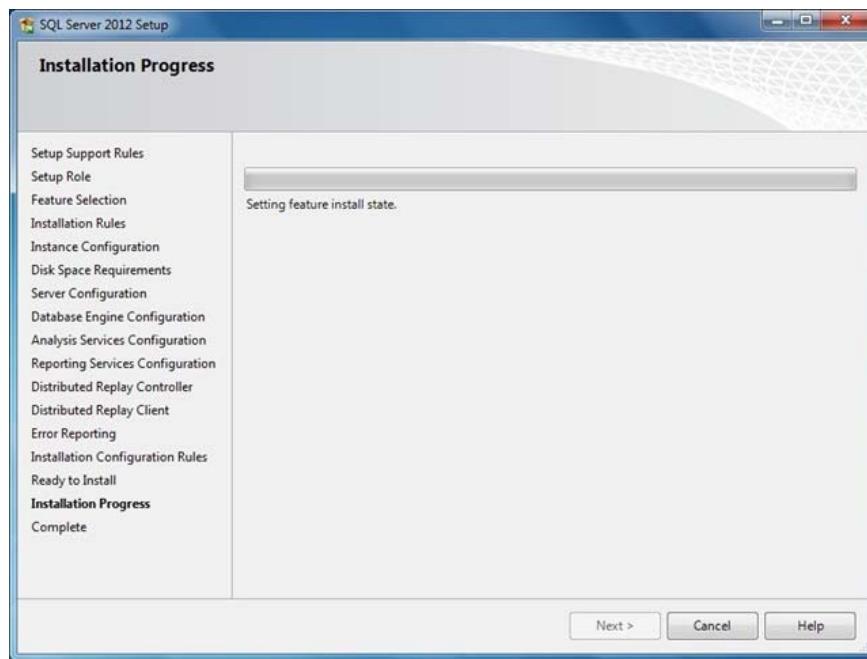
Click **Next**, Now System Configuration Checker will run some more rules that will validate your computer configuration with the SQL Server features you have specified:



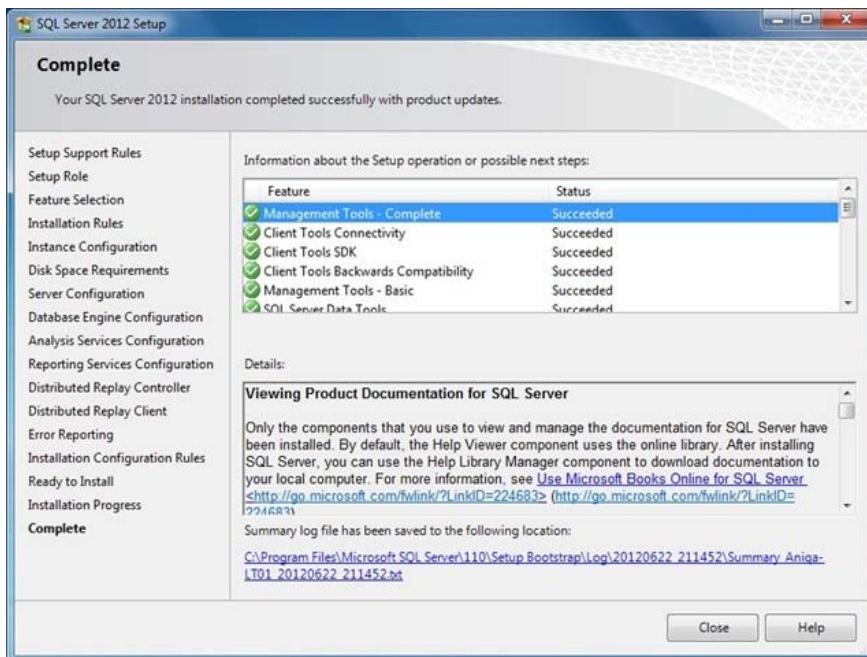
Correct any errors reported in the Installation Rules screen and click on **Next** to advance to Ready to install page. This page shows a tree view of installation options that were specified during Setup. On this page, Setup indicates whether the Product Update feature is enabled or disabled and the final update version:



Click **Install** button to start SQL Server 2012 installation. The Setup will first install the required prerequisites for the selected features followed by the feature installation. The Installation Progress page provides status so that you can monitor installation progress as Setup continues:



When the installation is complete, click on **Next** to Complete page:



Click on the link to review the installation log if appropriate, then click on **Close**. This finalizes the installation process for SQL Server 2012.

Best Practices on Installation:

1. Always fully document installs so that your SQL Server instances can easily be reproduced in an emergency.
2. If possible, install and configure all of your SQL Server instances consistently, following an agreed-upon organization standard.
3. Don't install SQL Server services you don't use, such as Microsoft Full-Text Search, Notification Services, or Analysis Services.
4. For best performance of SQL Server running under Windows, turn off any operating system services that aren't needed.
5. For optimum SQL Server performance, you want to dedicate your physical servers to only running a single instance of SQL Server, no other applications.
6. For best I/O performance, locate the database files (.mdf) and log files (.ldf) on separate arrays on your server to isolate potentially conflicting reads and writes.
7. If tempdb will be used heavily, also put it on its own separate array.
8. Do not install SQL Server on a domain controller.
9. Be sure that SQL Server is installed on an NTFS partition.
10. Don't use NTFS data file encryption (EFS) and compression on SQL Server database and log files.

Case Study/Practical Troubleshooting

Failed Installation/Moving System Databases

Accidently you choose program files as well as data files in the same directory, later you realized that that mistake. You are planning to un-install and reinstall the engine with specified settings. Here is the solution how to move system databases from SQL default location to another location.

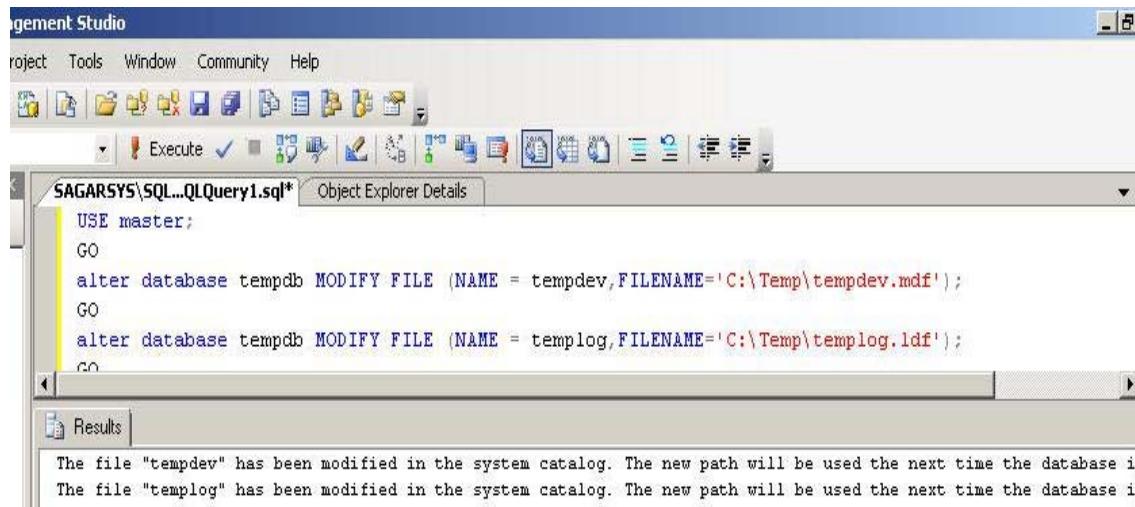
Tasks for moving system databases:

1. Moving tempdb databases.

a.) Execute the script below.

```
USE master;
GO
alter database tempdb MODIFY FILE (NAME = tempdev,FILENAME='NEW PATH');
GO
alter database tempdb MODIFY FILE (NAME = templog,FILENAME='NEW PATH');
GO
```

Example



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SAGARSYS\SQL...QLQuery1.sql*' is open, displaying the following T-SQL script:

```
USE master;
GO
alter database tempdb MODIFY FILE (NAME = tempdev,FILENAME='C:\Temp\tempdev.mdf');
GO
alter database tempdb MODIFY FILE (NAME = templog,FILENAME='C:\Temp\templog.ldf');
GO
```

Below the script, the 'Results' tab shows the output of the execution:

```
The file "tempdev" has been modified in the system catalog. The new path will be used the next time the database is started.
The file "templog" has been modified in the system catalog. The new path will be used the next time the database is started.
```

- b.) Restart services.
- c.) Confirm path of database files using the query USE tempdb; SELECT physical_name from sys.database_files.

2. Moving model and msdb databases.

- a.) Execute the script below.

```
USE master;
GO
alter database msdb MODIFY FILE (NAME = MSDBData,FILENAME='NEW PATH');
go
alter database msdb MODIFY FILE (NAME = MSDBLog,FILENAME='NEW PATH');
go
USE master;
GO
alter database model MODIFY FILE (NAME = modeldev,FILENAME='NEW PATH');
go
alter database model MODIFY FILE (NAME = modellog,FILENAME='NEW PATH');
go
```

Example

```

USE master;
GO
ALTER DATABASE msdb MODIFY FILE (NAME = MSDBData, FILENAME='C:\Temp\MSDBData.mdf');
GO
ALTER DATABASE msdb MODIFY FILE (NAME = MSDBLog, FILENAME='C:\Temp\MSDBLog.ldf');
GO
USE master;
GO
ALTER DATABASE model MODIFY FILE (NAME = modeldev, FILENAME='C:\Temp\model.mdf');
GO
ALTER DATABASE model MODIFY FILE (NAME = modellog, FILENAME='C:\Temp\modellog.ldf');
GO

```

The file "MSDBData" has been modified in the system catalog. The new path will be used the next time the database is opened.
The file "MSDBLog" has been modified in the system catalog. The new path will be used the next time the database is opened.
The file "modeldev" has been modified in the system catalog. The new path will be used the next time the database is opened.
The file "modellog" has been modified in the system catalog. The new path will be used the next time the database is opened.

- b.) Stop services
- c.) Copy the files to the new location
- d.) Restart services.
- e.) Confirm path of database files using the below query.

```

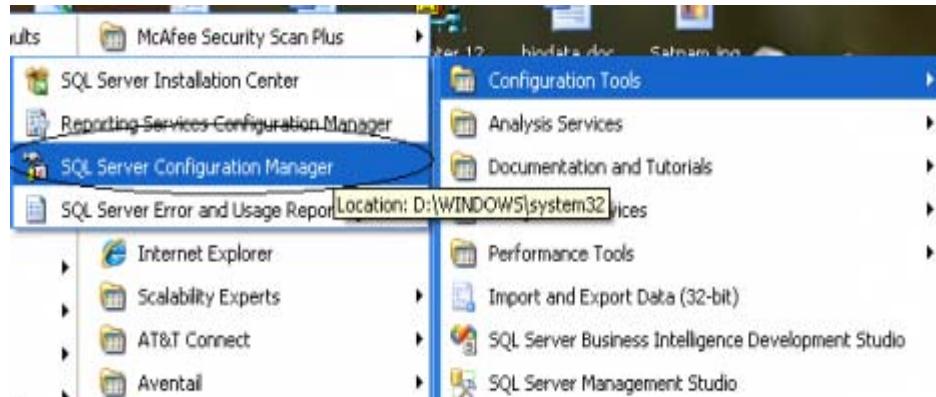
USE msdb;
SELECT physical_name from sys.database_files
USE model;
SELECT physical_name from sys.database_files

```

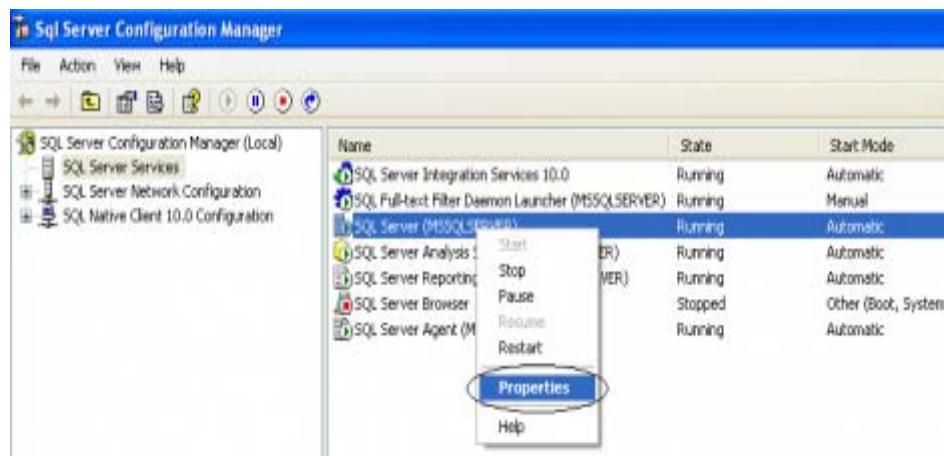
3.) Moving master database:

At present, master database is present under the following path D:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\Data and I want to move its mdf and ldf file to the paths named E:\Microsoft SQL Server\MSQL\Data and E:\Microsoft SQL Server\MSQL\Log respectively.

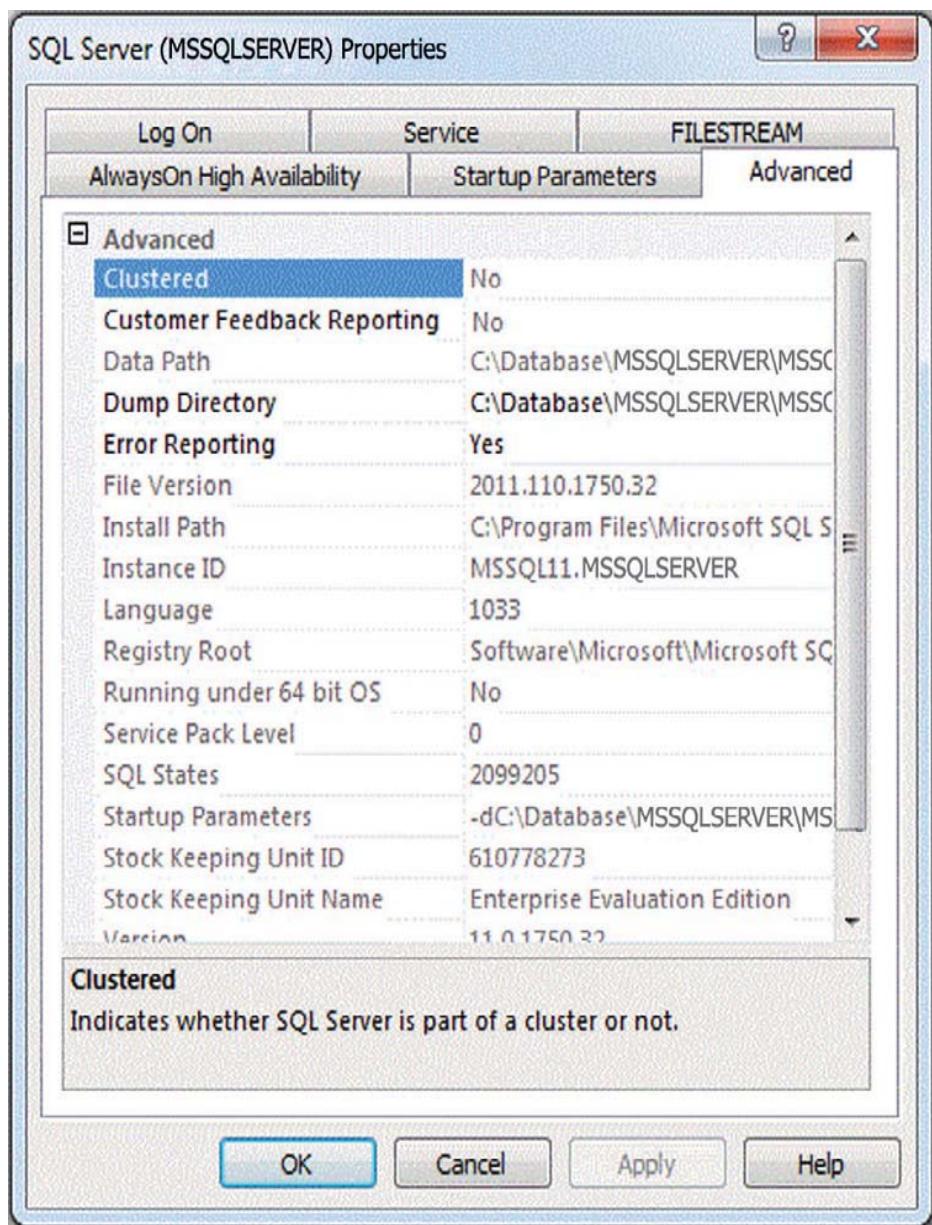
1. GoTo Start, Select All Programs, Select Microsoft SQL Server, Select Configuration Tools, Select SQL Server Configuration Manager.



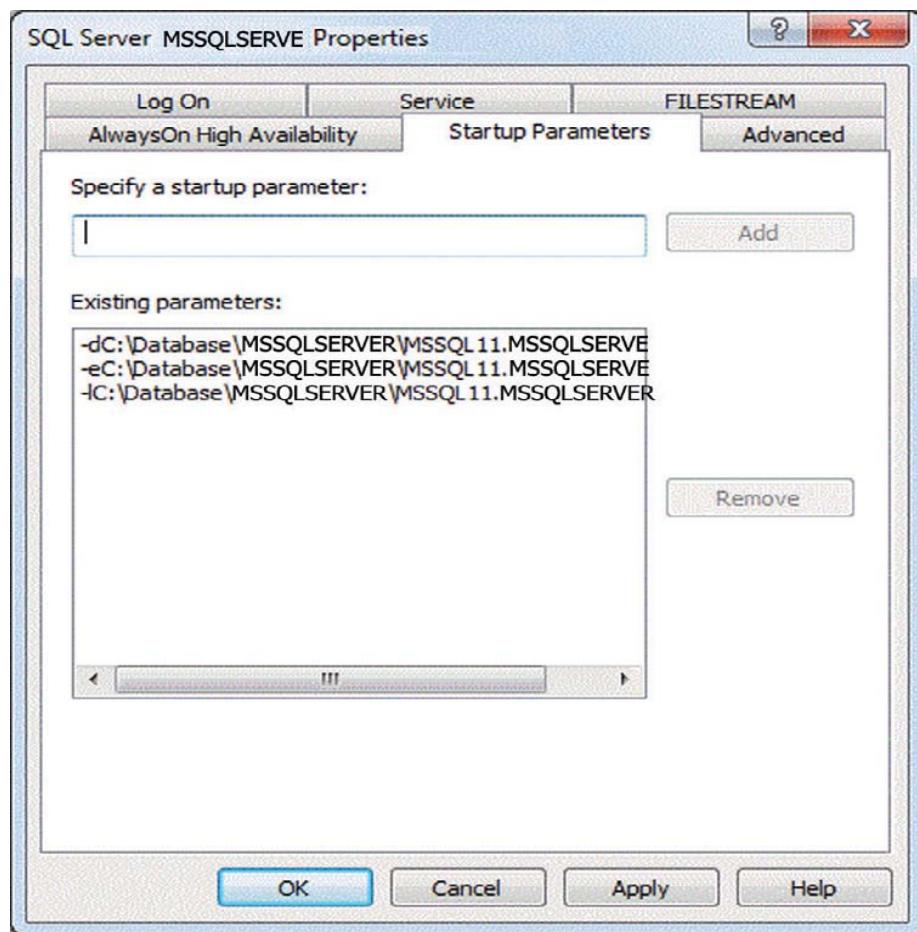
2. In the SQL Server Services node, right-click the instance of SQL Server, in our case the name of the SQL Server instance is MSSQLSERVER and choose Properties, please refer the screen capture below:



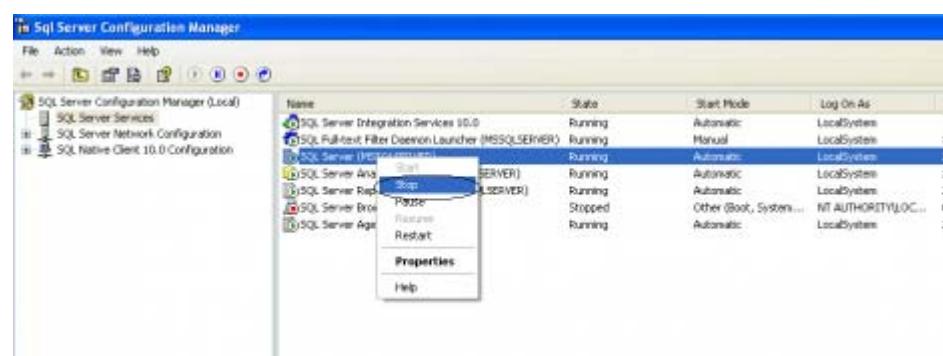
3. Click on the Advanced Tab as shown in the screen capture below:



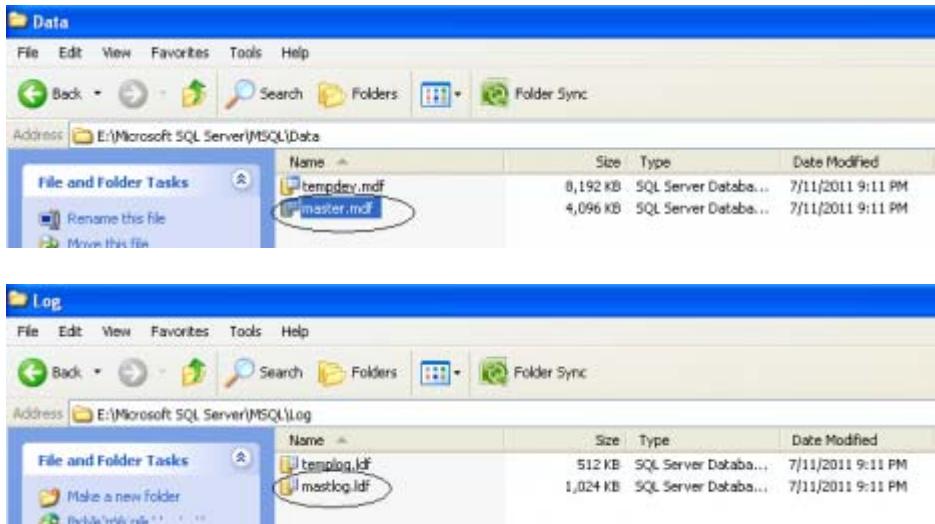
4. Select the text in the Startup parameters and change the desired path:



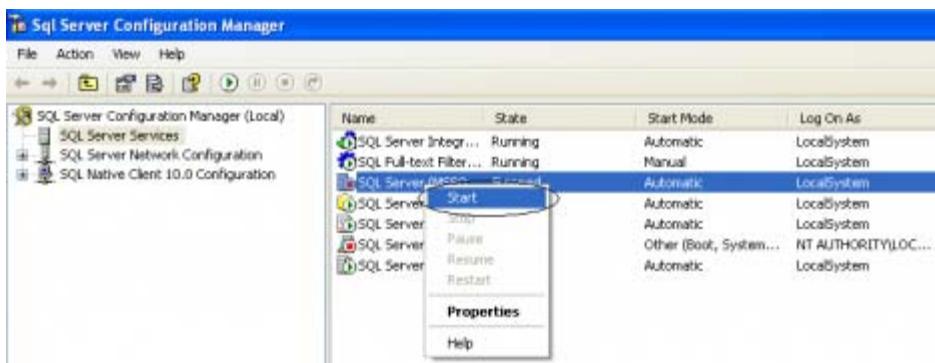
5. Select Apply and then Select OK.
6. Stop the SQL Server instance by Right Clicking on it and Selecting Stop.



7. Move the data and log file of the master database to the new location, please refer the screen capture below:



8. Restart the SQL Server instance by Right Clicking on it and Select Start; please refer the screen capture below.



Moving Resource Database:

Resource database is available from the SQL Server 2005 and higher level versions. Resource database is read only and hidden database. Resource database contains all the system objects that shipped with SQL Server. SQL Server system objects, such as **sys.objects**, are physically persisted in the Resource database, but they logically appear in the sys schema of every database.

Resource database will be very useful in doing **upgrades** or in **un-installing up-grades**. In the previous versions of SQL Server up-grade needs creating/dropping of the system objects.

From the SQL Server 2005 version upgrade is just procedure to **copying resource database file** to local server.

Name of Resource database data and log file.

mssqlsystemresource.mdf
mssqlsystemresource.ldf

Resource database data and log file location is **same as the Master database location**. In case if you are moving Master database you have to move the Resource database as well to the same location.

You can check the Resource database version and last up-grade time using the SERVERPROPERTY function.

```
SELECT SERVERPROPERTY('RESOURCEVERSION');  
SELECT SERVERPROPERTY('RESOURCELASTUPDATEDATETIME');
```

To move the resource database, you have to start the SQL Server service using either -m (single user mode) or -f (minimal configuration) and using -T3608 trace flag which will skip the recovery of all the databases other than the master database.

You can do it either from the Configuration manager or from the command prompt using below command.

Default Instance: NET START MSSQLSERVER /f /T3608

Named Instance: NET START MSSQL\$instanceName /f /T3608

Execute the below ALTER command once you have started the SQL Service by specifying the new location, location should be same as Master database location.

```
ALTER DATABASE mssqlsystemresource MODIFY FILE (NAME=data, FILENAME='\\mssqlsystemresource.mdf')
```

```
ALTER DATABASE mssqlsystemresource MODIFY FILE (NAME=log, FILENAME='\\mssqlsystemresource.ldf')
```

Chapter – 7

Upgrading the SQL server

Upgrading By applying service pack

SQL Server 2008 R2 Service Pack installation process, which differs quite a bit from SQL Server 2000 and SQL Server 2005 installations.

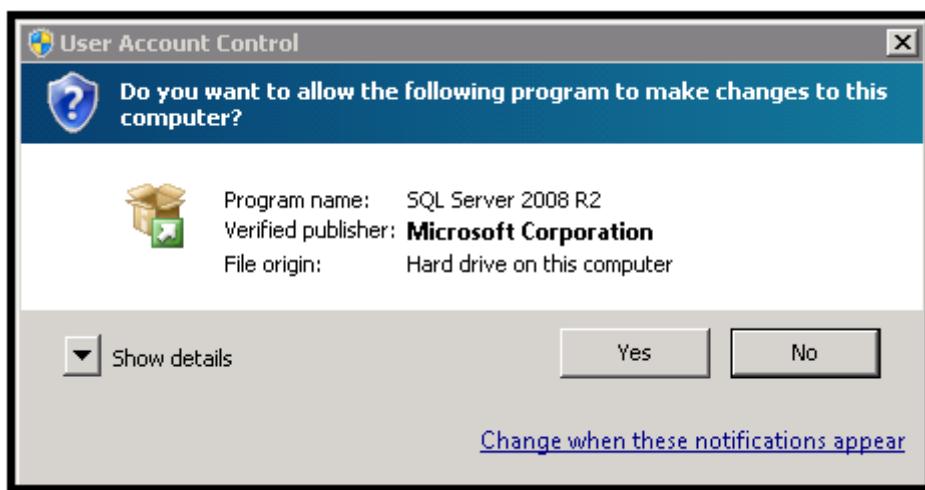
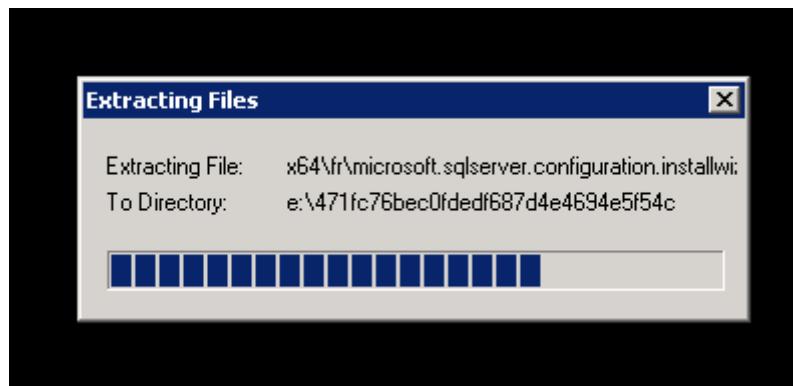
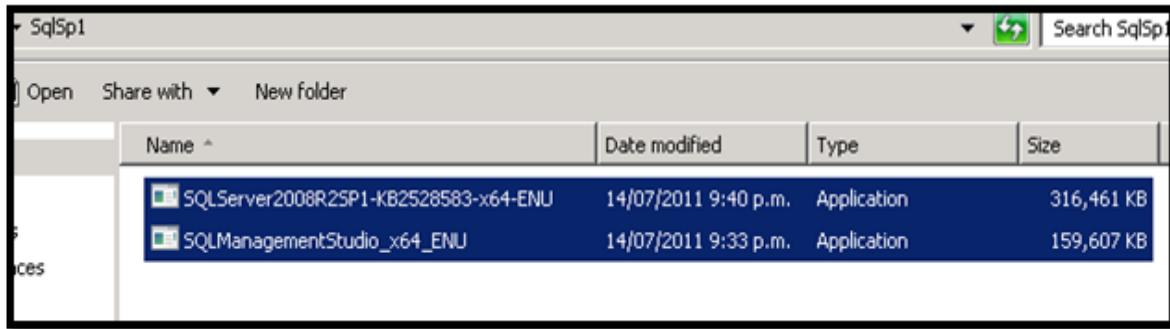
It is recommended that before applying any service pack on SQL Server, database administrators should take a full backup of all the user and system databases including the Resource database which was first introduced in SQL Server 2005. The **Resource** database is a read-only database that contains all the system objects that are included with SQL Server.

Database Administrators should read the `readme.txt` file which comes with Service Pack Installation before applying the Service Pack. However, Database Administrators should first apply the service pack on development and test servers before applying on production servers. If there are no issues reported in the development and test Environments after testing your applications, then you can apply them on the production servers. Here are the essential steps that can make your SQL Server 2008 R2 Service Pack (SP1) installation as smooth as possible:

1. Development environments:
 1. Issue a full backup of all user and system databases including the Resource database.
 2. Take note of all the Startup parameters, Memory Usage, CPU Usage etc.
 3. Install the service pack on development SQL Servers.
 4. Create a backup and restore plan with the steps for "**What to do if application is not working properly after installing the new Service Pack?**"
2. Test environments:
 1. Issue a full backup of all user and system databases including the Resource database.
 2. Take note of all the Startup parameters, Memory Usage, CPU Usage etc
 3. Install the service pack on test SQL Servers.
 4. Conduct testing for administrative process as well as coordinate testing with the Development and QA Teams to ensure the application is performing as expected.
 5. Test the rollback plan.
3. Production environments:
 1. Plan for a scheduled downtime on the Production Servers as it takes approximately 30 minutes to apply the service pack on SQL Server 2008 R2.
 2. Issue a full backup of all user and system databases including the Resource database.
 3. Take note of all the Startup parameters, Memory Usage, CPU Usage etc
 4. Install the service pack on test SQL Servers.
 5. Validate the application is working properly.

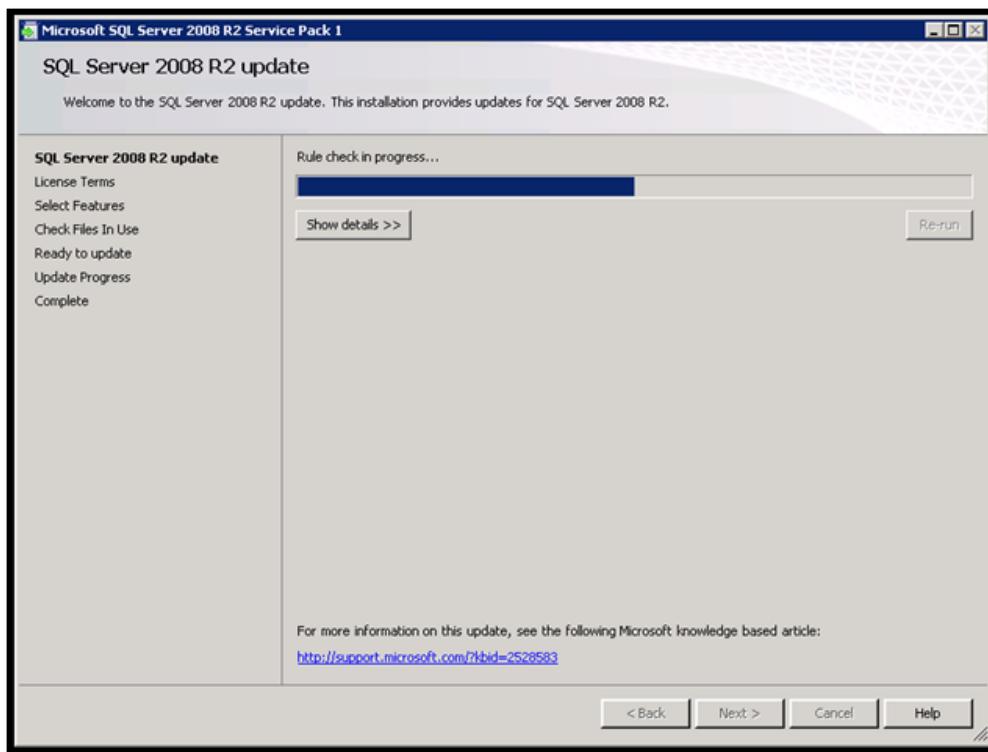
Steps to Install SQL Server 2008 R2 Service Pack 1 (SP1)

1. Download SQL Server 2008 R2 Service Pack 1 (SP1) from the following [link](#).
2. Double Click **Setup.exe** to extract the Service Pack installation files from the setup.

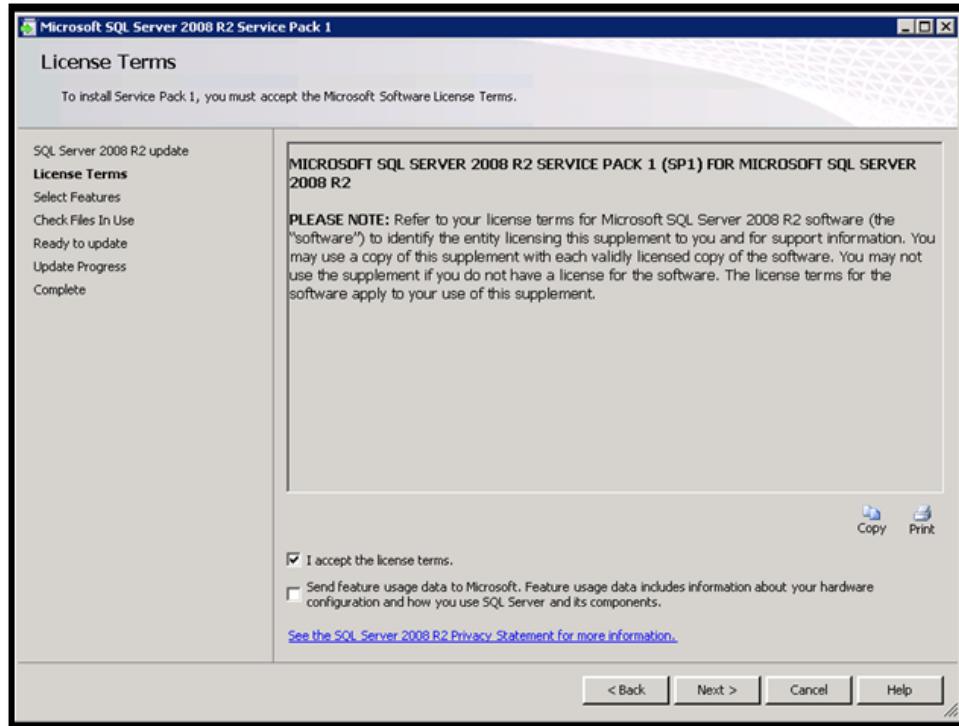




- Once setup files are extracted go to the folder where the files are extracted and click **Setup.exe**. This will open up SQL Server 2008 R2 Service Pack 1 screen as shown in the snippet below. In the **Welcome** screen the SQL Server 2008 R2 Setup program will check for few rules before applying the Service Packs. If any of the rules are failing, the installation will not continue further. Hence, you need to fix those issues before continuing with the installation.

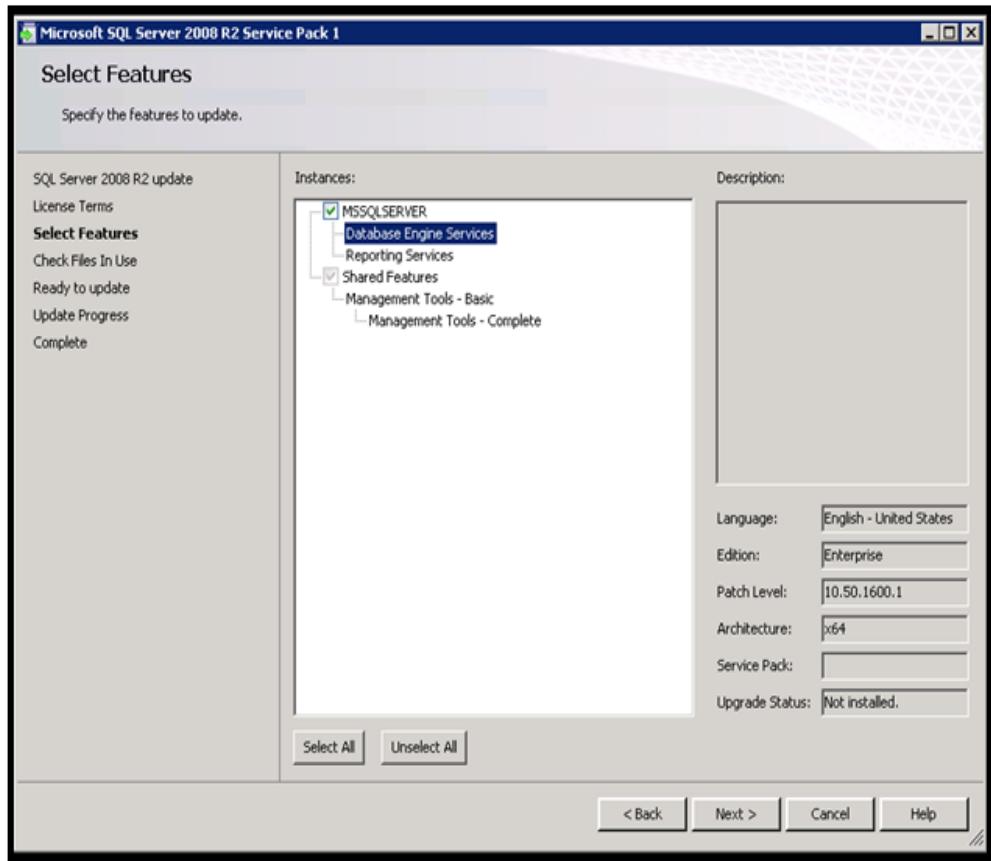


- In **License Terms** screen, read the license agreement and then select the check box at the bottom of the page to accept the licensing terms and conditions of the product. Click Next to continue with the installation.



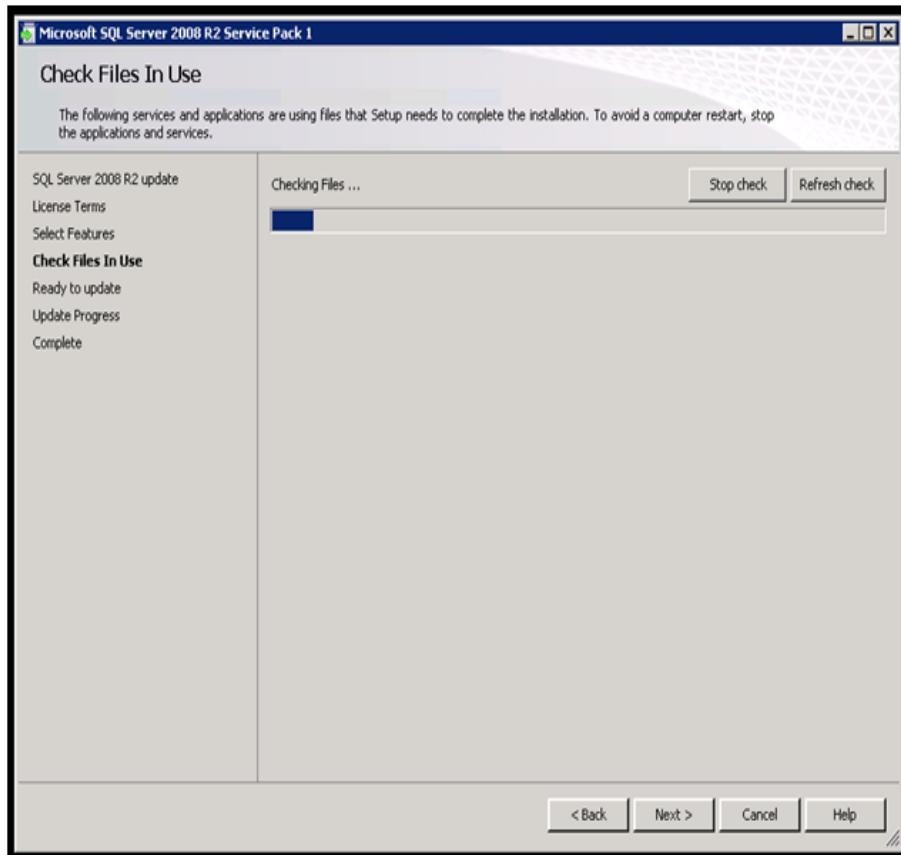
5. In **Select Features** screen, select the SQL Server Instance and select the features of SQL Server 2008 R2 which need to be upgraded. A brief description about each feature is shown in the right side panel along with the **Language**, **Edition**, **Patch Level**, **Architecture**, **Service Pack** and **Upgrade Status** when each of the features is selected. The different components which are available for upgrade within SQL Server 2008 R2 are mentioned below:

Database Engine Services
SQL Server Replication
Full-Text Replication
Analysis Services
Reporting Services
Shared Features
Business Intelligence Development Studio
Client Tools Connectivity
Integration Services
Client Tools Backwards Compatibility
Client Tools SDK
SQL Server Books Online
Management Tools – Basic
Management Tools - Complete

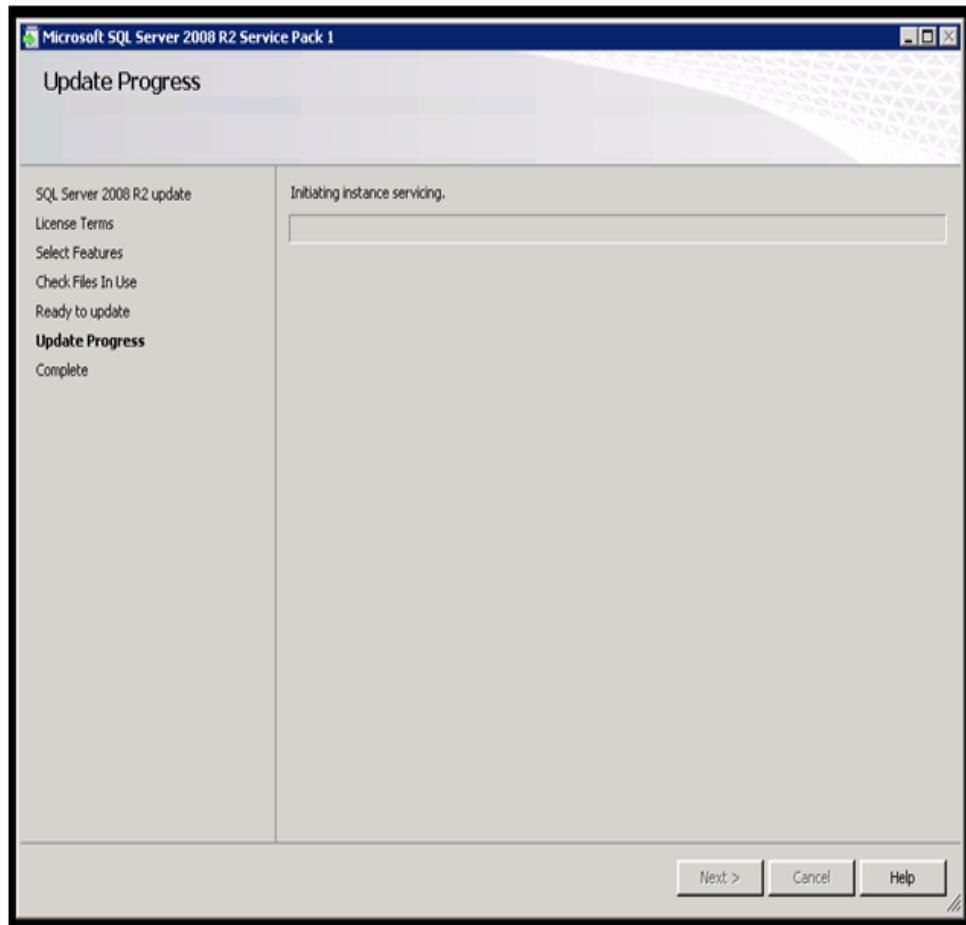


Once all the features are selected click Next to continue with the SQL Server 2008 R2 Service Pack 1 (SP1) installation.

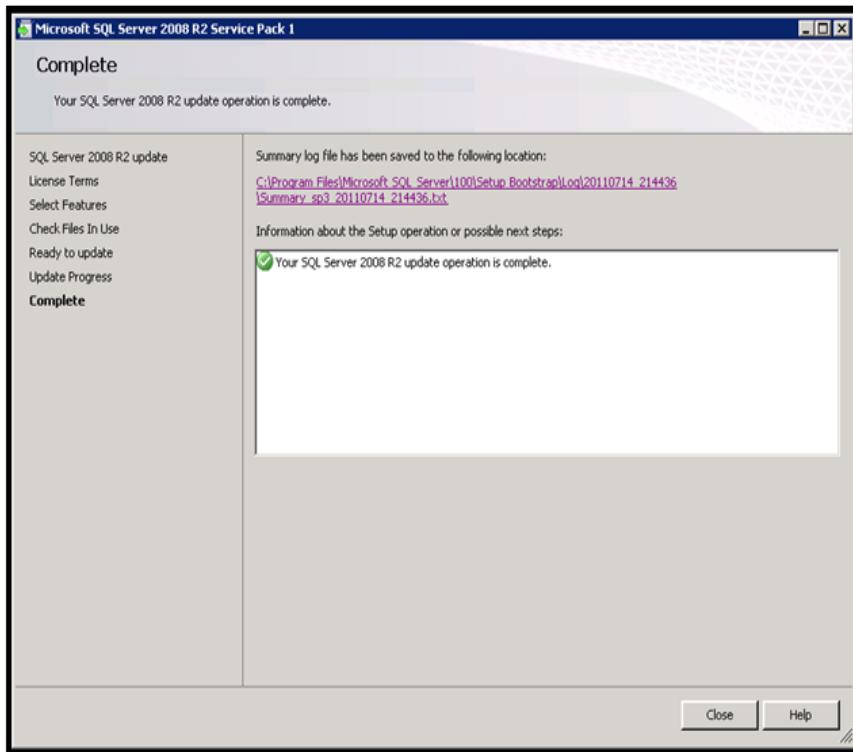
6. In **Check Files In Use** screen the setup will check for SQL Server related files which are currently being used. If there are any such files then they need to be released before performing the Service Pack installation. Click Next to continue with the installation.



7. In **Ready to Upgrade** screen you can verify the list of features which will be upgraded during the installation. If you are fine with the list of features, then click **Upgrade** to perform the actual Service Pack installation.
8. In **Upgrade Progress** screen you will be able to see the installation progress. It will approximately take 30 minutes to complete the installation. The installation time varies depending upon the hardware configuration.



9. Once the Service Pack installation is successful you will be able to see the **Success** message as shown in the snippet below.
10. On the **Complete** screen you will be able to see "**Your SQL Server 2008 R2 update operation has completed successfully**" message. Click Close to end the installation setup.



11. You can verify the SQL Server 2008 R2 Service Pack 1 (SP1) installation by executing the query below in SQL Server Management Studio (SSMS).

```
SELECT SERVERPROPERTY('Edition') AS 'Edition', SERVERPROPERTY('ProductVersion') AS 'ProductVersion', SERVERPROPERTY('ProductLevel') AS 'ProductLevel', SERVERPROPERTY('ResourceLastUpdateDateTime') AS 'ResourceLastUpdateDateTime', SERVERPROPERTY('ResourceVersion') AS 'ResourceVersion'
```

UPGRADING TO SQL SERVER 2012

A smooth upgrade requires a good plan. When you devise an upgrade plan, you need to break down the upgrade process into individual tasks. This plan should have sections for pre-upgrade tasks, upgrade tasks, and postupgrade tasks:

Your ***pre-upgrade tasks*** consider SQL Server 2012 minimum hardware and software requirements. You should have an inventory of your applications that access the server, database-collation requirements, server dependencies, and legacy-systems requirements such as data-access methods. Your list should include database consistency checks and backup of all databases. Plans should be in place for testing the upgrade process and applications. You should have a thorough understanding of backward-compatibility issues and identify

workarounds or fixes. You should also use the SQL Server 2012 Upgrade Advisor, as described later in this chapter, to assist in identifying and resolving these issues. The *upgrade execution process* is a smooth execution of your well-documented and rehearsed plan. To reiterate the importance of this step, ensure you make a backup of all the databases before you execute the upgrade process.

Post-upgrade tasks consist of reviewing the upgrade process, bringing the systems back online, monitoring, and testing the system. You need to perform specific database maintenance before releasing the system to the user community. These and other recommended steps are outlined later in the chapter. Run your database in backward compatibility mode after the upgrade to minimize the amount of change to your environment. Update the database-compatibility mode as part of a follow-up upgrade process and enable new SQL Server 2012 features. As part of deciding your upgrade strategy, consider both in-place (upgrade) and side-by-side migration methods for upgrading.

In-Place Upgrading

The in-place server upgrade is the easier but riskier of the two options. This is an all-or-nothing approach to upgrading; meaning that after you initiate the upgrade there is no simple rollback procedure. This type of upgrade has the added requirement of greater upfront testing to avoid using a complex back-out plan. The benefit of this approach is that you don't need to worry about users and logins remaining in sync, and database connectivity changes are not required for applications. In addition, SQL Server Agent jobs migrate during the upgrade process.

Following is a high-level scenario of an in-place upgrade based on the below Figure:

1. First, install the prerequisite files on your system. Before upgrading to SQL Server 2012, your server needs, at a minimum, the following:

.NET Framework 4.0

Windows PowerShell 2.0

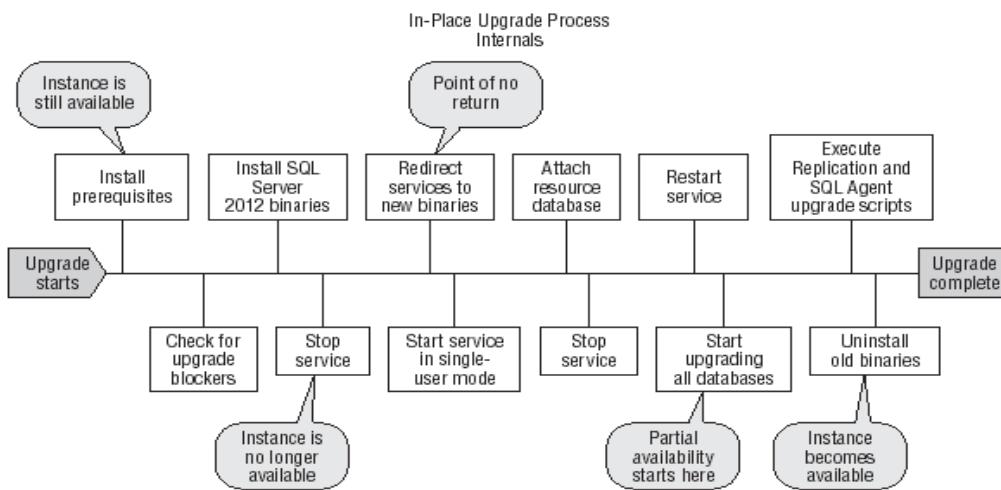
.NET 3.5 with Service Pack 1

A current instance of SQL Server 2005, SQL Server 2008, or SQL Server 2008 R2

2. Next run the System Configuration Checker (SCC). The SCC examines the destination computer for conditions that would prevent an upgrade from completing, such as not meeting the minimum hardware or software requirements. If such a condition is found, setup aborts and the SQL Server 2012 components uninstall.
3. Once verified, the SQL Server setup program can lay the 2012 bits and backward-compatibility support files on a disk while SQL Server 2008 (or 2005) is still available to users.

However, don't plan to upgrade a server while users are online. The setup program takes the server offline by stopping the existing SQL Server services. The 2012-based services assume control of the master database and the server identity. At this point, the SQL Server service takes over the databases and begins to update them while not allowing users back into the environment. When a request for data occurs in a database that has been only partially updated, the data associated with this request is updated, processed, and then returned to the user.

4. Finally, kick off the uninstall procedure for the old binaries. This step occurs only if no remaining SQL Server 2005 or 2008 instances are on the server. SQL Server Agent jobs are now migrated.



Following are the advantages of an in-place upgrade:

- Fast, easy, and automated (best for small systems).
- No additional hardware required.
- Applications retain same instance name.
- Preserves SQL Server 2008 (or 2005) functionality automatically.

The disadvantages of an in-place upgrade are as follows:

- Downtime incurred because the entire SQL Server instance is off-line during upgrade.
- No support for component-level upgrades.
- Complex rollback strategy.
- Backward-compatibility issues must be addressed for that SQL instance.
- In-place upgrade is not supported for all SQL Server components.
- Large databases require substantial rollback time.

Additionally, if you would like to change editions as a part of your upgrade, you must be aware of some limitations. You can upgrade SQL Server 2005 and 2008/R2 Enterprise, Developer, Standard, and Workgroup editions to different editions of SQL Server 2012. However, SQL Server 2005 and 2008 Express Editions may only be upgraded to SQL Server 2012 Express Edition. If this is of interest to you, see SQL Server 2012 Books Online (BOL), "Version and Edition Upgrades," under the section "Upgrading to SQL Server 2012."

Side-by-Side Upgrade

In a side-by-side upgrade, SQL Server 2012 installs either along with SQL Server 2008 (or 2005) as a separate instance or on a different server. This process is essentially a new installation followed by a database migration. You may want to select this option as part of a hardware refresh or migration to a new platform, such as Itanium or x64. Because of the backup and restore times involved in a back-out scenario, if you have a sizable database, this is definitely the option to use.

As part of this method, you can simply back up the databases from the original server and then restore them to the SQL Server 2012 instance. Other options are to manually detach your database from the old instance and reattach it to the new instance, use log shipping, or database mirroring. You can also leverage the Copy Database Wizard to migrate your databases to the new server. Although this approach provides for a good recovery scenario, it has additional requirements beyond those of the in-place upgrade, such as maintaining the original server name, caring for application connectivity, and keeping users and their logins in sync.

Following are the arguments in favor of a side-by-side upgrade:

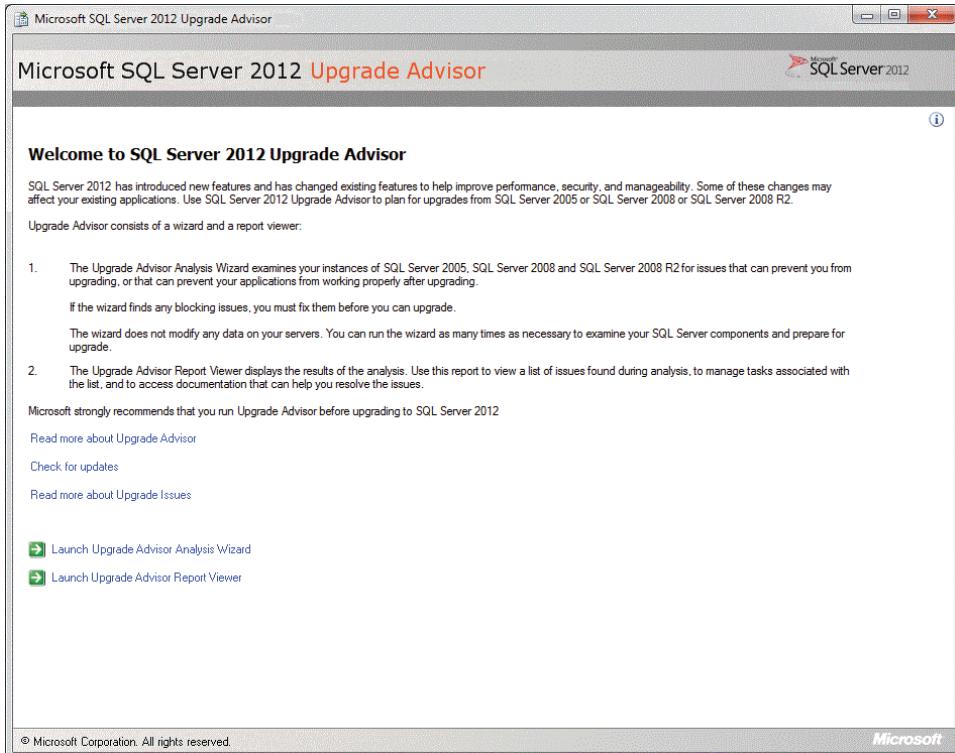
- More granular control over upgrade component-level process (database, Analysis Services, and others)
- Ability to run SQL Servers side-by-side for testing and verification
- Ability to gather real matrix for upgrade (outage window)
- Rollback strategy because original server is still intact
- Best for large databases because restore time could be sizable

The arguments against a side-by-side upgrade are as follows:

- Does not preserve SQL Server 2008 (or 2005) functionality.
- Issue of instance name for connecting applications.

SQL Server Upgrade Advisor

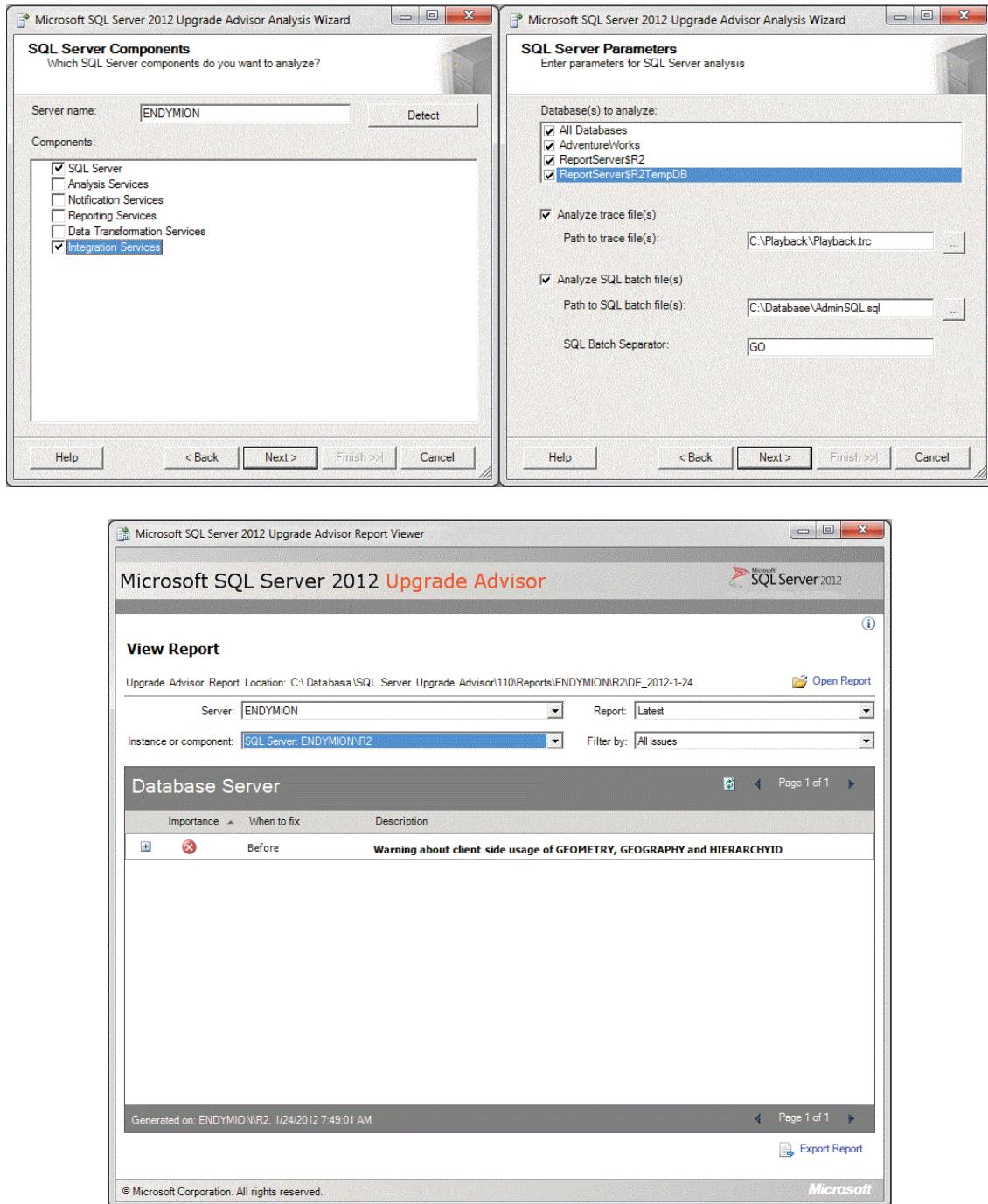
The rules checked by Upgrade Advisor represent conditions, situations, or known errors that might affect your upgrade to SQL Server 2012. If you want to take advantage of the lessons other SQL Server users have learned about upgrading, the SQL Server 2012 Upgrade Advisor is the tool for you.



Using the Upgrade Advisor

When installed, the Upgrade Advisor presents you with two choices, Upgrade Advisor Analysis Wizard and Upgrade Advisor Report Viewer. Launch the Upgrade Advisor Analysis Wizard to run the tool. As shown in Figure, you simply select a server and the components to analyze for upgrade, or you can click the Detect button, which starts the inspection process that selects the components installed on your system.

After you select the components for testing, the next decision is to select the databases that you would like to have evaluated for upgrade, as shown in Figure. The best part of this process is that you have the option to analyze SQL Profiler trace and SQL batch files to help make this a comprehensive analysis. That is, by adding these files to the evaluation process, Upgrade Advisor evaluates not only the database but its trace workload and SQL scripts as well. By evaluating this additional information, Upgrade Advisor evaluates not only the database as it exists right now, but also information about past database usage and behavior contained in the trace and batch files. All you need to do is select the path to the directory where your trace files or your batch files are located.



When the test completes, you can view the discovered issues via the Upgrade Advisor Report Viewer.

In-Place Upgrade to SQL Server 2012

This is probably the easiest of all of the upgrade strategies. As long as the computer meets the minimum system requirements, setup will allow you to upgrade a SQL Server installation in-place. The instance name is preserved so you don't have to update the connection strings of dependent applications.. You can only upgrade to the same or higher edition using this approach.

In-place upgrade is only supported for the following versions of SQL Server (note minimum required service pack level):

- SQL Server 2005 (SP4)
- SQL Server 2008 (SP2)
- SQL Server 2008 R2

If you are running SQL Server 2000, you will need to upgrade to SQL Server 2008 R2 first before attempting to upgrade to SQL Server 2012.

Note that you cannot change platforms as part of your in-place upgrade. For example, if you are currently running SQL Server 2008 R2 (x86) on Windows Server 2008 R2 (x64), you cannot perform an in-place upgrade to SQL Server 2012 (x64). You can only upgrade in-place to SQL Server 2012 (x86).

The disadvantage of this approach is that there is no going back short of uninstalling SQL Server 2012 and re-installing your previous edition or restoring the whole server from a backup, so this is probably only suitable for servers that don't affect business operations in any substantial way.

1. Insert the SQL Server installation media, and from the root folder, double-click setup.exe. To install from a network share, move to the root folder on the share, and then double-click setup.exe. If the Microsoft SQL Server 2012 Setup dialog box appears, click **OK** to install the prerequisites, then click **Cancel** to exit SQL Server 2012 installation.
2. Windows Installer 4.5 is also required, and may be installed by the Installation Wizard. If you are prompted to restart your computer, restart, and then run SQL Server 2012 Setup.exe again.
3. When prerequisites are installed, the Installation Wizard will start the SQL Server Installation Center. To upgrade an existing instance of SQL Server 2012, click **Upgrade from SQL Server 2005, 2008 & 2008 R2**.
4. If Setup support files are required, SQL Server Setup will install them. If you are instructed to restart your computer, restart before you continue.
5. The System Configuration Checker will run a discovery operation on your computer. To continue, click **OK**. Setup log files are created for your installation.

6. On the Product key page, click an option button to indicate whether you are upgrading to a free edition of SQL Server, or whether you have a PID key for a production version of the product.
7. On the License Terms page, read the license agreement, and then select the check box to accept the license terms and conditions. **Click Next to continue.** To end Setup, click **Cancel**.
8. On the Select Instance page, specify the instance of SQL Server to upgrade.
9. On the Feature Selection page, the features to upgrade will be preselected. A description for each component group appears in the right pane after you select the feature name. Be aware that you cannot change the features to be upgraded, and you cannot add features during the upgrade operation. To add features to an upgraded instance of SQL Server 2012 after the upgrade operation is complete.
10. On the Instance Configuration page, specify whether to install a default or a named instance.

Instance ID — By default, the instance name is used as the Instance ID. This is used to identify installation directories and registry keys for your instance of SQL Server. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be MSSQLSERVER. To use a nondefault instance ID, select the **Instance ID** check box and provide a value.

Instance root directory — By default, the instance root directory is C:\Program Files\Microsoft SQL Server\1. To specify a nondefault root directory, use the field provided, or click **Browse** to locate an installation folder.

All SQL Server service packs and upgrades will apply to every component of an instance of SQL Server.

Detected instances and features — The grid will show instances of SQL Server that are on the computer where Setup is running. If a default instance is already installed on the computer, you must install a named instance of SQL Server 2012. **Click Next to continue.**

11. The Disk Space Requirements page calculates the required disk space for the features that you specify, and compares requirements to the available disk space on the computer where Setup is running.
12. Work flow for the rest of this topic depends on the features that you have specified for your installation. You might not see all the pages, depending on your selections.
13. On the Server Configuration — Service Accounts page, specify login accounts for SQL Server services. The actual services that are configured on this page depend on the features that you are upgrading.

Authentication and login information will be carried forward from the previous instance of SQL Server. You can assign the same login account to all SQL Server services, or you can configure each service account individually. You can also specify whether services start automatically, are started manually, or are disabled. Microsoft recommends that

you configure service accounts individually so that SQL Server services are granted the minimum permissions they have to have to complete their tasks.

To specify the same login account for all service accounts in this instance of SQL Server, provide credentials in the fields at the bottom of the page.

Security Note Do not use a blank password. Use a strong password.

When you are finished specifying login information for SQL Server services, click **Next**.

14. Use the **Server Configuration — Collation** tab to specify nondefault collations for the Database Engine and Analysis Services.
15. On the Full-Text Search Upgrade Options page, specify the upgrade options for the databases being upgraded.
16. On the **Error and Usage Reporting** page, specify the information that you want to send to Microsoft that will help improve SQL Server. By default, options for error reporting and feature usage are enabled.
17. The System Configuration Checker will run one more set of rules to validate your computer configuration with the SQL Server features that you have specified before the upgrade operation begins.
18. The Ready to Upgrade page displays a tree view of installation options that were specified during Setup. To continue, click **Install**.
19. During installation, the progress page provides status so that you can monitor installation progress as Setup continues.
20. After installation, the Complete page provides a link to the summary log file for the installation and other important notes. To complete the SQL Server installation process, click **Close**.
21. If you are instructed to restart the computer, do so now. It is important to read the message from the Installation Wizard when you have finished with Setup.

Best Practices while Upgrading:

1. Run the Upgrade Advisor before upgrading. Make any necessary changes before performing the upgrade.
2. Perform a test upgrade of your test SQL Servers before you upgrade your production servers. And don't forget to test your applications with the new version also.
3. Before you upgrade, be sure you have a plan in place to fall back to in case the upgrade is problematic.
4. Don't upgrade SQL Server clusters in place. Instead, rebuild them on new hardware.
5. If you upgrade from a previous version of SQL Server, you should update all of the statistics in all your databases using either UPDATE STATISTICS or sp_updatestats. This is because statistics are not automatically updated during the upgrade process.

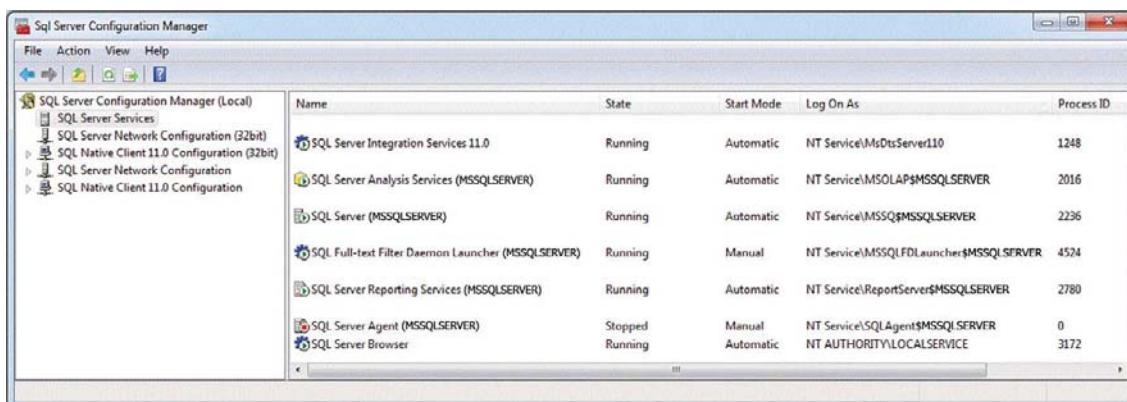
Chapter – 8

Configuring SQL Server

After you install SQL Server or upgrade to SQL Server 2012, you will likely need to configure it for your needs. In SQL Server 2012, Microsoft maintained its policy to increase out-of-the-box security established with SQL Server 2008 by turning off features after installation, thereby reducing the software footprint. The features turned off vary based on the edition of SQL Server. For example, TCP/IP is disabled in Developer Edition by default, and every edition has CLR integration turned off. This makes the environment more usable for you as an administrator because features you don't care about are not crowding your administration screen. It also reduces the options that a hacker or, more likely, a malicious user can use to penetrate your system. As a result of these security measures, there is even more reason to invest time in configuring SQL Server 2012 towards your specific needs. In this section, you learn how to configure SQL Server for your specific environment and security needs in a few ways: by using SQL Server Configuration Manager, startup parameters, startup stored procedures, and partially contained databases. SQL Server Configuration Manager is the best place to start.

SQL Server Configuration Manager

The SQL Server Configuration Manager configures the SQL Server services much like the Services applet in the Control Panel, but it has much more functionality than the applet. For example, the program can also change what ports SQL Server listens on and what protocols each instance uses. You can open the program from Start → All Programs → Microsoft SQL Server 2012 → Configuration Tools → SQL Server Configuration Manager.



Network Protocols from SQL Server configuration manager

SQL Server 2012 is a client-server application designed to efficiently exchange data and instructions over one or more network connections.

SQL Server 2012 Network Protocols

SQL Server 2012 provides support for three protocols:

- Shared Memory
- TCP/IP
- Named Pipes

By default, the only network protocols enabled for most editions of SQL Server are TCP/IP and Shared Memory. The Developer and Enterprise Evaluation editions are configured with all protocols except Shared Memory disabled during installation, but the remaining protocols can be enabled if required. If a protocol is not enabled, SQL Server will not listen on an endpoint that is configured to utilize that protocol.

The SQL Server Configuration Manager is used to configure server protocols.

Shared Memory

The Shared Memory protocol can only be used by local connections, because it is a shared memory and process space used for inter-server communication. It has only one configurable property: Enabled. The Enabled property can be set to Yes or No, resulting in a status of Enabled or Disabled.

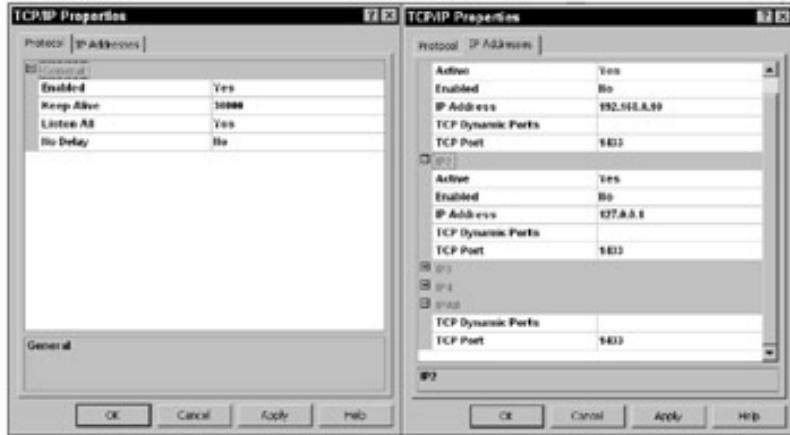
Named Pipes

Named Pipes uses Inter-Process Communication (IPC) channels for efficient inter-server communication, as well as local area network (LAN) communication. The Named Pipes protocol has some enhancements in SQL Server 2012 including support for encrypted traffic, but because of the excessive overhead of Named Pipes when connecting across networks or firewalls, and the additional port that Named Pipes requires to be opened (445), it is generally a good idea to leave the Named Pipes protocol disabled. However, there are many applications that take advantage of the Named Pipes protocol because they were designed for local network implementations. Named Pipes provides easy access to Remote Procedure Calls (RPC) within a single security domain, and so is advantageous to these applications. If you need to support one of these applications, and the SQL Server is not exposed to external traffic, the risk of enabling the Named Pipes protocol and corresponding endpoint is minimal.

Named Pipes has two configurable properties: Enabled and Pipe Name. The Enabled property works the same as the Shared Memory protocol. The Pipe Name specifies the inter-process pipe that SQL Server will listen on. The default pipe is `\.\pipe\sql\query`.

TCP/IP

The TCP/IP protocol is the primary and preferred protocol for most SQL Server installations. It is configured on two separate tabs on the TCP/IP Properties window: the Protocol tab and the IP Addresses tab, as shown



The Protocol tab has the following four configurable properties:

- Enabled — This works the same as the other protocols.
- Keep Alive — This specifies how many milliseconds SQL Server waits to verify an idle connection is still valid by sending a KEEPALIVE packet. The default is 30,000 milliseconds.
- Listen All — This specifies whether SQL Server will listen on all IP addresses configured on the server.
- No Delay — This option specifies whether the TCP protocol queues small packets to send out larger packets. This queuing is typically undesirable in transaction-based systems, and so it should be left in its default configuration of No.

The Dedicated Administrator Connection

The DAC is a specialized diagnostic connection that can be used when standard connections to the server are not possible. When you need to connect to the server to diagnose and troubleshoot problems, the DAC is an invaluable administration tool to have.

In SQL Server 2005, Microsoft introduced a new feature called Dedicated Administrator Connection (DAC). Using this feature a SQL Server Database Administrator can connect to a SQL Server Instance when the database engine is not responding to regular connections. During such a scenario a DBA can connect to the SQL Server Instance to troubleshoot and to kill any of the SQL Server Processes which are causing the issues.

The DAC allows database administrators to connect to a SQL Server Instance and to execute T-SQL commands to troubleshoot and fix issues rather than rebooting the SQL Server which could lead to database corruption or other problems. By default, the remote Dedicated Administrator Connection feature is disabled in SQL Server 2005 and later versions. It's a good practice to enable the DAC feature once the SQL Server 2008 or SQL Server 2012 is installed on every instance as this will help you troubleshoot issues when regular connections are not responding. However, only one dedicated administrator connection is allowed at a time on SQL Server 2005 and later versions.

Enable Dedicated Administrator Connection in SQL Server 2012 Using TSQL

Execute the below T-SQL to enable remote clients to utilize the Dedicated Administrator Connection.

```
Use master  
GO  
sp_configure 'show advanced options' , 1  
GO  
/* 0 = Allow Local Connection, 1 = Allow Remote Connections*/  
sp_configure 'remote admin connections', 1  
GO  
RECONFIGURE  
GO
```

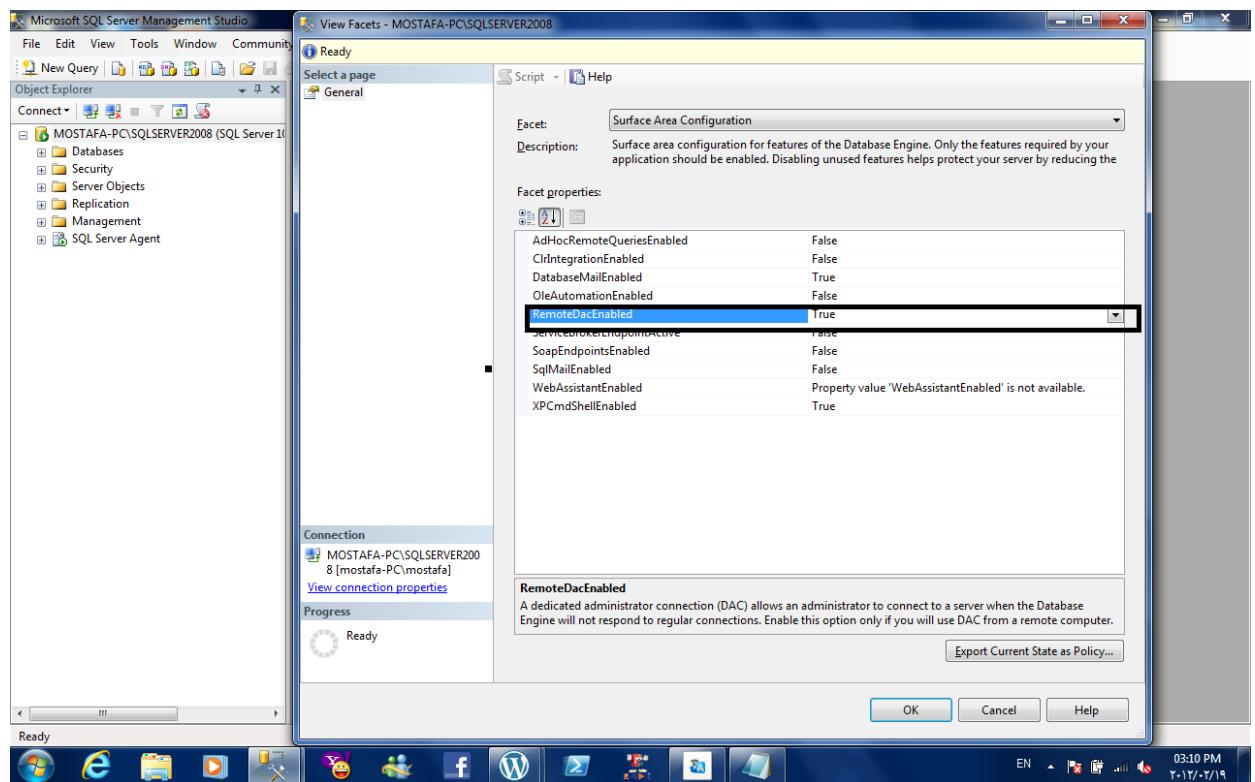
Enable Dedicated Administrator Connection in SQL Server 2012 Using SQL Server 2012 Management Studio

Database Administrators can also enable Dedicated Administrator Connection Feature using SQL Server 2012 Management Studio. This can be done by right clicking the SQL Server Instance and selecting the Facets option from the drop down list as shown in the snippet below.

Right click on your SQL Server Properties >> Facets >> Surface Area Configuration >> RemoteDacEnabled → Change it from False to True

This will open up View Facets window as shown in the snippet below. Here you need to select Surface Area Configuration facet as highlighted and then select the option as “True” for RemoteDacEnabled.

Finally, click OK to save the configuration changes in the View Facets window.

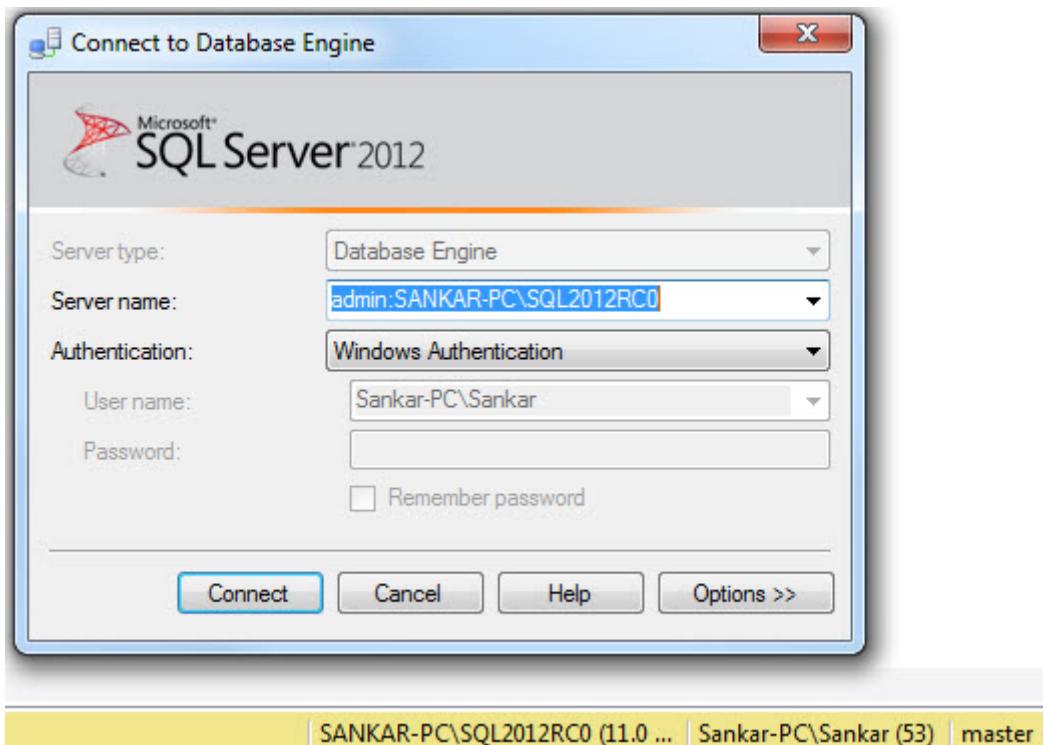
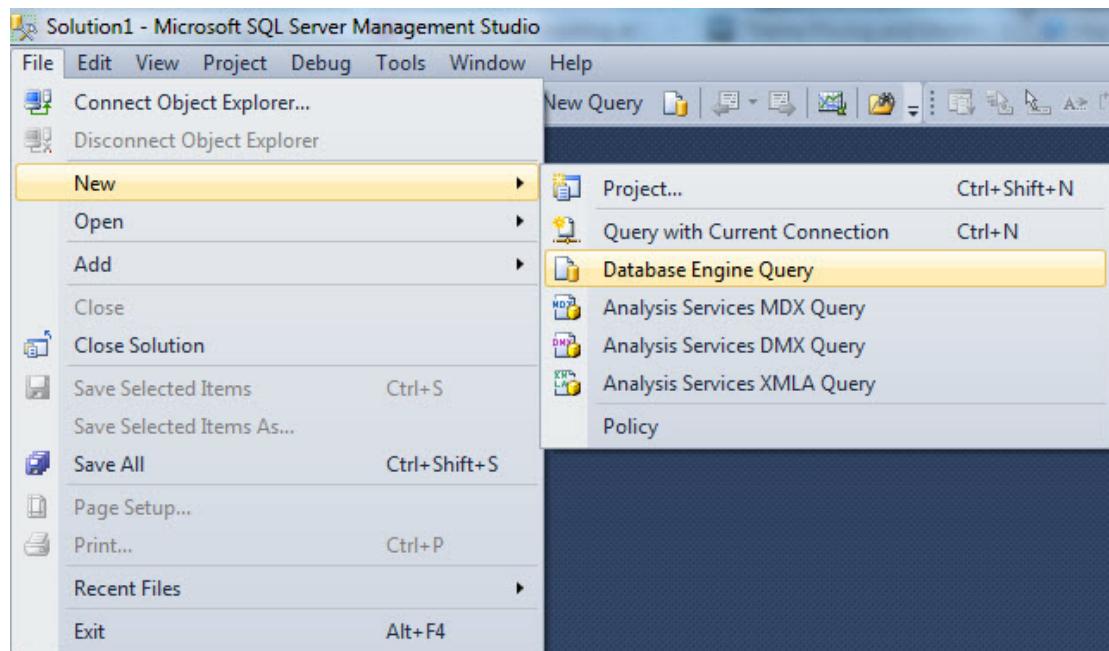


Once the Dedicated Administrator Connection is enabled you can connect to SQL Server 2012 using either SQL Server Management Studio or using SQLCMD.

Using DAC with SQL Server Management Studio

You need to specify "ADMIN:" before the SQL Server Instance name when trying to connect to an SQL Server Instance to using DAC feature as shown in the snippet below.

Once you are connected to SQL Server Instance using DAC, then you can execute code such as the code below to check the SQL Server health.



```
-- Locking Information
SELECT * FROM sys.dm_tran_locks
GO

-- Cache Status
SELECT * FROM sys.dm_os_memory_cache_counters
GO

-- Active Sessions
SELECT * FROM sys.dm_exec_sessions
GO

-- Requests Status
SELECT * FROM sys.dm_exec_requests
GO
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A script window titled 'SQLQuery1.sql ...ster (sa (51))' is open, displaying the T-SQL code provided above. Below the script window, there are three result panes:

- Results pane:** Shows the output of the 'SELECT * FROM sys.dm_os_memory_cache_counters' query. The results are as follows:

cache_address	name	type	single_pages_kb	multi_pages_kb	single_pages_in_kb
1 0x00C46048	SOS_StackFramesStore	CACHESTORE_STACKFRAMES	0	8	8
2 0x0497A598	EventNotificationCache	CACHESTORE_EVENTS	16	0	0
3 0x0497A960	Object Plans	CACHESTORE_OBJCP	2584	48	0
4 0x0497AD28	SQL Plans	CACHESTORE_SQLCP	79496	296	312

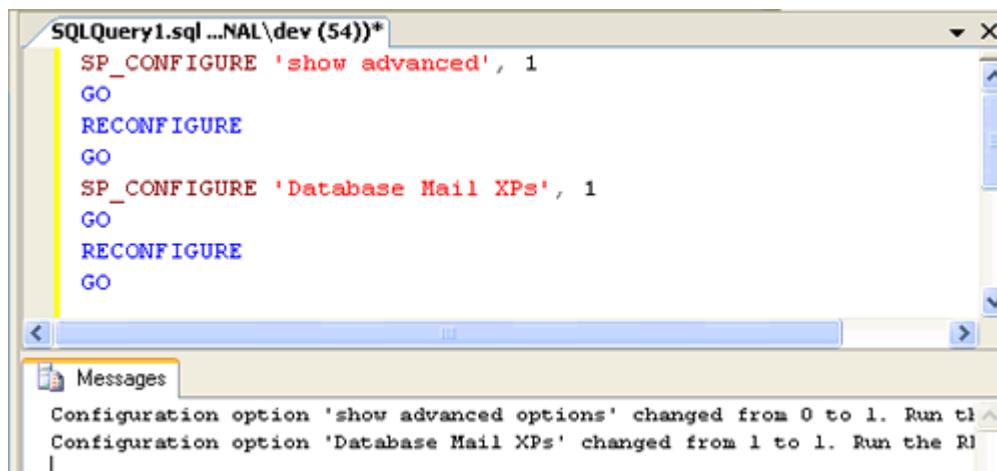
- Messages pane:** Displays the message "Query executed successfully."
- Status bar:** Shows the session details: ADMIN:AKMEHTA(SQL2008 (10.0...), sa (51), master, 00:00:00, 229 rows).

Configuring Database Mail

Database Mail appeared with SQL Server 2005 and was a welcome replacement for SQLMail; Database Mail and SQLMail both enable you to notify operators via e-mail and to send e-mails via stored procedures, but Database Mail is more secure, more reliable, and does not rely on MAPI. It uses Simple Mail Transfer Protocol (SMTP). It is cluster-aware, and enables automatic retry of failed e-mail messages and failover to another SMTP server should the first become unavailable. Database Mail also enables you to set up multiple accounts and provide secured or public access to the accounts.

For security reasons, Database Mail is disabled by default. You can enable and configure it by running the Database Mail Configuration Wizard, or running the Database Mail XPs.

```
sp_CONFIGURE'show advanced',1  
GO  
RECONFIGURE  
GO  
sp_CONFIGURE'Database Mail XPs',1  
GO  
RECONFIGURE  
GO
```



```
SQLQuery1.sql ...NAL\dev (54)*  
SP_CONFIGURE 'show advanced', 1  
GO  
RECONFIGURE  
GO  
SP_CONFIGURE 'Database Mail XPs', 1  
GO  
RECONFIGURE  
GO
```

Messages

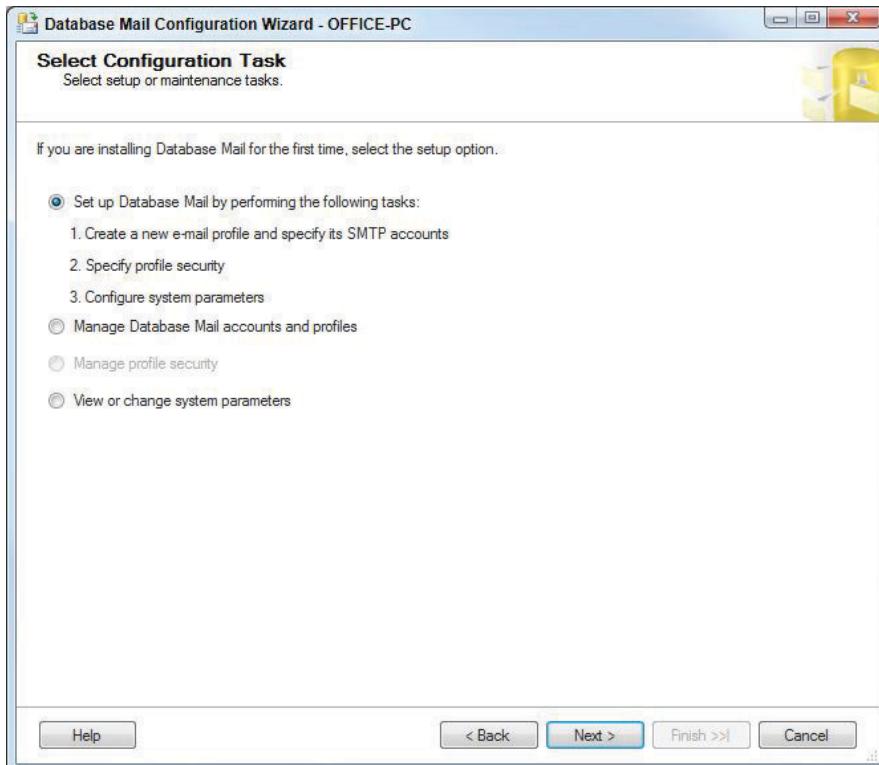
```
Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE option to complete the change.  
Configuration option 'Database Mail XPs' changed from 1 to 1. Run the RECONFIGURE option to complete the change.
```

To send Database Mail, you must either be a member of the sysadmin fixed-server role or be a member of the DatabaseMailUserRole in msdb. You can place a size limit on mail attachments and prohibit attachments with certain file extensions.

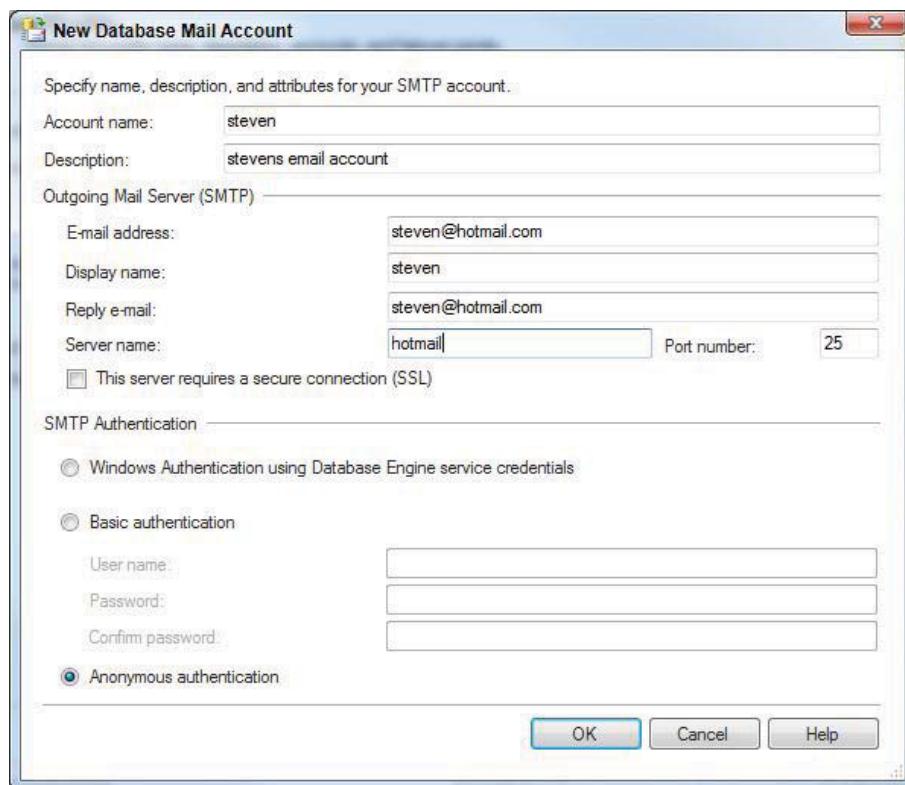
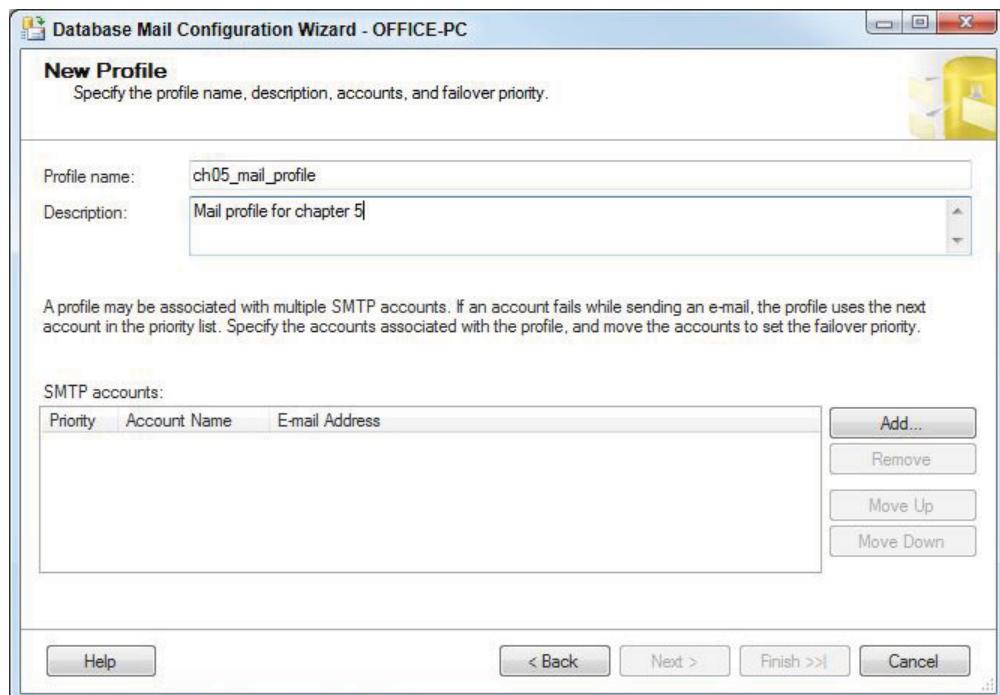
Configuration

To use Database Mail, you need to do some configuration. Specifically, you need to set up the Database Mail account, configure the mail procedure itself, and set up archiving. This is achieved using the Database Mail configuration

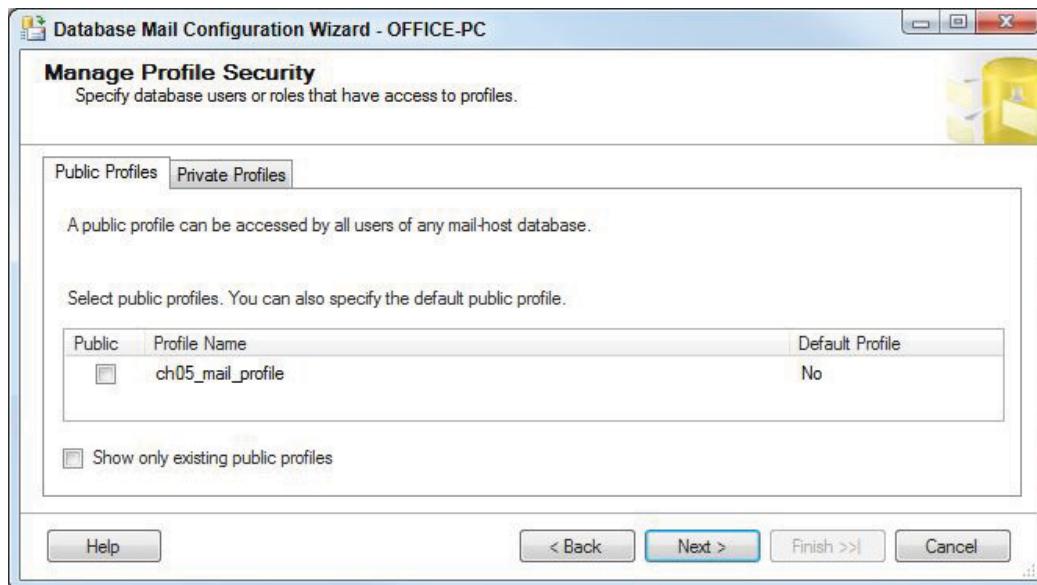
1. Use the wizard by expanding Management in SQL Server Management Studio. Right-click Database Mail and choose Configure Database Mail. This launches the Database Mail Configuration Wizard and shows the wizards welcome page.



2. Click next on the Welcome page. The next page in the wizard is the Select Configuration Page, Check the top radio button to indicate you are Setting up Database Mail for the first time. Click Next.
 3. If you haven't previously enabled Database Mail, you receive a message box asking if you want to enable the Database Mail feature. Choose yes and continue.
 4. This brings you to the New Profile dialog box, To continue, you need to add at least one mail account. Click the Add button to display the New Database Mail Account dialog, Here you provide the information needed to communicate with an SMTP server. Choose a name and description for this account.
- E-mails sent from this account will be tagged from the e-mail address and display name that you set in this section. If the recipients reply to the e-mail, the reply will be sent to the address you supply in the Reply e-mail text box.



5. In the Server name text box, provide the name of the SMTP server. This is usually in the form of `smtp.myserver.com`. Do not provide the complete URL, such as `http://smtp.myserver.com`. Database Mail does this for you. If you check the box labeled This Server Requires a Secure Connection (SSL), the URL created will be `https://smtp.myserver.com`. The default port number of 25 will suffice unless you have changed the SMTP port number.
6. Provide the SMTP login information in the SMTP authentication section. Not all SMTP servers require authentication; some require only a known sender e-mail, and others require nothing. After supplying the needed information, click OK to return to the New Profile dialog, and then click next.
7. The next page is the Manage Profile Security dialog; here you set up public and private profiles. Check the Public check box next to a profile to make it public. You may also want to set this as a default profile.



8. Click Next to move to the Configure System Parameters page. On this page you can change the values for system parameters such as retry attempts, maximum file size, prohibited extensions, and logging level. The default values work well in most cases.
9. Click Next to view the Complete the Wizard page, where you have a last chance to confirm the selections you made before they are applied.
10. Click Finish to apply the changes you made and view progress and a completion report as each set of changes is made.

11. To ensure things are working properly, you should send a test e-mail. In SQL Server Management Studio, expand Management, right-click Database Mail, and choose Send test E-Mail. You are prompted to enter an e-mail address. Send the mail and wait for receipt.

Central Management Servers:

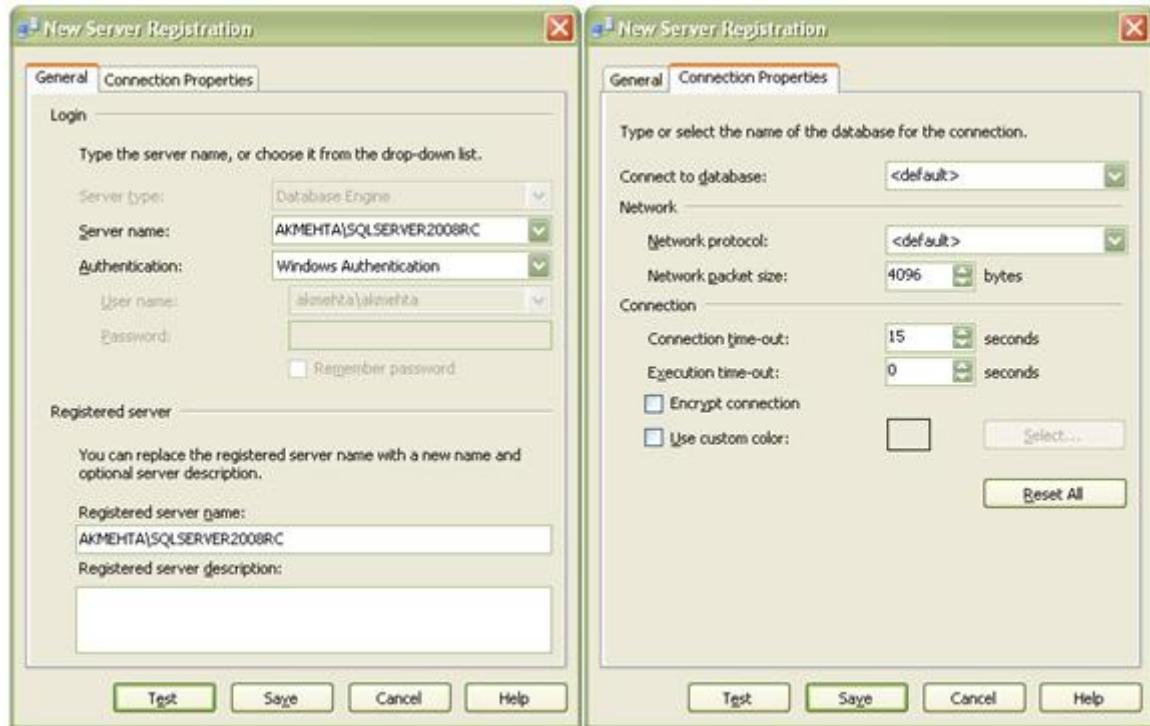
SQL Server 2008 introduces a new feature called Central Management Server (CMS) which can be used by Database Administrators within an organization to manage all the SQL Servers like 2000, 2005 and 2008 using an instance of SQL Server 2008. Central Management Server (CMS) acts as a central repository which holds the list of all the SQL Servers within an organization that needs to be managed by the team of dedicated DBAs. It would be ideal to configure a dedicated instance of SQL Server 2008 Developer Edition to act as a Central Management Server.

Configuring a Central Management Server (CMS)

1. Connect to SQL Server 2008 Instance using SQL Server Management Studio
2. On the View menu, click Registered Servers.
3. In Registered Servers window, you need to expand the Database Engine node and right click Central Management Servers to select Register Central Management Server.... from the popup window as shown in the below snippet.

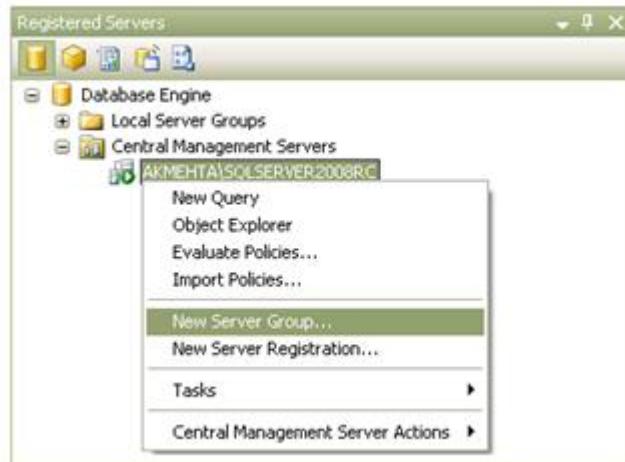


4. In New Server Registration dialog box, you need to mention the name of the SQL Server Instance which will act as the Central Management Server (CMS). Central Management Server requires Database Administrator to register a SQL Server which will act as a CMS. All the SQL Servers which are registered to Central Management Server needs to be configured using Windows Authentication Mode. You won't be able to register SQL Servers using SQL Server Authentication mode as it is not supported.



In the Connection Properties tab you can make the changes as appropriate or else you can use the default settings. To check the server connectivity you can click Test button and finally to create a Central Management Server click on Save button.

5. Once the Central Management Server is configured successfully then the next step will be create different groups under the registered CMS to organize SQL Servers. You can register SQL Server like 2000, 2005 and 2008 in CMS using SQL Server 2008. To create a new server group, right click the Central Management Server and click New Server Group.... as shown in the below snippet.



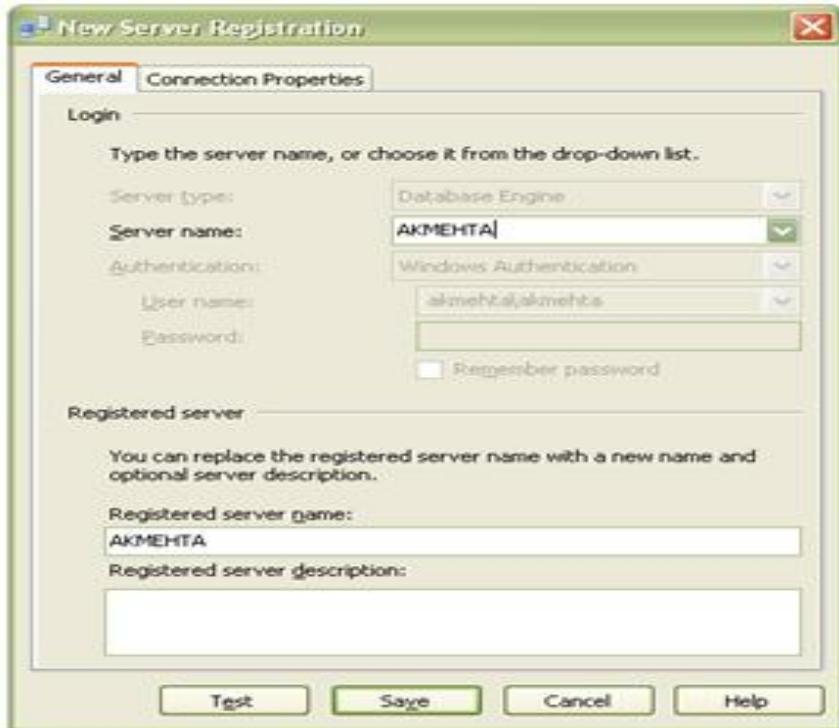
6. In the New Server Group Properties you need to provide the Group Name and a small Group description if you like. For this example the Group name used provided is MySQLServers, click OK to save the group name.



7. Next step will be add SQL Servers which needs to be managed using Central Management Server (CMS). You can add SQL Server 2000, 2005 or 2008 servers to CMS. In order to add a SQL Server, right click Group name (MySQLServers) and select New Server Registration.... from the popup window as shown in the below snippet.

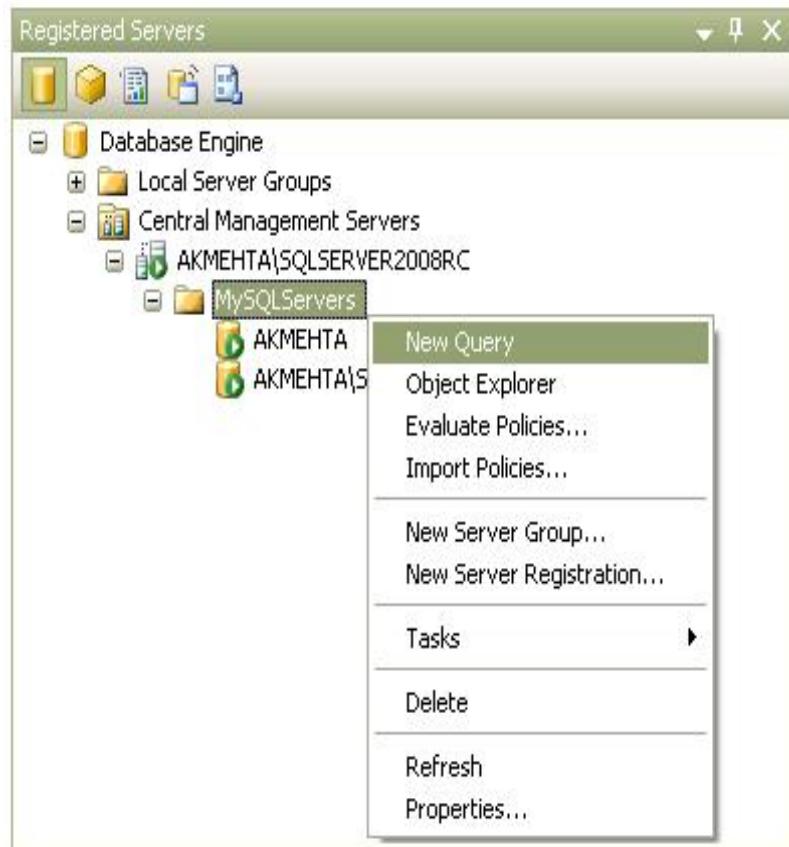


8. In the New Server Registration dialog box, you need to add one by one all the SQL Servers which needs to part of MySQLServers group and needs to be managed using Central Management Server (CMS). All the SQL Servers which are registered to Central Management Server needs to be configured using Windows Authentication Mode.



In the Connection Properties tab you can make the changes as appropriate or else you can use the default settings. To check the server connectivity you can click Test button and finally to register SQL Server click Save button.

9. Once all the SQL Servers which you want to manage using CMS are registered under a particular user group, then the next step will be to right click the user group and click New Query as shown in the below snippet to execute the query against all the servers which are part of that group.



10. In the New Query window, execute the below mentioned query to identify the SQL Server Edition, Product Level, Product Version and SQL Server Default Collation Server Properties.

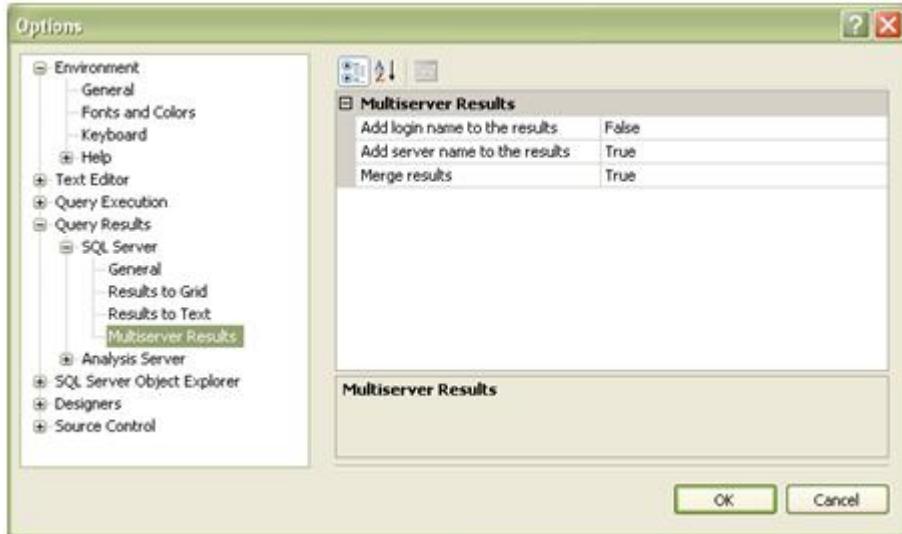
```
SELECT           SERVERPROPERTY('Edition')           AS Edition
                , SERVERPROPERTY('ProductLevel')      AS ProductLevel
                , SERVERPROPERTY('ProductVersion')    AS ProductVersion
                , SERVERPROPERTY('Collation')        AS Collation
GO
```

```

Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Community Help
New Query Execute
Registered Servers CentralManagement...EAST\akmehta) X
Database Engine Local Server Groups Central Management Servers AKMEHTA\SQLSERVER2008RC MySQLServers AKMEHTA AKMEHTA\SQL2008
Results Messages
Server Name Edition ProductLevel ProductVersion Collation
1 AKMEHTA Developer Edition SP2 9.0.3282.00 SQL_Latin1_General_CI_AS
2 AKMEHTA\SQL2008 Developer Edition RTM 10.0.1600.22 SQL_Latin1_General_CI_AS
Query executed successfully. MySQLServers akmehta\akmehta master 00:00:00 2 rows
Ln 5 Col 3 Ch 3 INS ...
Ready

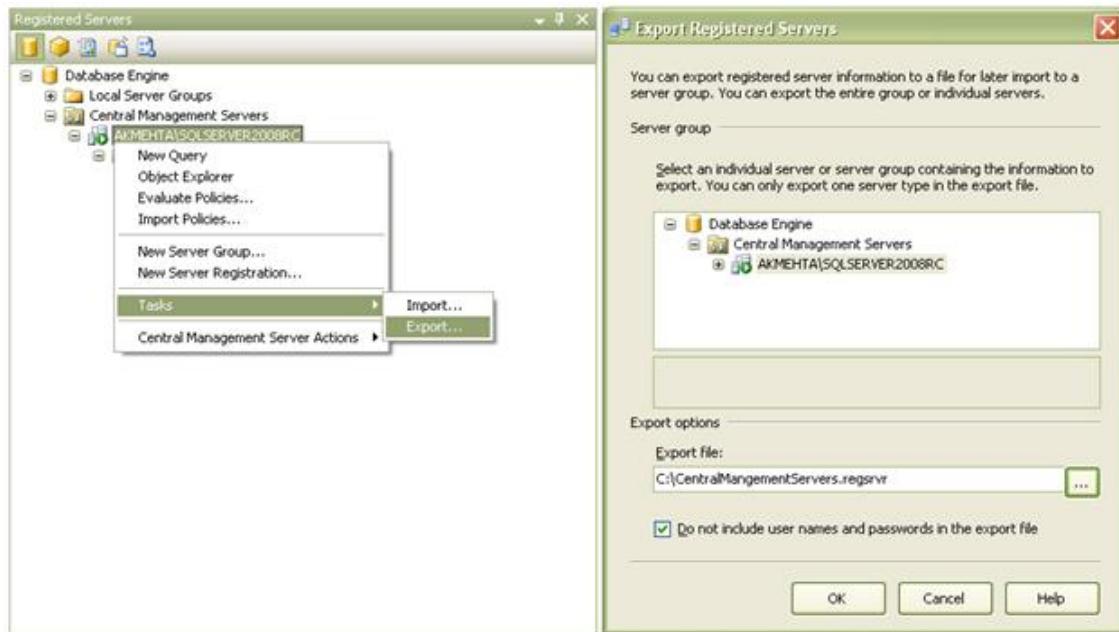
```

When you are executing the above query using CMS, internally SQL Server runs the query against all the SQL Servers which are registered under the group independently and finally the results are merged and displayed. If you want to change the way the results are display then you can go to Tools > Options > Query Results > SQL Server > MultiServer Results and make the changes as appropriate.

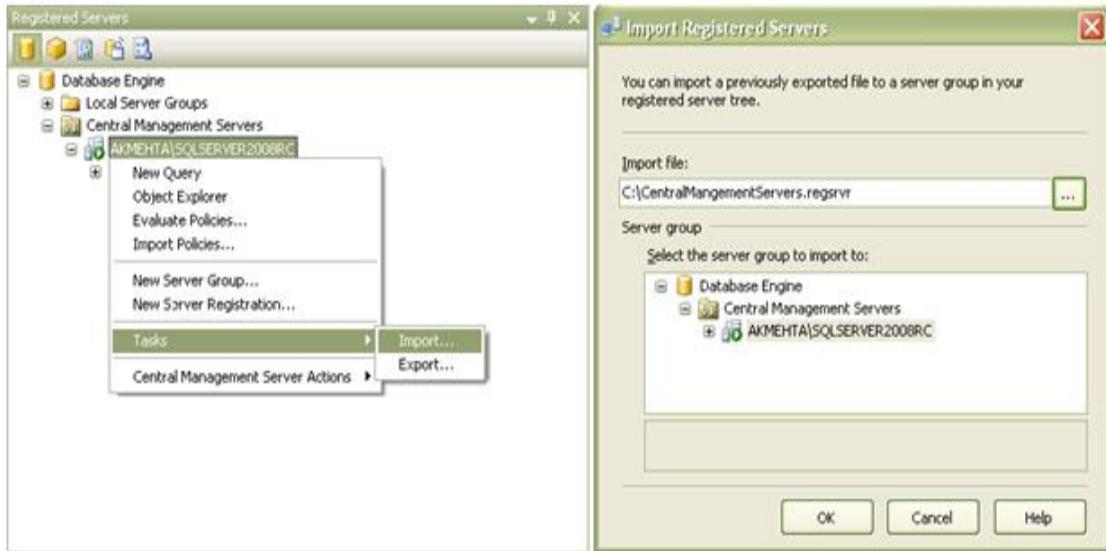


Export Registered Servers from Central Management Server
Database Administrators can export the list of registered server as a ".regsvr" file. This can be done by right clicking Central Management Server and then choose Tasks > Export.... In the

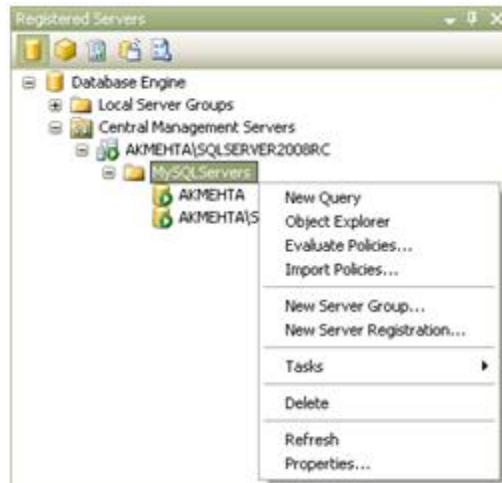
Export Registered Servers dialog box provide the file location where the registered servers need to be saved. Finally click OK to export the registered servers to a .regsvr file as shown in the below snippet. The generated file can be opened using an Internet Explorer or a Notepad to view and modify its content.



Import Registered Servers into Central Management Server Database Administrators can import the list of registered server which is saved as ".regsvr" file. This can be done by right clicking Central Management Server and choose Tasks > Import.... In the Import Registered Servers dialog box provide the location of .regsvr file which has the list of registered servers. Finally click OK to import the registered servers from .regsvr file as shown in the below snippet.



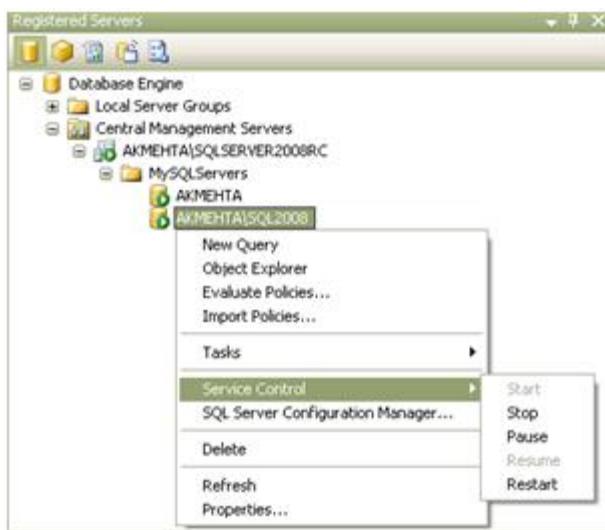
Deleting Registered Servers in Central Management Server Database Administrators can delete any registered server which is no longer required to be managed using Central Management Server. This can be done by right clicking the server and then choose the Delete option from the popup window.



Different Administrative Options Available in Central Management Servers
 Using Central Management Server database administrators can not only execute multi server queries but they can also evaluate and import policies created using Policy based Management feature of SQL Server 2008. To know more about Policy Based Management you can refer to my previous article titled "Configure and Manage Policy Based Management in SQL Server 2008". In order to

see all the registered servers within an object explorer, you can right click the server group and choose Object Explorer option from the popup window.

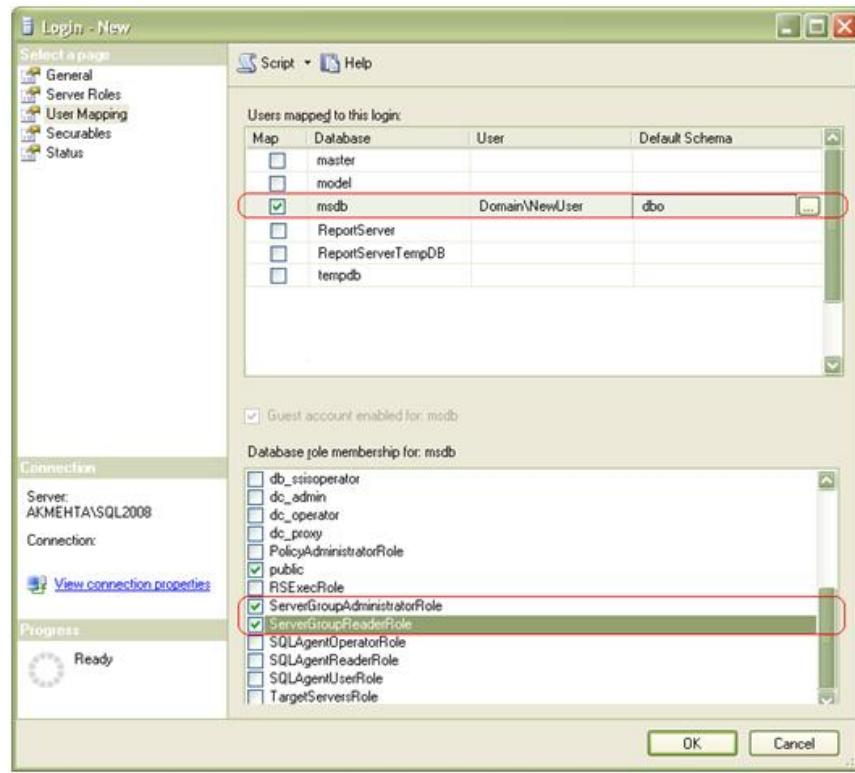
You can even start, stop, pause, resume or restart SQL Server using CMS as shown in the below snippet. If you want to open up SQL Server Configuration Manager then you need to right clicking the individual registered server and choosing SQL Server Configuration Manager... option from the popup window.



Some Drawback in Central Management Servers:

1. You won't be able to register the server which is acting as CMS under any of the groups, which means it is better to configure Central Management Server as an independent instance altogether. For this purpose you can ideal use a SQL Server 2008 Developer Edition.
2. It works only with Windows Authentication. Permissions Required to Manage and Use Central Management Servers In order to manage Central Management Server you need to be a member of Server Group Administrator Role database role within MSDB database.

In order to provide a new user access to Central Management Server (CMS), the user needs to be a member of Server Group Reader Role within MSDB database as shown in the below snippet.



Best Practices in Configuration/ Database Settings:

1. Unless you know exactly what you are doing and have already performed impartial experiments that prove that making SQL Server configuration changes helps you in your particular environment, do not change any of the SQL Server configuration settings.
2. In almost all cases, leave the “auto create statistics” and “auto update statistics” options on for all user databases.
3. In most cases, the settings for the “maximum server memory” and the “minimum server memory” should be left to their default values. This is because the default values allow SQL Server to dynamically allocate memory in the server for the best overall optimum performance. If you use AWE memory, then this recommendation is to be ignored, and maximum memory needs to be set manually.
4. Many databases need to be shrunk periodically in order to free up disk space as older data is deleted from the database. But don’t be tempted to use the “auto shrink” database option, as it can waste SQL Server resources unnecessarily. Instead, shrink databases manually.

5. Don't rely on AUTOGROWTH to automatically manage the size of your databases. Instead, proactively monitor and alter database size as circumstances dictate. Only use AUTOGROWTH to deal with unexpected growth.

Tempdb Configuration

In SQL Server 2008/2012, TempDB has taken on some additional responsibilities. As such, some of the best practice has changed and so has the necessity to follow these best practices on a more wide scale basis. In many cases TempDB has been left to default configurations in many of our SQL Server 2000 installations. Unfortunately, these configurations are not necessarily ideal in many environments. With some of the shifts in responsibilities in SQL Server 2008/2012 from the user defined databases to TempDB, what steps should be taken to ensure the SQL Server TempDB database is properly configured?

What is TempDB responsible for in SQL Server 2008/2012

- Global (##temp) or local (#temp) temporary tables, temporary table indexes, temporary stored procedures, table variables, tables returned in table-valued functions or cursors.
- Database Engine objects to complete a query such as work tables to store intermediate results for spools or sorting from particular GROUP BY, ORDER BY, or UNION queries.
- Row versioning values for online index processes, Multiple Active Result Sets (MARS) sessions, AFTER triggers and index operations (SORT_IN_TEMPDB).
- DBCC CHECKDB work tables.
- Large object (varchar(max), nvarchar(max), varbinary(max) text, ntext, image, xml) data type variables and parameters.

Case Study: Possible failures while shrinking tempdb in SQL server

Can I shrink the tempdb?

Yes, you can shrink tempdb. But shrinking files regularly is not a recommended practice, because these files may probably grow again. Also shrink operations causes' data fragmentation.

We have some limitations on shrinking tempdb:

- Shrink operations do not shrink the version store or internal objects.
- In some cases the DBA might need to restart the server in a single user mode to allow shrinking of the tempdb.

You can query sys.dm_db_file_space_usage table to see the space allocation. Use below query to see space allocation information for the files associated with tempdb

SELECT

```

SUM (user_object_reserved_page_count)*8 as usr_obj_kb,
SUM (internal_object_reserved_page_count)*8 as internal_obj_kb,
SUM (version_store_reserved_page_count)*8 as version_store_kb,
SUM (unallocated_extent_page_count)*8 as freespace_kb,
SUM (mixed_extent_page_count)*8 as mixedextent_kb
FROM sys.dm_db_file_space_usage

```

Each time the SQL Server service starts, the tempdb is newly created by copying from the model database and inheriting some database configurations from it. The default size of tempdb is 8MB

```

SELECT t1.session_id, t4.login_name, t4.host_name, t1.request_id, t3.TEXT,
t2.STATUS, t2.command, t2.blocking_session_id, t2.wait_type, t2.wait_time, t2.wait_resource,
t2.statement_start_offset, t2.statement_end_offset
FROM (SELECT session_id, request_id,
SUM(internal_objects_alloc_page_count) AS task_alloc,
SUM (internal_objects_dealloc_page_count) AS task_dealloc
FROM sys.dm_db_task_space_usage
GROUP BY session_id, request_id) AS t1,
sys.dm_exec_requests AS t2
CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS t3
INNER JOIN sys.dm_exec_sessions AS t4 ON t2.session_id = t4.session_id
WHERE t1.session_id = t2.session_id
AND (t1.request_id = t2.request_id)
--AND t1.session_id > 50
ORDER BY t1.task_alloc DESC

```

The above script will give you output which lists the actual query text, the host and login running the query, the current status and command, as well as blocking and wait information.

You can try below solutions to shrink the databases.

Method 1: Shrink database using DBCC SHRINKDATABASE.

```

DBCC SHRINKDATABASE (tempdb,'')
For ex:-
DBCC SHRINKDATABASE(tempdb,40)

```

This command shrinks the tempdb database as a whole with specified percentage.

Method 2: Shrink the tempdb data file or log file

```

Use tempdb
GO
--shrink the data file

```

```
DBCC shrinkfile (tempdb_data,'< target_size_in_MB'>)
GO
--shrink the log file
DBCC shrinkfile (tempdb_log, '<TARGET_SIZE_IN_MB'>)
```

Ex:-

```
DBCC shrinkfile (tempdb_data, 5000);
```

Errors while shrinking tempdb

If you have any active transactions in tempdb then you may receive below consistency errors.

1. Server: Msg 2501, Level 16, State 1, Line 1 Could not find table named '1525580473'. Check sysobjects.

This error may not be an indicative of any corruption, but it causes the shrink operation to fail.

2. Server: Msg 8909, Level 16, State 1, Line 0 Table Corrupt: Object ID 1, index ID 0, page ID %S_PGID. The PageId in the page header = %S_PGID.

This 8909 error indicates tempdb corruption. You can clean up the consistency errors by restarting the SQL Server. If you still see any errors then restore the database from valid backup.

3. DBCC SHRINKFILE: Page 1:456120 could not be moved because it is a work file page.

This error indicates that Page could not be moved because it is a work file page. You can release the cached objects by running "DBCC FREEPROCCACHE". Now try shrink tempdb once again to release the free space.

Major Problems with TEMPDB:-

1. tempdb has run out of space.
2. tempdb is experiencing I/O bottleneck
3. tempdb is experiencing contention

Best practices for TempDB

- Do not change collation from the SQL Server instance collation.
- Do not change the database owner from sa.
- Do not drop the TempDB database.
- Do not drop the guest user from the database.
- Do not change the recovery model from SIMPLE.

- Ensure the disk drives TempDB resides on have RAID protection i.e. 1, 1 + 0 or 5 in order to prevent a single disk failure from shutting down SQL Server. Keep in mind that if TempDB is not available then SQL Server cannot operate.
- If SQL Server system databases are installed on the system partition, at a minimum move the TempDB database from the system partition to another set of disks.
- Size the TempDB database appropriately. For example, if you use the SORT_IN_TEMPDB option when you rebuild indexes, be sure to have sufficient free space in TempDB to store sorting operations. In addition, if you are running into insufficient space errors in TempDB, be sure to determine the culprit and either expand TempDB or re-code the offending process.

Chapter – 9

Managing Database Services

Starting the SQL Server Service Automatically

You can configure an instance of Microsoft SQL Server (or SQL Server Agent) to start automatically each time you start the Microsoft Windows 2003 or Windows Server 2008 operating system. You can:

- Use the SQL Server Installer.
- Use SQL Server Configuration Manager.
- Use SQL Server Management Studio.

Normal configuration is to start the SQL Server service automatically. If a server is intentionally restarted because of software or hardware maintenance, or unintentionally restarted due to a power or hardware failure, SQL Server will become available without additional attention from an attendant.

Possible reasons to configure SQL Server not to start automatically include the following scenarios:

- You want to investigate the cause of the restart before making the database available.
- SQL Server is not always needed, and you wish to control conserve computer resources, such as on a laptop computer.

Starting SQL Server Manually

You can manually start an instance of Microsoft SQL Server or SQL Server Agent using the following methods.

Method	Description
SQL Server Configuration Manager	Start, pause, resume, and stop an instance of a local SQL Server or SQL Server Agent service.
Command prompt	Start an instance of SQL Server or SQL Server Agent service from a command prompt by the net start command or by running sqlservr.exe .

Use **sqlservr.exe** to start SQL Server from a command prompt only to troubleshoot SQL Server. Before you start an instance of SQL Server using **sqlservr.exe** from a command prompt (independent of SQL Server Configuration Manager), consider the following:

- SQL Server runs in the security context of the user, not the security context of the account assigned to run SQL Server during setup.

- All system messages appear in the window used to start an instance of SQL Server.

SQL Server can be stopped/started from SQL Server Configuration Manager

To start the default instance of SQL Server

- On the Start menu, point to All Programs, point to Microsoft SQL Server 2008, point to Configuration Tools, and then click SQL Server Configuration Manager.
- In SQL Server Configuration Manager, expand Services, and then click SQL Server.
- In the details pane, right-click SQL Server (MSSQLServer), and then click Start.
- A green arrow on the icon next to the server name and on the toolbar indicates that the server started successfully.
- Click OK to close SQL Server Configuration Manager.

To start a named instance of SQL Server

- On the Start menu, point to All Programs, point to Microsoft SQL Server 2008, point to Configuration Tools, and then click SQL Server Configuration Manager.
- In SQL Server Configuration Manager, expand Services, and then click SQL Server (*<instance_name>*).
- In the details pane, right-click the named instance of SQL Server, and then click Start.
- A green arrow on the icon next to the server name and on the toolbar indicates that the server started successfully.
- Click OK to close SQL Server Configuration Manager.

Microsoft SQL Server service can be started by using Net commands.

To start the default instance of SQL Server

From a command prompt, enter one of the following commands:

`net start "SQL Server (MSSQLSERVER)" -or-`

`net start MSSQLSERVER`

To start a named instance of SQL Server

From a command prompt, enter one of the following commands. Replace *<instancename>* with the name of the instance you want to manage.

`net start "SQL Server (instancename)" -or-`

`net start MSSQL$ instancename`

To start SQL Server with startup options

Add startup options to the end of the `net start "SQL Server (MSSQLSERVER)"` statement, separated by a space. When started using `net start`, startup options use a slash (/) instead of a hyphen (-).

```
net start "SQL Server (MSSQLSERVER)" /f /m -or-
net start MSSQLSERVER /f /m
```

Starting an Instance of SQL Server (sqlservr.exe)

If the SQL Server Database Engine does not start, one troubleshooting step is to attempt to start the Database Engine from the command prompt.

By default, sqlservr.exe is located at C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Binn. If a second instance of SQL Server is installed, a second copy of sqlservr.exe is located in a directory such as C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\binn. You can start one instance of SQL Server by using sqlservr.exe from a different instance, but SQL Server will start the version of the incorrect instance as well, including service packs, which may lead to unexpected results. To avoid this, use the MS-DOS change directory (**cd**) command to move to the correct directory before starting sqlservr.exe

To start a named instance of SQL Server in single-user mode from a command prompt

From a command prompt, enter the following command:

```
sqlservr.exe -m -s <instancename>
```

SQL Server Service Startup Options

When you install SQL Server, Setup writes a set of default startup options in the Microsoft Windows registry. You can use these startup options to specify an alternate **master** database file, **master** database log file, or error log file.

Startup options can be set by using SQL Server Configuration Manager.

Default startup options	Description
-d <i>master_file_path</i>	The fully qualified path for the master database file (typically, C:\Program Files\Microsoft SQL Server\MSSQL. <i>n</i> \MSSQL\Data\master.mdf). If you do not provide this option, the existing registry parameters are used.
-e <i>error_log_path</i>	The fully qualified path for the error log file (typically, C:\Program Files\Microsoft SQL Server\MSSQL. <i>n</i> \MSSQL\LOG\ERRORLOG). If you do not provide this option, the existing registry parameters are used.
-l <i>master_log_path</i>	The fully qualified path for the master database log file (typically C:\Program Files\Microsoft SQL Server\MSSQL. <i>n</i> \MSSQL\Data\mastlog.ldf). If you do not specify this option, the existing registry parameters are used.

You can override the default startup options temporarily and start an instance of SQL Server by using the following additional startup options.

-f	Starts an instance of SQL Server with minimal configuration. This is useful if the setting of a configuration value (for example, over-committing memory) has prevented the server from starting.
-m	Starts an instance of SQL Server with single user mode. i.e. only one administrator can connect to the instance at one time. No multiple connections are allowed in this mode.
-s	Allows you to start a named instance of SQL Server. Without the -s parameter set, the default instance will try to start. You must switch to the appropriate BINN directory for the instance at a command prompt before starting sqlservr.exe . For example, if Instance1 were to use <code>\mssql\$instance1</code> for its binaries, the user must be in the <code>\mssql\$instance1\binn</code> directory to start sqlservr.exe -s instance1 .
-T <i>trace#</i>	Indicates that an instance of SQL Server should be started with a specified trace flag (<i>trace#</i>) in effect. Trace flags are used to start the server with nonstandard behavior. For more information, see Trace Flags (Transact-SQL).

Starting SQL Server in Single-User Mode

Under certain circumstances, you may have to start an instance of Microsoft SQL Server in single-user mode by using the **startup option -m**. For example, you may want to change server configuration options or recover a damaged **master** database or other system database. Both actions require starting an instance of SQL Server in single-user mode. When you start an instance of SQL Server in single-user mode, note the following:

- Only one user can connect to the server.
- The CHECKPOINT process is not executed. By default, it is executed automatically at startup.

Starting SQL Server with Minimal Configuration

If you have configuration problems that prevent the server from starting, you can start an instance of Microsoft SQL Server by using the minimal configuration startup option. This is the startup option **-f** starting an instance of SQL Server with minimal configuration automatically puts the server in single-user mode.

When you start an instance of SQL Server in minimal configuration mode, note the following:

- Only a single user can connect, and the CHECKPOINT process is not executed.
- Remote access and read-ahead are disabled.
- Startup stored procedures do not run.

After the server has been started with minimal configuration, you should change the appropriate server option value or values, stop, and then restart the server.

Case Study/Practical Trouble Shooting

Problem:

Moved TempDB but forgot to add the name of the MDF and LDF at the end of the file path

Error: 5123, Severity: 16, State: 1 when moving TempDB

Solution:

To move tempdb is a fairly simple task, that can very easily be done incorrectly, which will cause the SQL server to not start up the next time it is restarted, which is generally immediately to put the file moves for tempdb into place. To start off with get the file path information for the current configuration of tempdb, you will need this to fall back to if you have a problem:

```
SELECT name, physical_name  
FROM sys.database_files
```

To move the files, you simply use ALTER DATABASE as follows:

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'tempdev', FILENAME = 'c:\program  
files\microsoft sql server\mssql.2\mssql\sql\data\tempdb.mdf')
```

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'templog', FILENAME = 'c:\program  
files\microsoft sql server\mssql.2\mssql\sql\data\tempdb.ldf')
```

To break my instance I am going to omit the file names and only provide the path, which is what was done in both of the posts that inspired this tread:

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'tempdev', FILENAME = 'c:\program  
files\microsoft sql server\mssql.2\mssql\sql\data\' )
```

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'templog', FILENAME = 'c:\program  
files\microsoft sql server\mssql.2\mssql\sql\data\' )
```

This will output the following result:

The file "tempdev" has been modified in the system catalog. The new path will be used the next time the database is started.

The file "templog" has been modified in the system catalog. The new path will be used the next time the database is started.

You can rerun the above query to validate the change occurred, but it won't take effect until you restart the service, so I went ahead and restarted my SQL Instance and it fails as expected with the following error log:

Error: 5123, Severity: 16, State: 1.

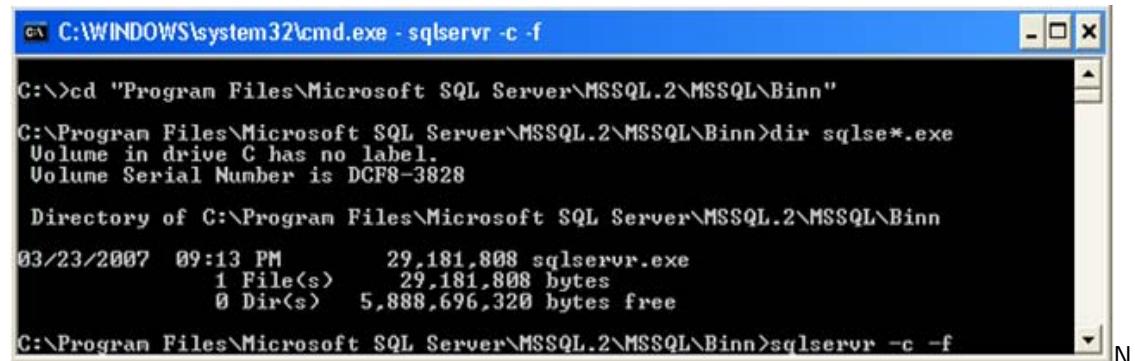
CREATE FILE encountered operating system error 3(The system cannot find the path specified.) while attempting to open or create the physical file 'c:\program files\microsoft sql server\mssql.2\mssql\data\'.

Could not create tempdb. You may not have enough disk space available. Free additional disk space by deleting other files on the tempdb drive and then restart SQL Server. Check for additional errors in the event log that may indicate why the tempdb files could not be initialized.

I had given the error information from the log file for why SQL Server failed to start. So now that we can't get into SQL Server how do we fix it. Well it isn't all that difficult to do, but you have to drop to the command prompt to do it. First open the command prompt by running cmd in the Run box:

Then change directories to the Binn directory under your SQL Instances path:

Then run sqlservr with the -c and -f startup parameters which will start SQL Server in minimal configuration mode.



```
C:\>cd "Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\Binn"
C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\Binn>dir sqlse*.exe
Volume in drive C has no label.
Volume Serial Number is DCF8-3828

Directory of C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\Binn
03/23/2007  09:13 PM      29,181,808 sqlservr.exe
               1 File(s)     29,181,808 bytes
               0 Dir(s)   5,888,696,320 bytes free
C:\Program Files\Microsoft SQL Server\MSSQL.2\MSSQL\Binn>sqlservr -c -f
```

OTE: Do not use any other startup parameters or Trace Flags as these can cause SQL to try to create tempdb from the settings that are wrong and again fail to start.

When you start SQL Server from the command prompt it will spool the log information out to the command prompt screen. When it shows Recovery is complete the SQL Server Instance is running in single user mode and can be connected to through SSMS or sqlcmd.

```
cmd C:\WINDOWS\system32\cmd.exe -sqlservr -c -f
2009-01-06 21:19:38.53 spid5s      CHECKDB for database 'master' finished without errors on 2008-05-10 01:30:00.483 (local time). This is an informational message only; no user action is required.
2009-01-06 21:19:38.54 spid5s      Server started with '-f' option. Auditing will not be started. This is an informational message only; no user action is required.
2009-01-06 21:19:38.57 spid5s      Starting up database 'mssqlsystemresource'.
2009-01-06 21:19:38.59 spid5s      The resource database build version is 9.00.3042. This is an informational message only. No user action is required.
2009-01-06 21:19:38.98 spid6s      Starting up database 'model'.
2009-01-06 21:19:38.98 spid5s      Server name is 'LI-JKEHAYIAS'. This is an informational message only. No user action is required.
2009-01-06 21:19:39.17 spid6s      CHECKDB for database 'model' finished without errors on 2008-05-10 01:30:19.403 (local time). This is an informational message only; no user action is required.
2009-01-06 21:19:39.18 spid6s      Clearing tempdb database.
2009-01-06 21:19:39.59 Server      A self-generated certificate was successfully loaded for encryption.
2009-01-06 21:19:39.62 Server      Server is listening on [ 'any' <ipv4> 1433].
2009-01-06 21:19:39.62 Server      Server local connection provider is ready to accept connection on [ \.\pipe\SQLLocal\MSSQLSERVER ].
2009-01-06 21:19:39.64 Server      Server named pipe provider is ready to accept connection on [ \.\pipe\sql\query ].
2009-01-06 21:19:39.65 Server      Server is listening on [ 127.0.0.1 <ipv4> 1434].
2009-01-06 21:19:39.65 Server      Dedicated admin connection support was established for listening locally on port 1434.
2009-01-06 21:19:39.87 spid6s      Starting up database 'tempdb'.
2009-01-06 21:19:39.98 spid6s      CHECKDB for database 'tempdb' finished without errors on 2008-05-10 01:30:19.403 (local time). This is an informational message only; no user action is required.
2009-01-06 21:19:42.31 Server      The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b, state: 3. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.
2009-01-06 21:19:42.31 Server      SQL Server is now ready for client connections. This is an informational message; no user action is required.
2009-01-06 21:19:42.39 spid5s      Recovery is complete. This is an informational message only. No user action is required.
```

Once you connect to the object explorer details [query analyzer] then run the correct ALTER DATABASE scripts to fix the tempdb path.

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'tempdev', FILENAME = 'c:\program files\microsoft sql server\mssql.2\mssql\data\tempdb.mdf')
```

```
ALTER DATABASE tempdb MODIFY FILE (NAME = 'templog', FILENAME = 'c:\program files\microsoft sql server\mssql.2\mssql\data\tempdb.ldf')
```

Once this has been run, you can close the SQL Server Instance running in the command prompt by pressing Ctrl+C with the window active. Then restart the SQL Service from the Services.msc snapin or the Computer Management Console and you should be back in business.

Case Study: Rebuilding System databases

We need to rebuild the system databases if the master database is corrupted or damaged. Let us discuss in detail how to rebuild system databases in SQL server 2012.

To rebuild your system databases, follow these steps:

Step 1: Go to a command prompt.

Step 2: From the command prompt, run setup.exe as if you were installing SQL Server, but pass in a few switches as shown here:

```
setup.exe /QUIET /ACTION=REBUILDDATABASE /INSTANCENAME=instance_name  
/SQLSYSADMI
```

```
NACCOUNTS=accounts /SAPWD=sa password
```

The switches indicate the following:

/QUIET: suppresses the errors and warnings while the rebuild runs. You see a blank screen while the process completes. Errors will still be logged to the Error Log.

/ACTION: indicates that the action is to rebuild the database, by providing the REBUILDDATABASE parameter.

/INSTANCENAME: provides the name of the instance where system databases should be rebuilt. Use MSSQLSERVER for a default instance.

/SQLSYSADMINACCOUNTS: provides windows groups or individual accounts that should be provisioned as sysadmin. Use this option when SQL Server is configured for Windows Authentication Mode.

/SAPWD: specifies the SA password. Use this option when SQL Server is configured for Mixed Authentication Mode.

```
start /wait setup.exe /qn INSTANCENAME="MSSQLSERVER" REINSTALL=SQL_Engine  
REBUILDDATABASE=1 SAPWD="XXXX"
```

where XXXX is the name of the password.

INSTANCENAME="MSSQLSERVER" for default instance of SQL 2012 and

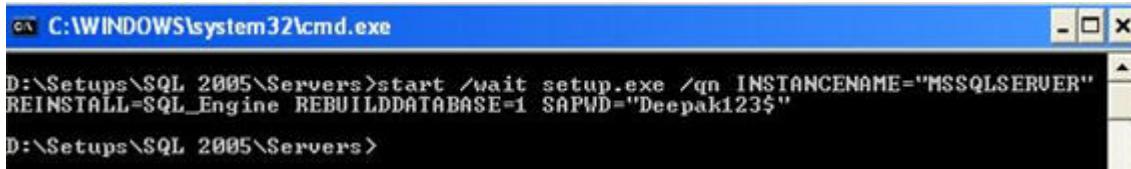
INSTANCENAME="MSSQL\$INSTANCENAME" for named instance of SQL 2012.

For example,

If you have a named instance named as "Deepak\Test" then type as below,

INSTANCENAME="MSSQL\$TEST" in the above command

Refer the below screenshot for the same.



```
C:\WINDOWS\system32\cmd.exe
D:\Setups\SQL 2005\Servers>start /wait setup.exe /qn INSTANCENAME="MSSQLSERVER"
REINSTALL=SQL_Engine REBUILDDATABASE=1 SAPWD="Deepak123$"
D:\Setups\SQL 2005\Servers>
```

Step 3: After executing the command in step 2 the rebuild process will start and will complete within 5 minutes. You can verify whether the databases are rebuilt by navigating to folder containing the data and log files for the system databases. If you arrange them using modified date it will clearly show the time when it was last modified and it is the time when we executed the command in Step 2.

Step 4: Once the rebuild is completed, connect to the SQL server using SSMS. In the object explorer only the system databases will be available.

If any user db were present prior to rebuild it will be lost and we need to perform as below to retrieve it.

1. Restore from the backup taken in Step 1 (or)
2. We can attach from the data and log files of the user db as they will not be cleaned up during rebuild process.

NOTE : No Need to detach all the user databases before rebuild as the ldf and mdf files will be present in the same path as usual and will not be overwritten.

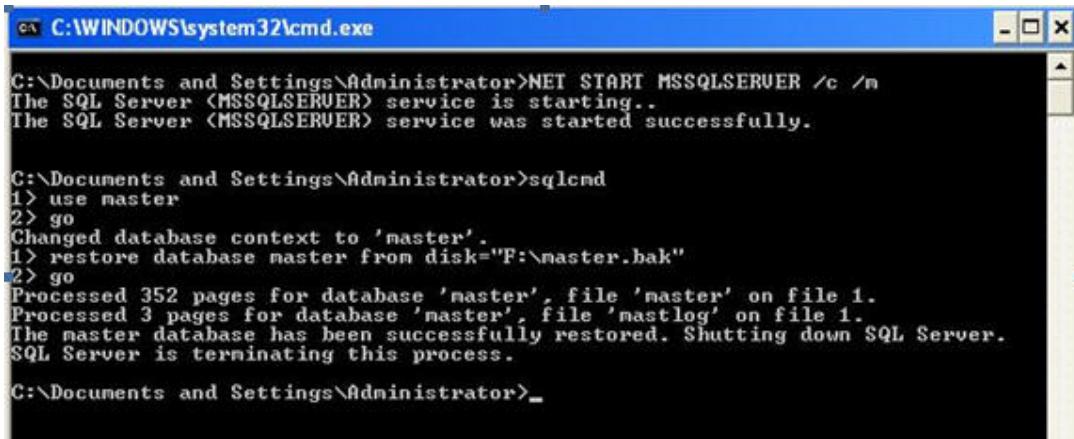
Now we need to restore the system databases from the backup which we took in Step 1.

Master database can be restored only in Single user mode (refer Step 5) and other dbs can be restored normally.

NOTE : The ldf and mdf files of the system databases will be overwritten and hence we cannot perform detach/ attach operation.

Step 5: In order to restore master database perform the below steps,

- Stop SQL server and start it using the below command from the command prompt
- NET START MSSQLSERVER /c /m which will start SQL in single user mode
Note: For default instance its MSSQLSERVER, for named instance its MSSQL\$instanceName-Type as shown in the below screenshot. Once the restore is completed SQL server will shut down and hence we need to start it normally and access the databases.



```
C:\Documents and Settings\Administrator>NET START MSSQLSERUER /c /n
The SQL Server <MSSQLSERVER> service is starting..
The SQL Server <MSSQLSERVER> service was started successfully.

C:\Documents and Settings\Administrator>sqlcmd
1> use master
2> go
Changed database context to 'master'.
1> restore database master from disk="F:\master.bak"
2> go
Processed 352 pages for database 'master', file 'master' on file 1.
Processed 3 pages for database 'master', file 'mastlog' on file 1.
The master database has been successfully restored. Shutting down SQL Server.
SQL Server is terminating this process.

C:\Documents and Settings\Administrator>_
```

Case Study: Server Collation in SQL Server 2008

While installing SQL Server 2008 we may miss to choose the right collation and we need to rectify this by changing the collation at server level. You can change the collation of sql server without uninstalling. Let's discuss the necessary steps for changing collation for sql server.

Steps for changing collation

- Take backup of all the databases & logins exists in the server for safer side.
- Detach all the user databases
- Insert SQL Server 2008 CD \ DVD into drive.
- Below is the syntax for changing the collation at serverlevel, please note that this will rebuild all the system databases in that instance/.

```
setup.exe /q /ACTION=RebuildDatabase /INSTANCENAME=InstanceName /SAPWD="New SA Password" /SQLCollation=CollationName /SQLSYSADMINACCOUNTS="Admin ID"
```

Where,

/q - perform silent installation

/Action - We are rebuilding the system databases to change the collation hence the parameter is always RebuildDatabase only

/INSTANCENAME - Name of the instance you are going to change the collation

/SAPWD - Provide new password for SA login

/SQLCollation - Provide the new collation name of SQL Server

/SQLSYSADMINACCOUNTS - Provide a account name which has admin rights in sql server.

Please note that this account should be windows authenticated account having sysadmin privilege in sql server

- Once its done check the new collation of sql server
- Attach all the user databases to SQL Server and re-create the logins.
- Check application functionality.

I'm going to test the above steps with an SQL Server 2008 environment which has an existing collation "SQL_Latin1_General_CI_AS (Latin1-General, case-insensitive, accent-insensitive, kanatype-insensitive, width-insensitive for Unicode Data, SQL Server Sort Order 54 on Code Page 1252 for non-Unicode Data)" to the collation "SQL_Ukrainian_CI_AS (Ukrainian, case-sensitive, accent-sensitive, kanatype-insensitive, width-insensitive for Unicode Data, SQL Server Sort Order 107 on Code Page 1251 for non-Unicode Data)"

Sample exercise to change the collation

- Before changing the collation you can find the collation name.

```
SQLQuery1.sql - W2GZ36Y...)* Object Explorer Details
SELECT @@VERSION as 'Server Version'
SELECT SERVERPROPERTY('COLLATION') as 'Server Collation'

Results | Messages |
Server Version
1 Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 ...
Server Collation
1 SQL_Latin1_General_CI_AS
```

- Execute the below command in Dos prompt to start changing the collation

```
M:\SQL2k8\setup.exe /q /ACTION=RebuildDatabase /INSTANCENAME=SQLEXPRESS
/SAPWD="SQL2K8" /SQLCollation=SQL_Ukrainian_CI_AS
/SQLSYSADMINACCOUNTS="SAGARSYS\Admin"
```

Where M:\ is my DVD drive letter.

```
C:\WINDOWS\system32\cmd.exe
D:\>M:\SQL2k8\setup.exe /q /ACTION=RebuildDatabase /INSTANCENAME=SQLEXPRESS /SAPWD="SQL2K8" /SQLCollation=SQL_Ukrainian_CI_AS /SQLSYSADMINACCOUNTS="SAGARSYS\Admin"
```

- Since this is silent installation it wont ask anything it will just start working on it, once its done the DOS prompt will be like below

```
Microsoft (R) SQL Server 2008 Setup 10.00.1600.22
Copyright (c) Microsoft Corporation. All rights reserved.

D:\>
```

- You can find from the screenshot below is that the new collation will be Cyrillic_General_CI_AI. That's all collation change has successfully completed.

The screenshot shows a SQL Server Management Studio window with two results panes. The top pane displays the following T-SQL code:

```
SELECT @@VERSION as 'Server Version'  
SELECT SERVERPROPERTY('COLLATION') as 'Server Collation'
```

The bottom pane has two results sections. The first section, titled "Server Version", shows one row with the value "Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 (...)" in the "Results" tab. The second section, titled "Server Collation", also shows one row with the value "SQL_Ukrainian_CI_AS_WS" in the "Results" tab.

Chapter – 10

Migrating SQL Server

Side-by-Side Migration

In a side-by-side upgrade, SQL Server 2012 installs either along with SQL Server 2008 (or 2008 R2) as a separate instance or on a different server. This process is essentially a new installation followed by a database migration. You may want to select this option as part of a hardware refresh or migration to a new platform, such as Itanium or x64. Because of the backup and restore times involved in a back-out scenario, if you have a sizable database, this is definitely the option to use.

As part of this method, you can simply back up the databases from the original server and then restore them to the SQL Server 2012 instance. Other options are to manually detach your database from the old instance and reattach it to the new instance, use log shipping, or database mirroring. You can also leverage the Copy Database Wizard to migrate your databases to the new server. Although this approach provides for a good recovery scenario, it has additional requirements beyond those of the in-place upgrade, such as maintaining the original server name, caring for application connectivity, and keeping users and their logins in sync.

In-Place Upgrade versus Side-By-Side Upgrade Considerations

Consider numerous factors before selecting an upgrade strategy. Your strategy should include the need for a component-level upgrade, the ability to roll back in case of failure, the size of your databases, and the need for partial upgrade. Your top priorities might depend upon if you can upgrade to new hardware, facilitate a change of strategy such as a server consolidation, and manage a small server outage window for the upgrade.

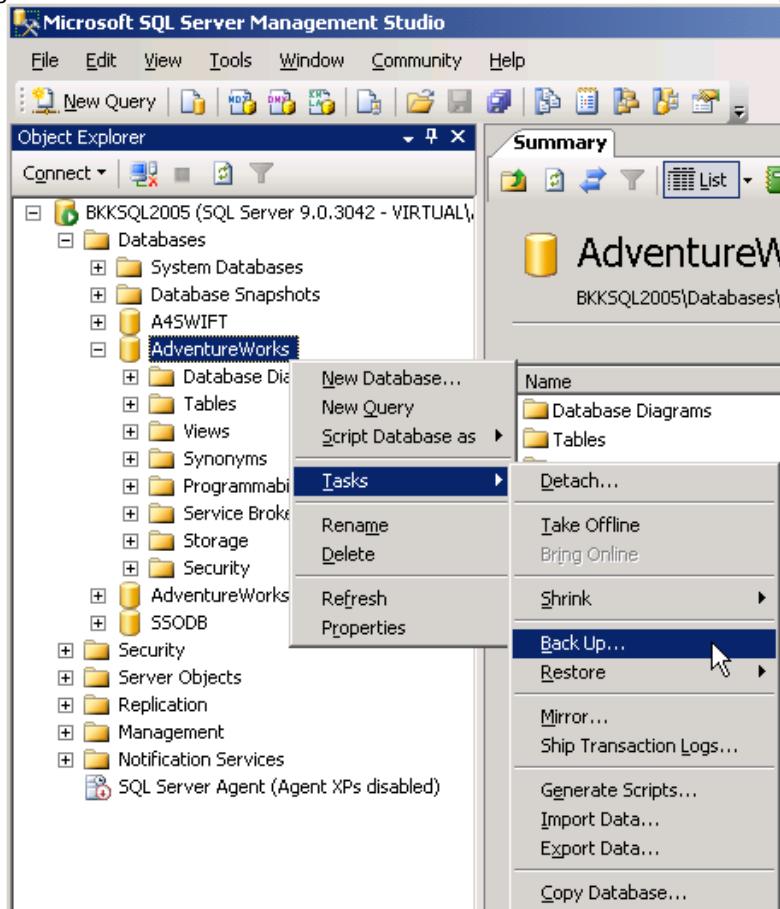
PROCESS	IN-PLACE UPGRADE	SIDE-BY-SIDE UPGRADE
Number of resulting instances	One	Two
Data file transfer	Automatic	Manual
SQL Server instance configuration	Automatic	Manual
Supporting upgrade utility	SQL Server setup	Various migration and data transfer methods

1. Migrating the database by using Backup & Restore

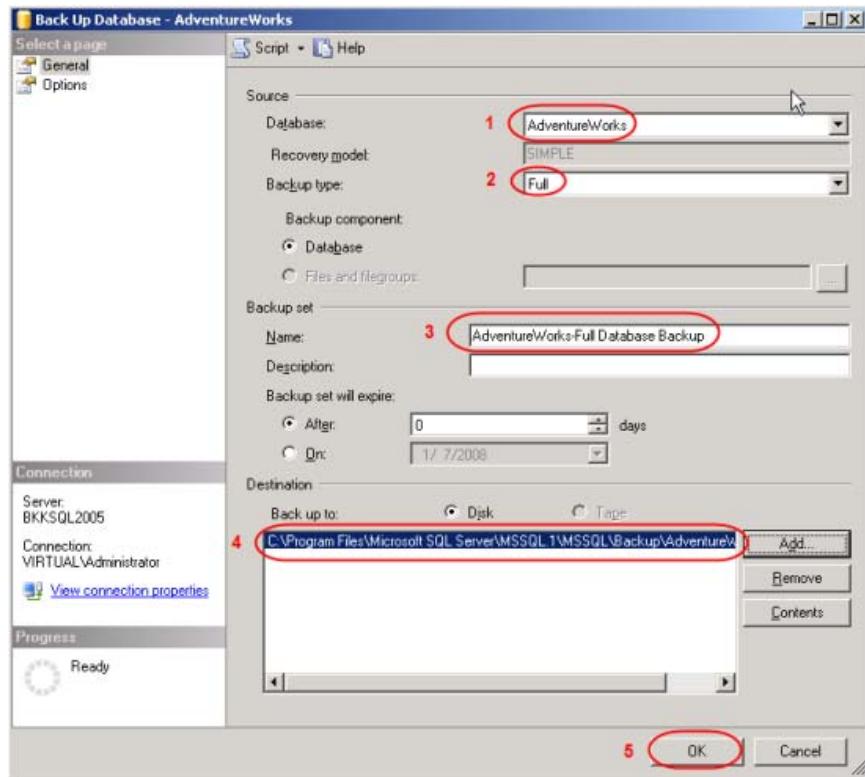
By using backup, you can backup a database without interrupt any transactions on the database. I will backup a database from SQL Server 2005/2008 R2 and restore the database to another SQL Server 2012 instance.

1. Connect to source 2005/2008 instance. Open Microsoft SQL Server Management Studio and connect to BKSQL2008 R2.

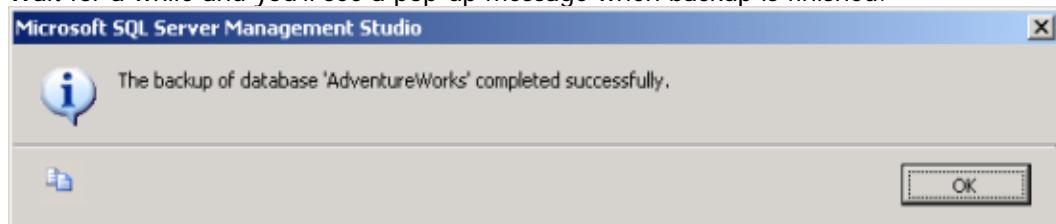
2. Right-click on the Adventure Works database. Select Tasks -> Backup...



3. On Back up Database window, you can configure about backup information. If you're not familiar these configurations, you can leave default values. Here are some short descriptions.
1. Database – a database that you want to backup.
 2. Backup type – you can select 2 options: Full and Differential. If this is the first time you backup the database, you must select Full.
 3. Name – Name of this backup, you can name anything as you want.
 4. Destination – the file that will be backup to. You can leave as default. Default will backup to "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup".
 5. Click OK to proceed backup.

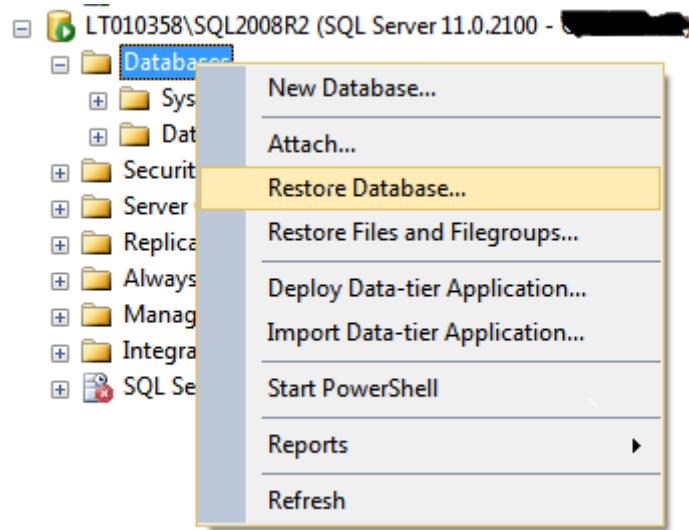


4. Wait for a while and you'll see a pop-up message when backup is finished.

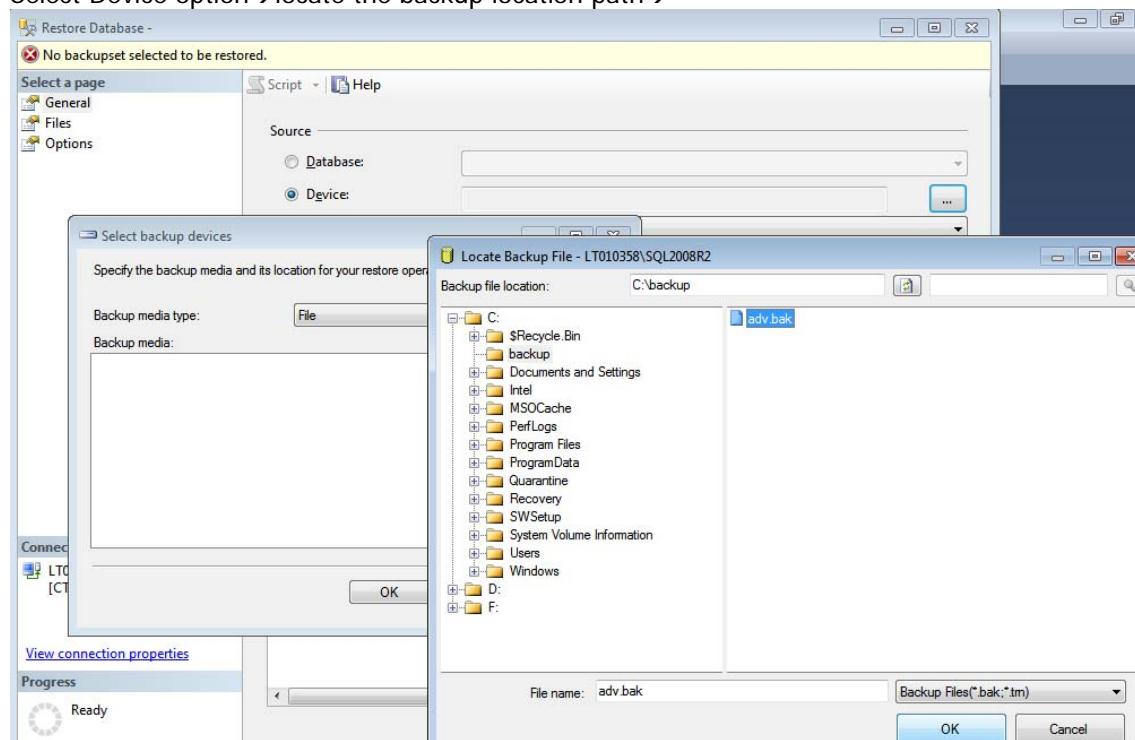


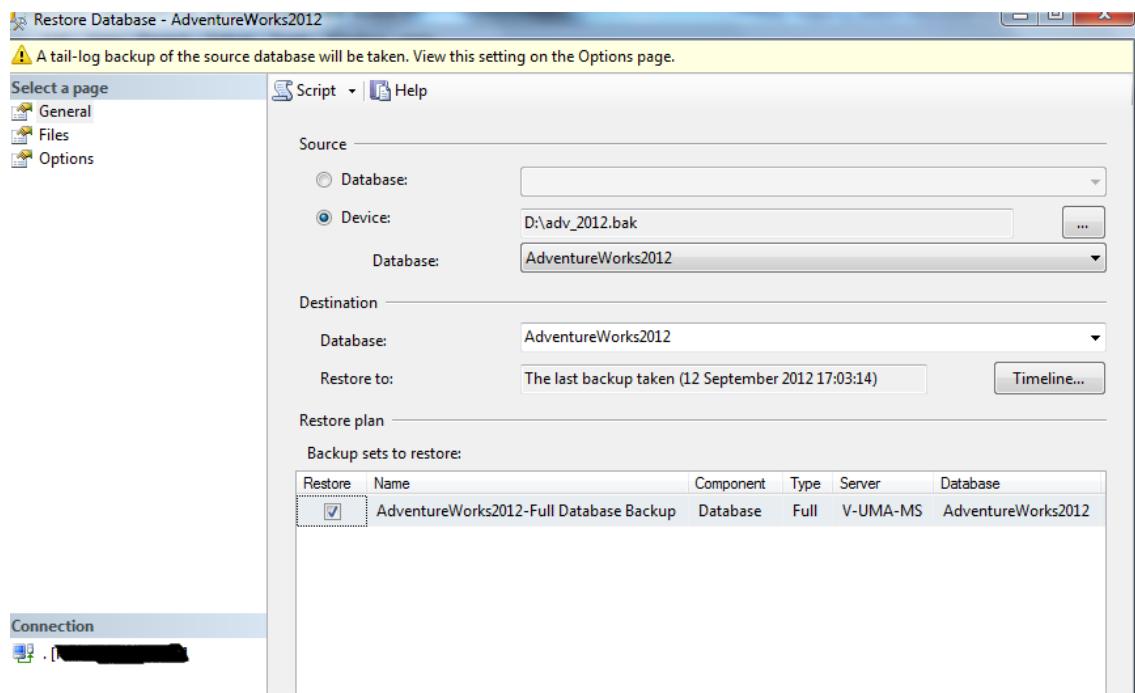
Restore the database.

Go to the 2012 instance → select databases → select restore database option → give the database name in General tab → select the from databases → give the backup file path



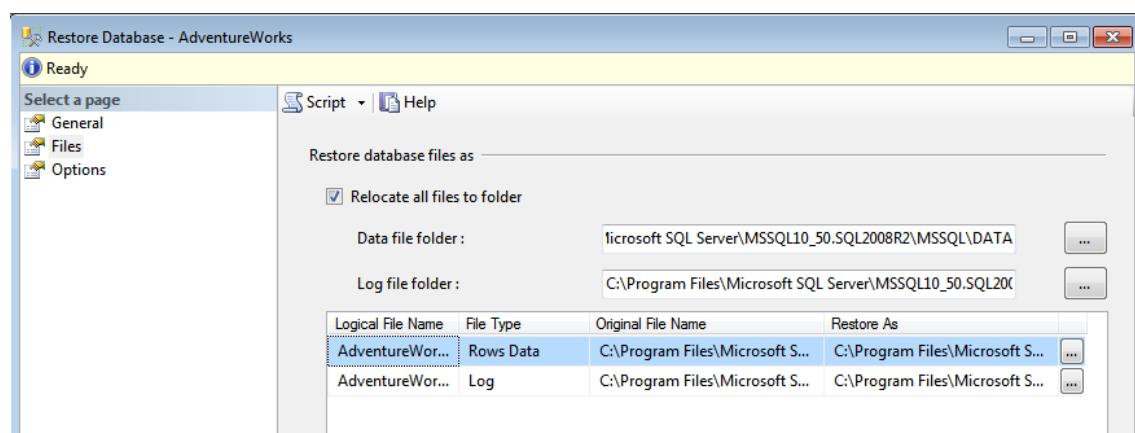
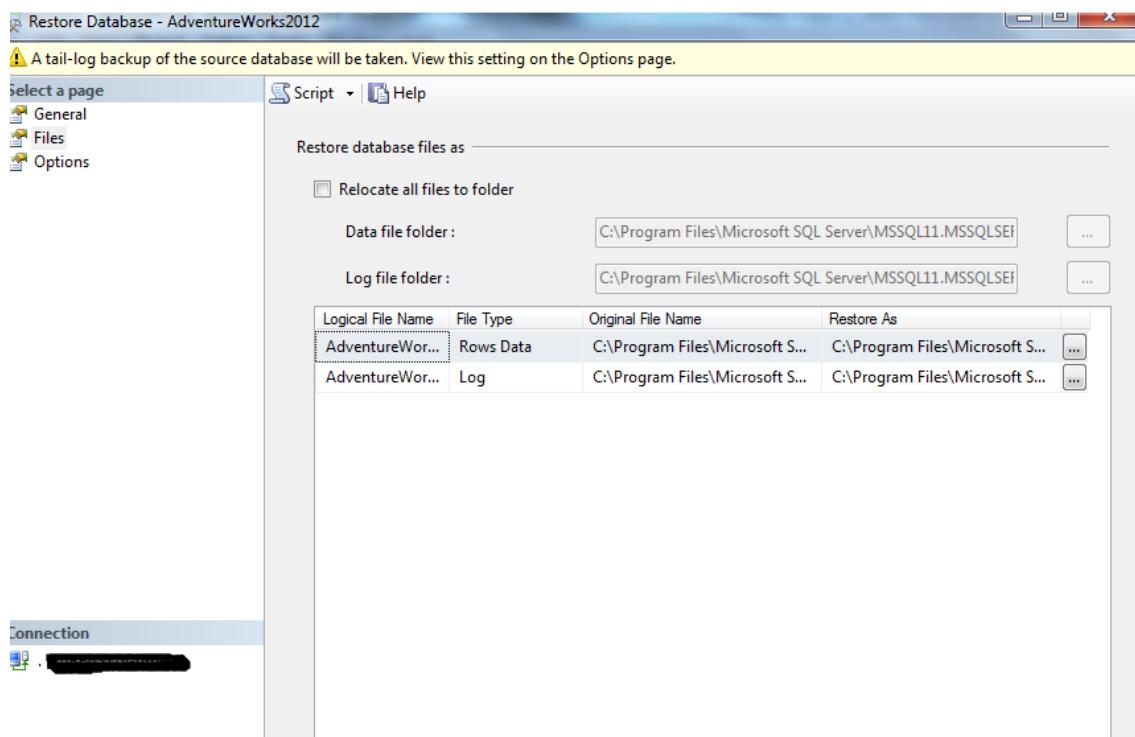
Select Device option → locate the backup location path →



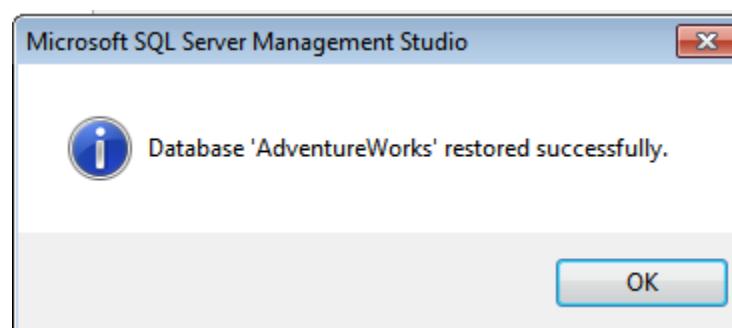
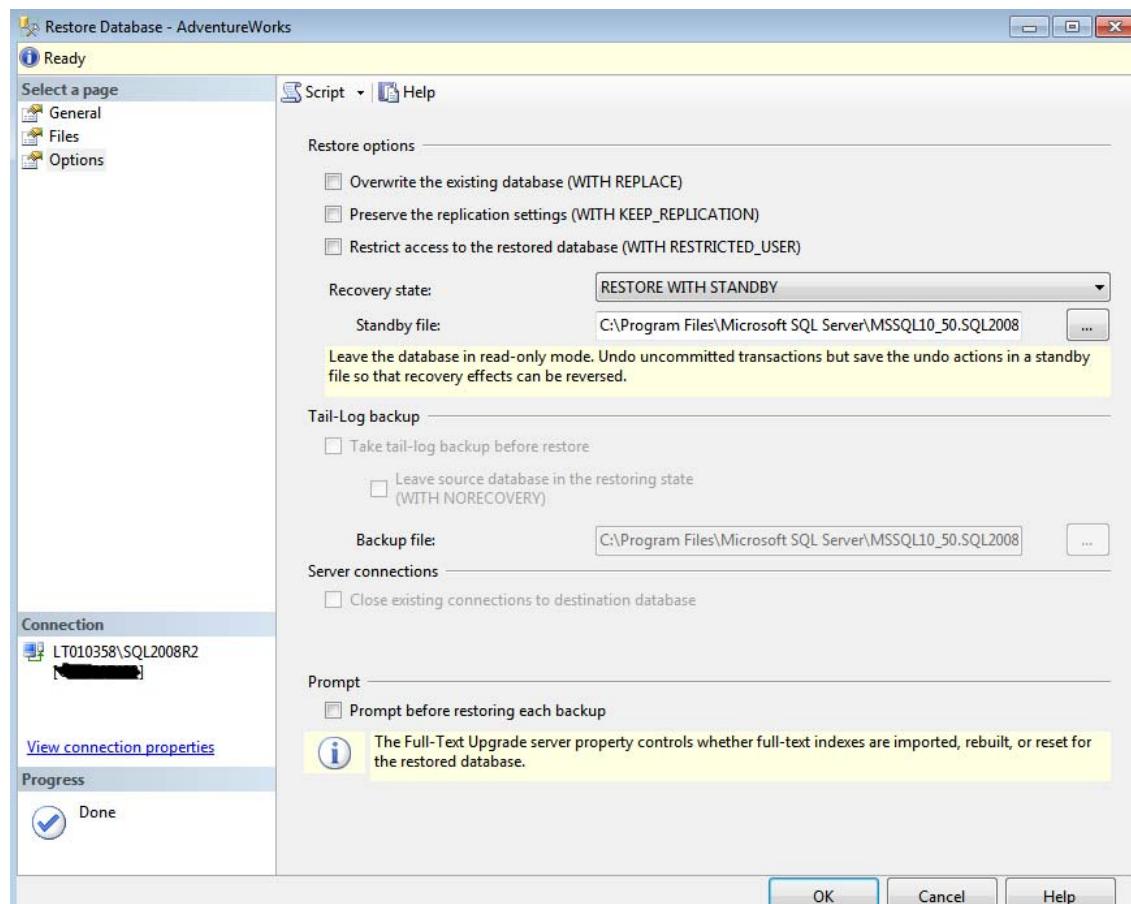


Goto files pane → give the correct data and log file path according to the that server settings
→ and click ok

Go to → option pane select the recovery option(stand by/with recovery)—>click ok



Make sure that you are un-checking the tail-log checkbox.



2. Migrating the database using copy database wizard

What is copy database wizard?

Copy Database Wizard is a new feature from SQL Server 2005 onwards. You can make use of this feature to copy \ move databases between different instances of SQL Server. It can be used for the below purposes

- Transfer a database when the database is still available to users by using the SQL Server Management Objects (SMO) method.
- Transfer a database by the faster detach-and-attach method with the database unavailable during the transfer.
- Transfer databases between different instances of SQL Server 2008 R2.
- Upgrade databases from SQL Server 2000 to SQL Server 2005.

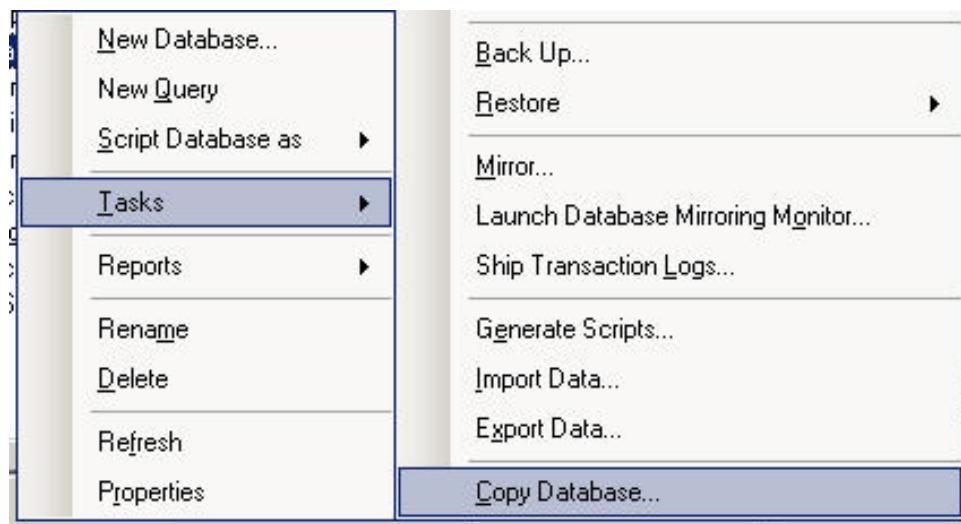
Note: The Server from which you are running CDW should be patched with minimum SQL Server SP2 (better update with latest SP) for Copy Database Wizard (CDW) to work properly

Permission Required:

To use the Copy Database Wizard, you must be a member of the sysadmin fixed server role on the source and destination servers. To transfer databases by using the detach-and-attach method, you must have file system access to the file-system share that contains the source database files.

How to Use it?

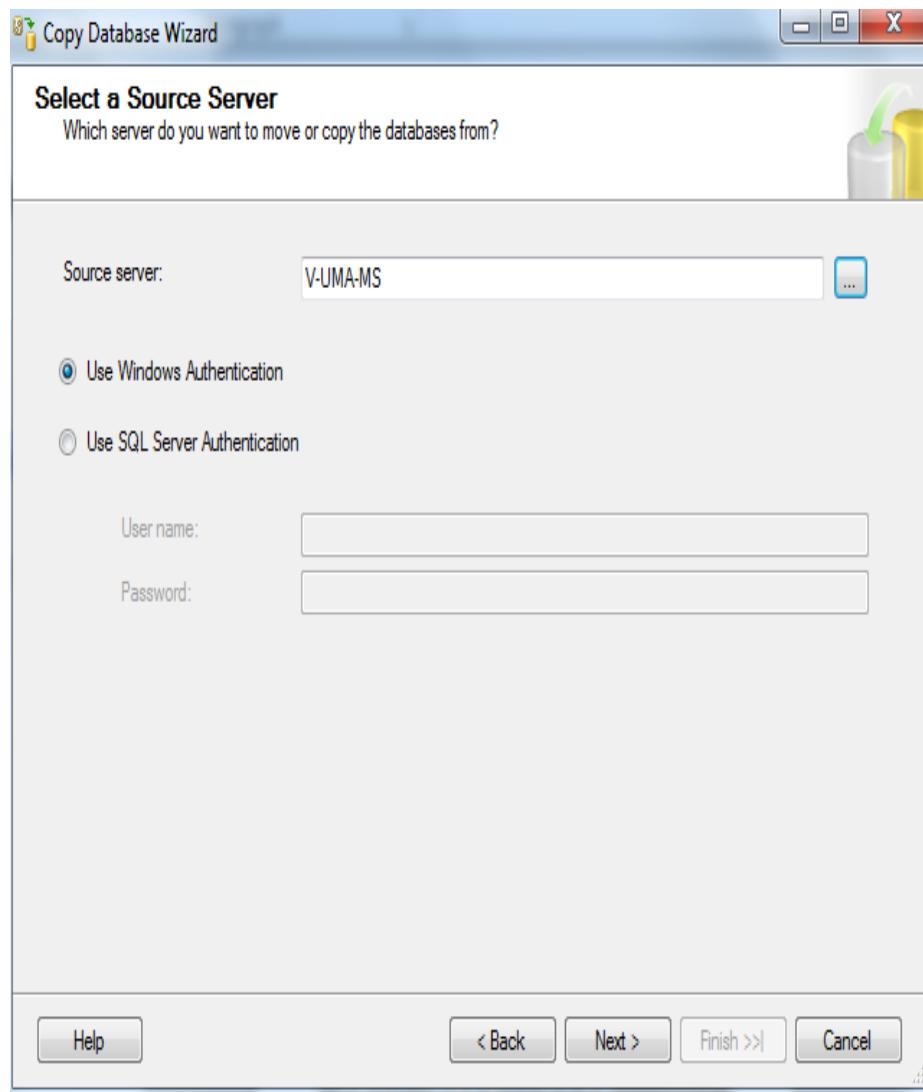
- Open SSMS in source or destination server which is running SQL Server 2005/2008
- Right click on any of the database and then click on Tasks from the select Copy database wizard as shown below



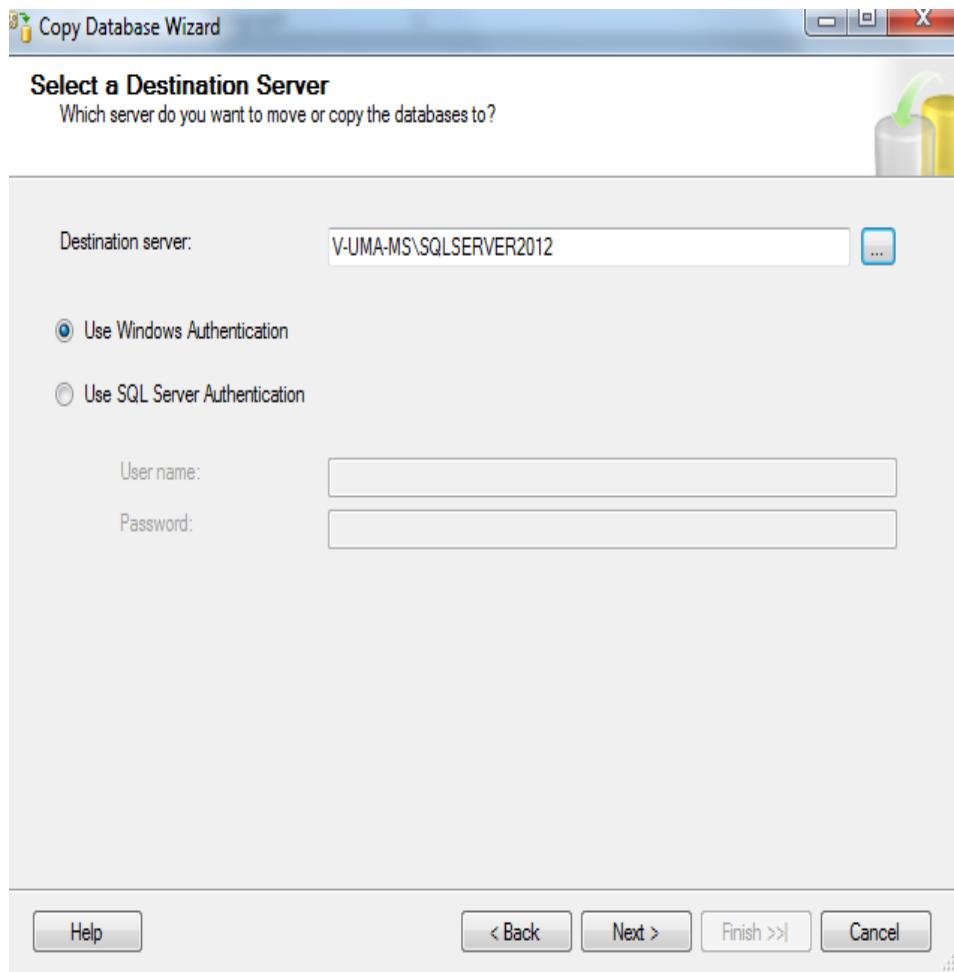
- Once opened you can see the welcome screen as below



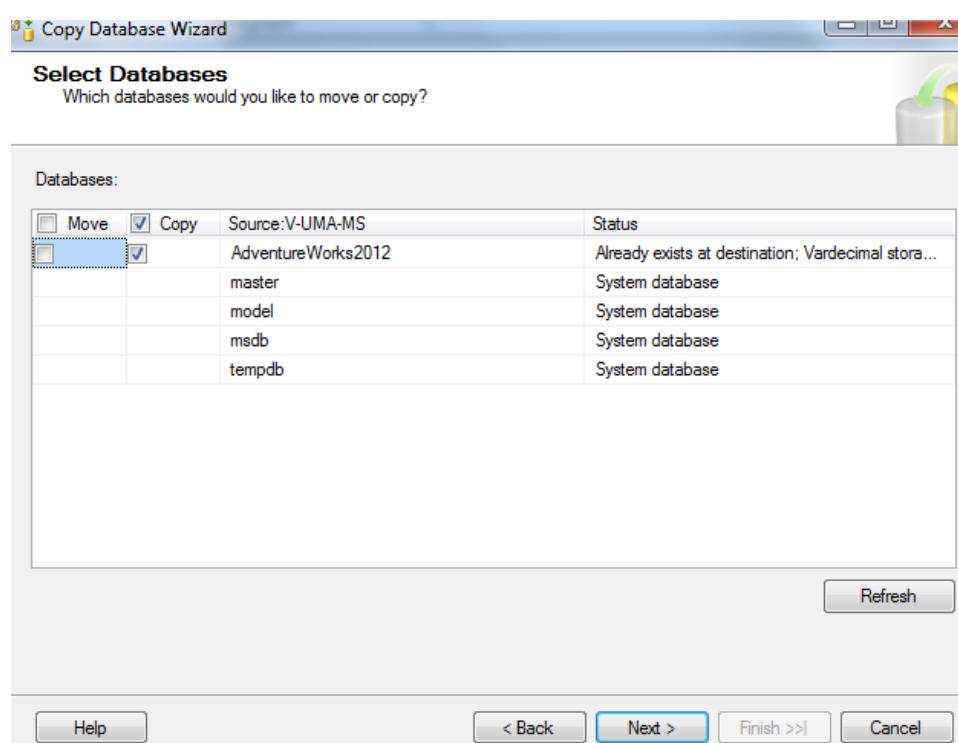
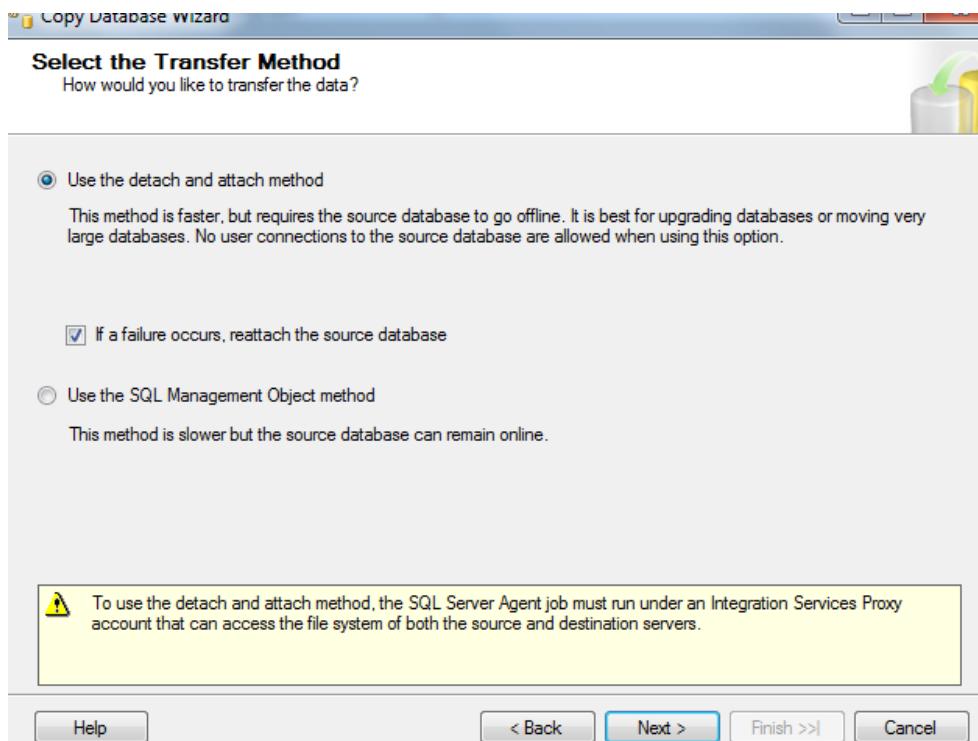
- Click on next to proceed with wizard, In this screen you need to provide the source server name i.e. 2005/2008 and the credentials



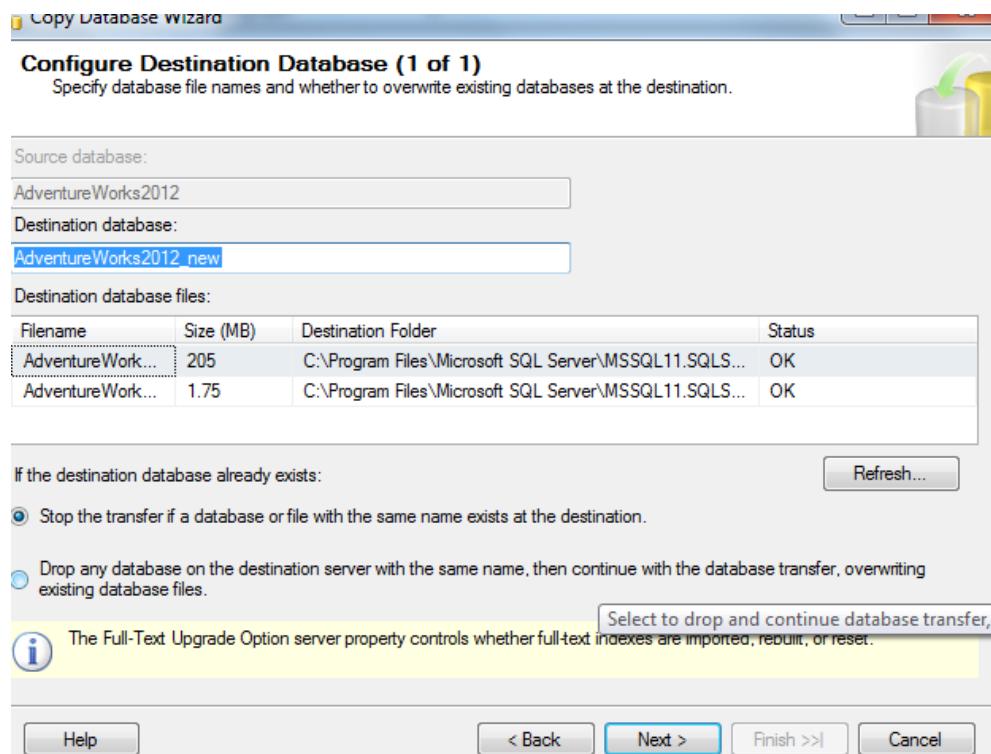
- After this click on next and in this screen provide the destination server(which should be SQL Server 2012) and its credentials



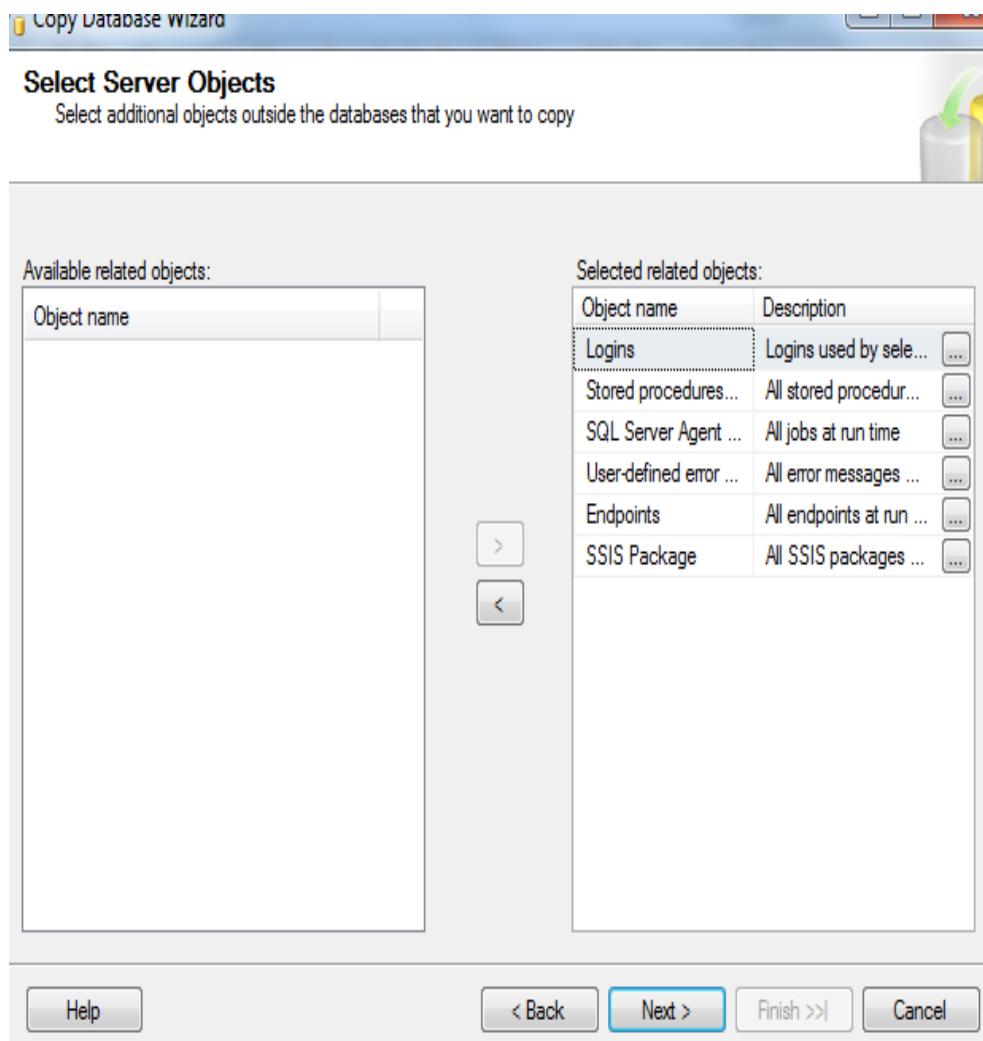
- Once source & destination server details given, you need to select the way by which you are going to copy \ move the database.
 - Detach \ Attach — Faster methods, requires db to be offline. Users will be disconnected and physical files of the db will be copied to the destination server
 - SMO — Slower method, db will be in online state. This will create the db in the destination server with the same name and copy all the data's from source



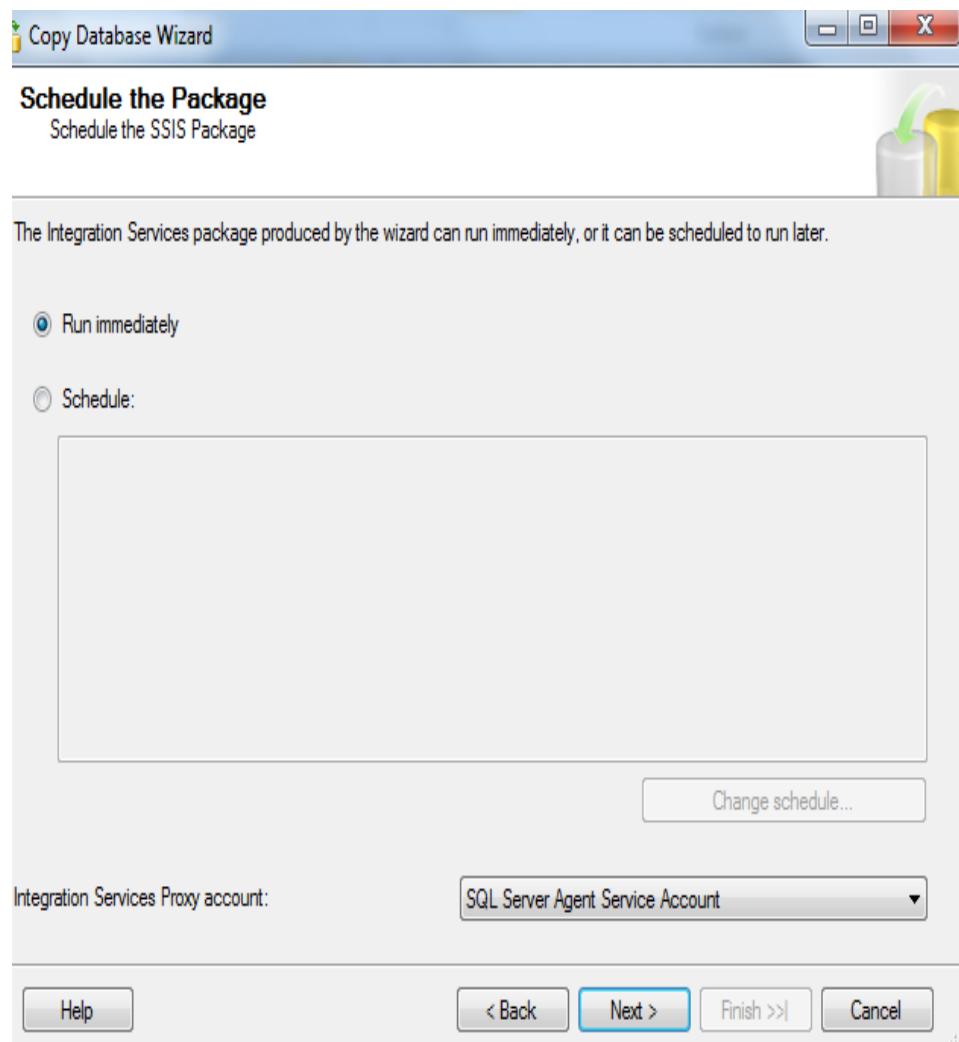
- Once the movement method chosen you can select the databases from the source server and you need to specify whether its move or copy.
 - Copy — Will copy the db to the destination server and the database will be online in both the servers
 - Move — In this method the wizard automatically deletes the source database after moving the database to destination
- In the next page you need to provide the new db name and the path where CDW should place the physical files in the destination server



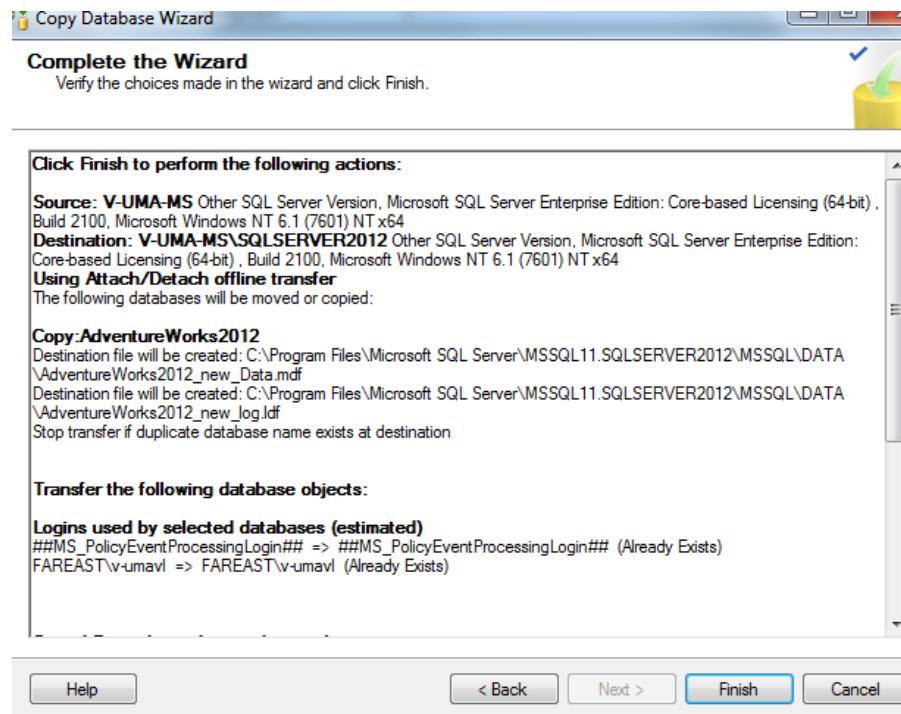
- The next page is the good feature in CDW, here you can select the logins, objects, jobs & SP which is related to the database you are trying to copy making our job simpler. Here I'm just copy the logins alone.



- In the next page you need to provide the package name and the log file for this process, so that incase of failure you can review it.
- In the next window you can select the package to execute immediately or else you can schedule the same to run after some time after EOD



- You can review the full summary before it proceeds as shown in the below screenshot



- Clicking on finish will create a job with the name mentioned in "Configure the package" page. If you have selected to run immediately the job will be scheduled to run in the near time or else it will be scheduled as given in the page.
 - **Note:** Once the db copy \ movement is done you can delete the job or else it will be stayed in your job list.
- Once it succeeded the db will be moved \ copied to the destination server. You can query sys.databases catalog view to check the same.

Considerations:

- You cannot move system databases
- Selecting move option will delete the source db once it moves the db to destination server
- If you use SMO method to move full text catalogs then you need to repopulate it
- SQL Server Agent should be running or else it will fail in job creation step
- You can't move encrypted objects (like objects, certificates etc) using CDW

3. Migrating the database by using Detach & Attach method.

If database is to be from one database to another database, the following script can be used detach from old server and attach to new server.

Process to move database:

----Step 1: Detach Database using following script

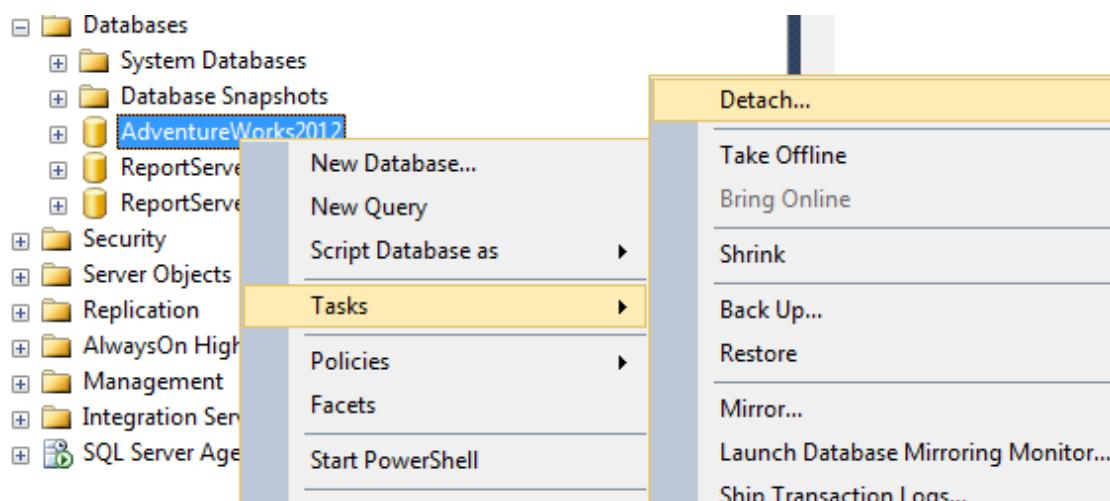
```
USE [master]
EXEC master.dbo.sp_detach_db @dbname = N'AdventureWorks',
GO
```

----Step 2 : Move Data files and Log files to new location

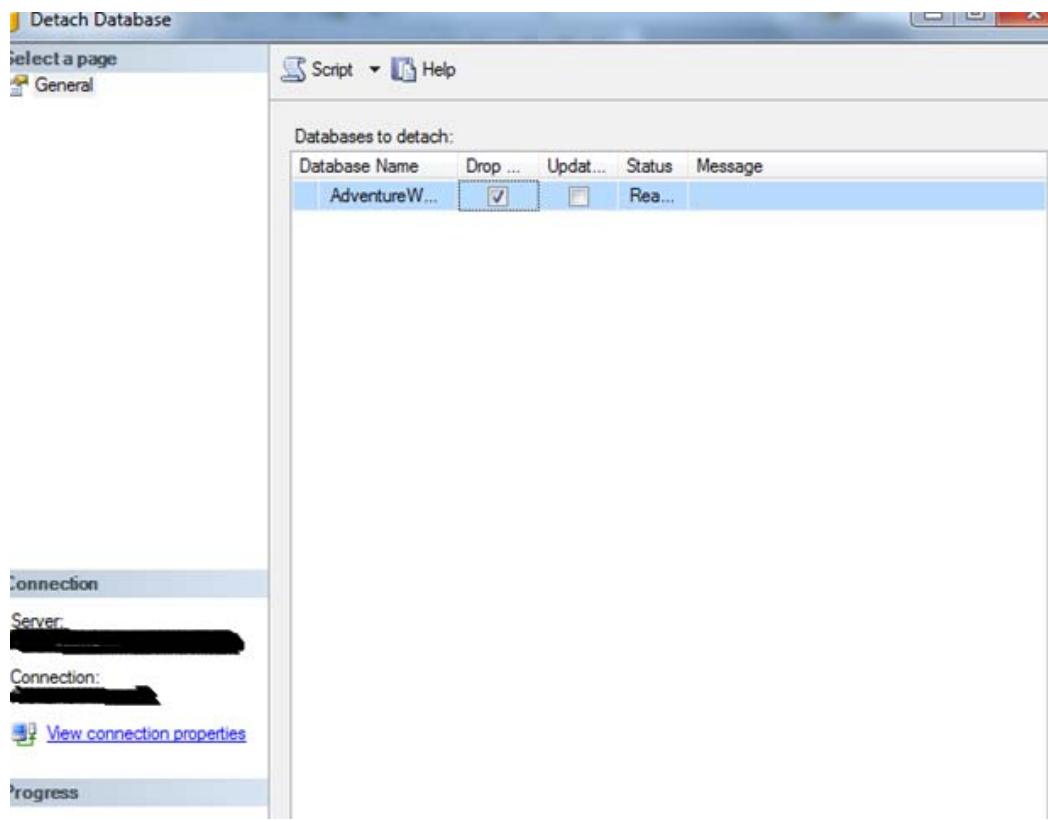
--Step 3 : Attach Database using following

Detach database in 2005/2008/R2:

Go to the server→place the cursor on particular database→tasks→select the detach option

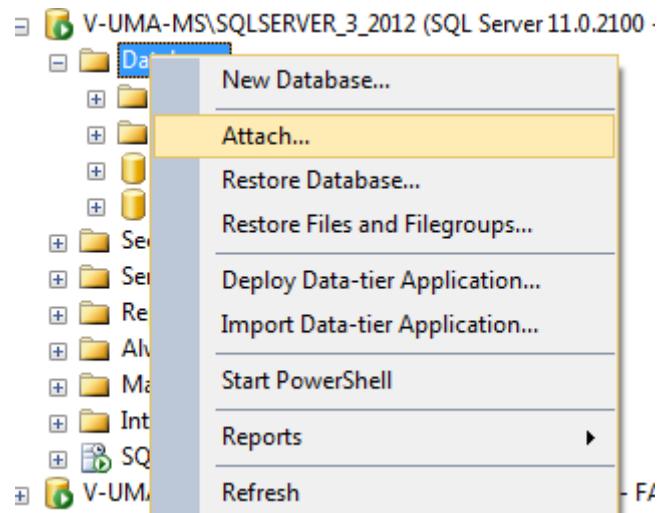


In General tab→check the drop existing connections option –click ok

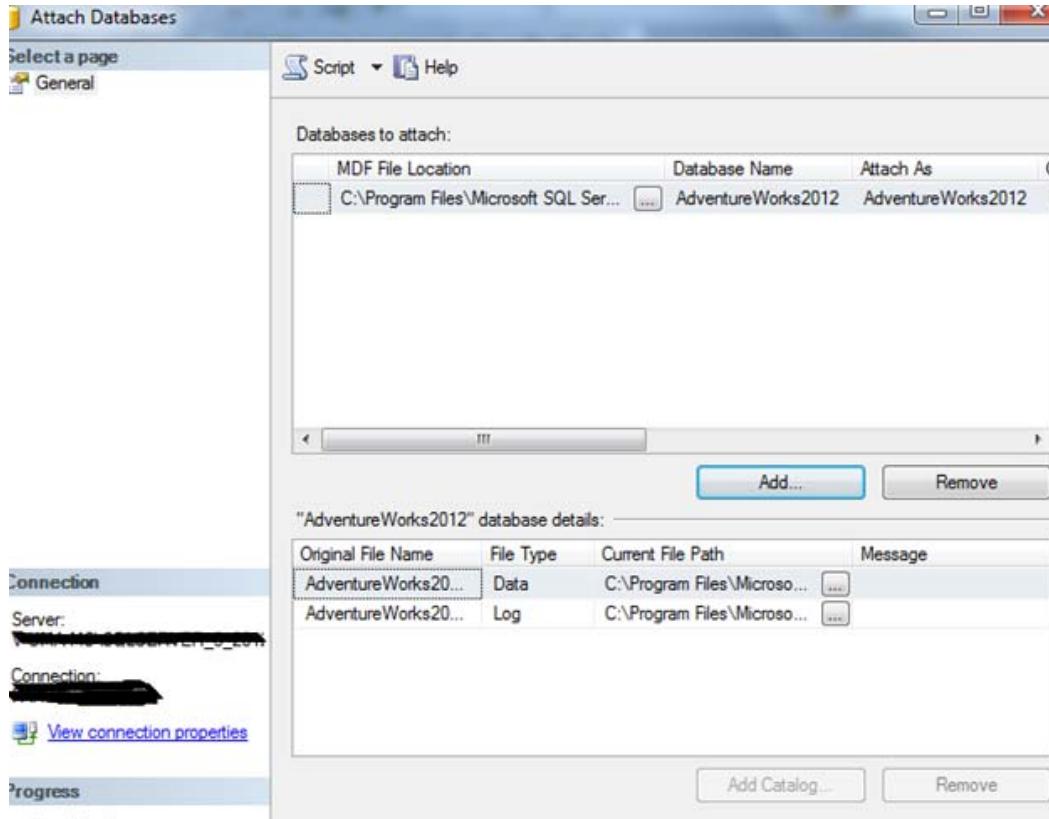


Attach Database in 2012:

Open Object Explorer in SQL Server and right click on Databases and select Attach menu item as you see in the below image.



Then go to → general tab → add the particular database data file location (it will automatically pick the log file location)—click ok



Now, select a file option and use Browse button to browse your MDF and LDF files as seen in below image.

That's it. Your database is attached now and ready to use.

Migrating Logins & Fixing Orphaned Users:

To Detect Orphaned Users:

To detect orphaned users, execute the following Transact-SQL statements:

```
USE <database_name>;
GO;
sp_change_users_login @Action='Report';
or
EXEC sp_change_users_login 'Report';
```

Note: **sp_change_users_login** cannot be used with SQL Server logins that are created from Windows.

To Resolve an Orphaned User

-----Script to find orphahend users-----

```
exec sp_change_users_login 'Report'
```

Use Master

```
SET NOCOUNT ON
```

```
SELECT 'EXEC sp_addlogin @loginame = "' + loginname + """  
, '@defdb = "' + dbname + """  
, '@deflanguage = "' + language + """  
, '@encryptopt = "skip_encryption""  
, '@passwd ='  
, cast(password AS varbinary(256))  
, '@sid ='  
, sid
```

```
FROM sys.syslogins
```

where loginname Not like 'NA%' and

loginname not like 'Builtin%' and loginname Not like 'sa'

Run the above script on Source server i.e. 2005/2008/R2, copy the result and execute on Destination server 2012.

Eg:--

```
EXEC sp_addlogin @loginame = 'CorpCommUser' , @defdb = 'CorpComm'  
@deflanguage = 'us_english' , @encryptopt = 'skip_encryption' , @passwd =  
0x01003F04413C64CEE4767BA2DD0053A02C6056640C4C88C24DFA , @sid =  
0xCEE1766A76520E43A98DCB141B031F7E
```

Mapping a database user to a new SQL Server login:

```
-- --Map database user MB-Sales to login MaryB.  
EXEC sp_change_users_login 'Update_One', 'MB-Sales', 'MaryB';
```

Automatically mapping a user to a login:

how to use Auto_Fix to map an existing user to a login of the same name, or to create the SQL Server login Mary that has the password B3r12-3x\$098f6 if the login Mary does not exist.

```
EXEC sp_change_users_login 'Auto_Fix', 'Mary', NULL, 'B3r12-3x$098f6';
```

The Import and Export Wizard

The import and export wizard was available even with SQL 2000 has remained an important tool for exporting from and importing into SQL Server data from many different kinds of data sources. It can also be used for transferring data between non-Microsoft data sources. In this I will show how to transfer data from MS Excel spreadsheet data to SQL Server 2008. In any of the transformations it is important to realize that data types used in data sources are not exactly the same and that there are differences to be reckoned with. The basic steps to take are to indicate the source of data and the destination to which it needs to be transferred. In order to match the differences some mappings may be necessary if the source and destination are not both SQL Servers.

The MS Excel file PrincetonTemp.xls used in this example is a simple spread sheet data that shows the temperature variations during a year and the maximum recorded temperature. The data type used for the column 'Month' is text and of the others are numbers.

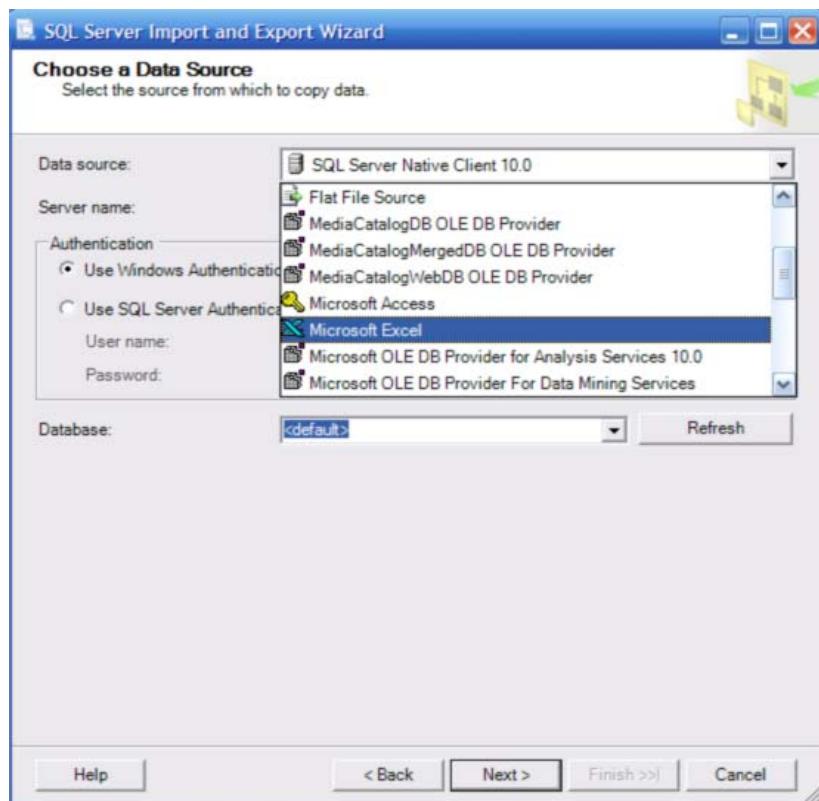
Id	Month	Temperature	RecordHigh
1	Jan	40	60
2	Feb	32	50
3	Mar	43	65
4	Apr	50	70
5	May	53	74
6	Jun	60	78
7	Jul	68	70
8	Aug	71	70
9	Sep	60	82
10	Oct	55	67
11	Nov	45	55
12	Dec	40	62

Invoke the Import and Export Wizard

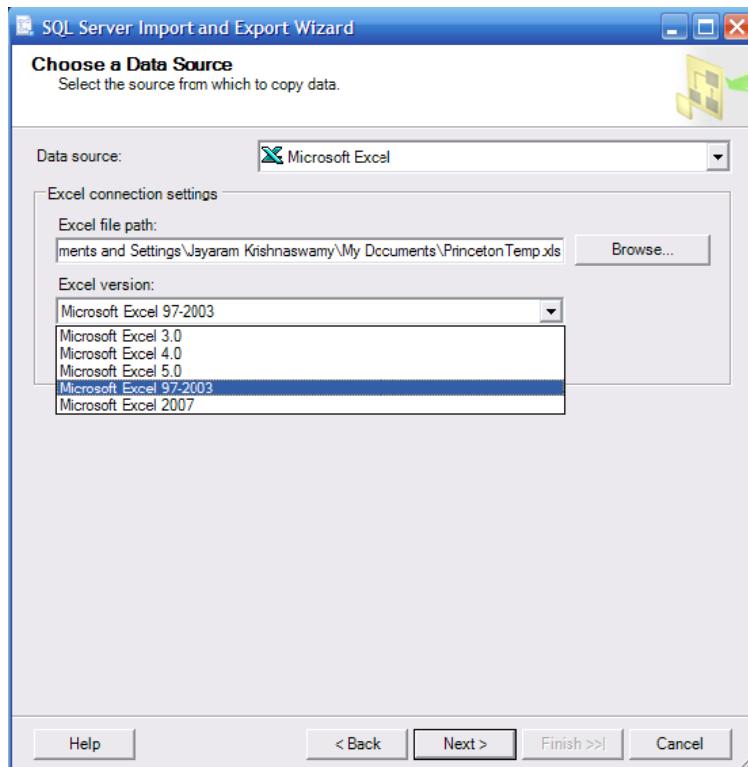
Bring up the Import and Export wizard from Start → All Programs → Microsoft SQL Server 2008 → Import and Export Data (32 bit). This pops-up the Welcome Wizard as shown. Make sure you read the explanations provided.



Click Next. The default page gets displayed. In the 'Choose a Data Source' page click on the handle along the data source and choose Microsoft Excel file as the data source as shown.

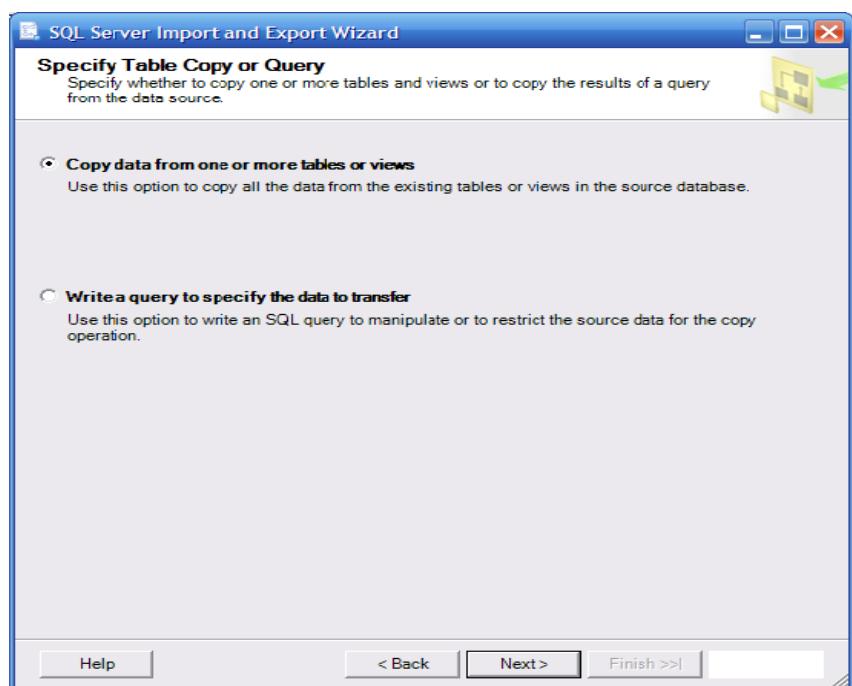
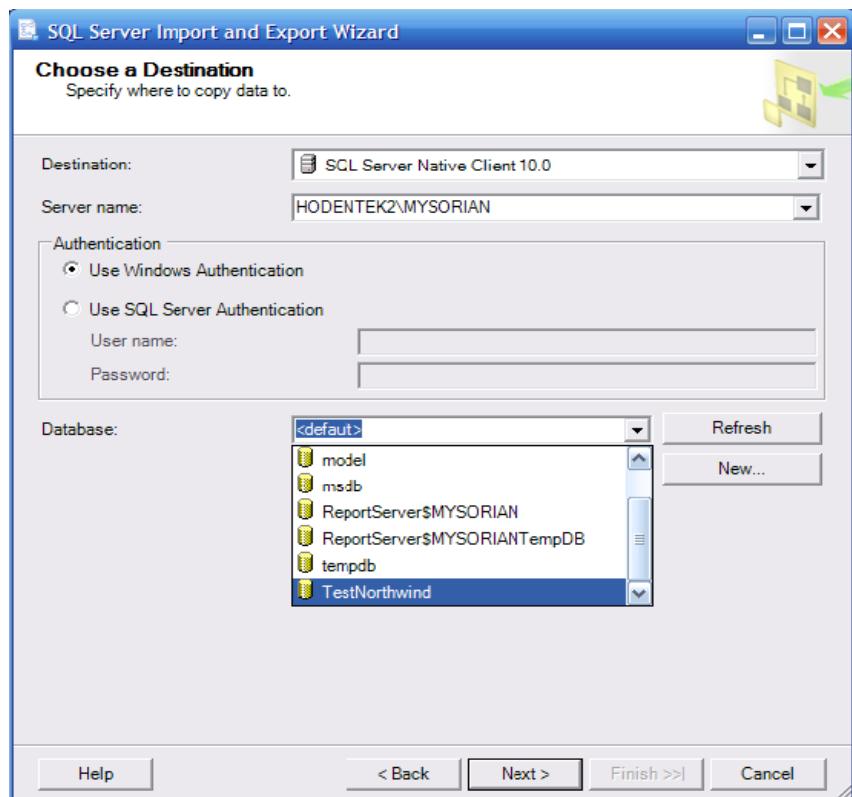


Click Next. The 'Select the source from which to copy data' shows up. Use the Browse...button to bring in the location information of PrincetonTemp.xls to the window as shown. The Excel version displayed by default (Microsoft Excel 97-2003) is proper for the MS Access version used in this article. Keep the 'First row has column names' option checked. Note that the MS Access 2007 is not supported.

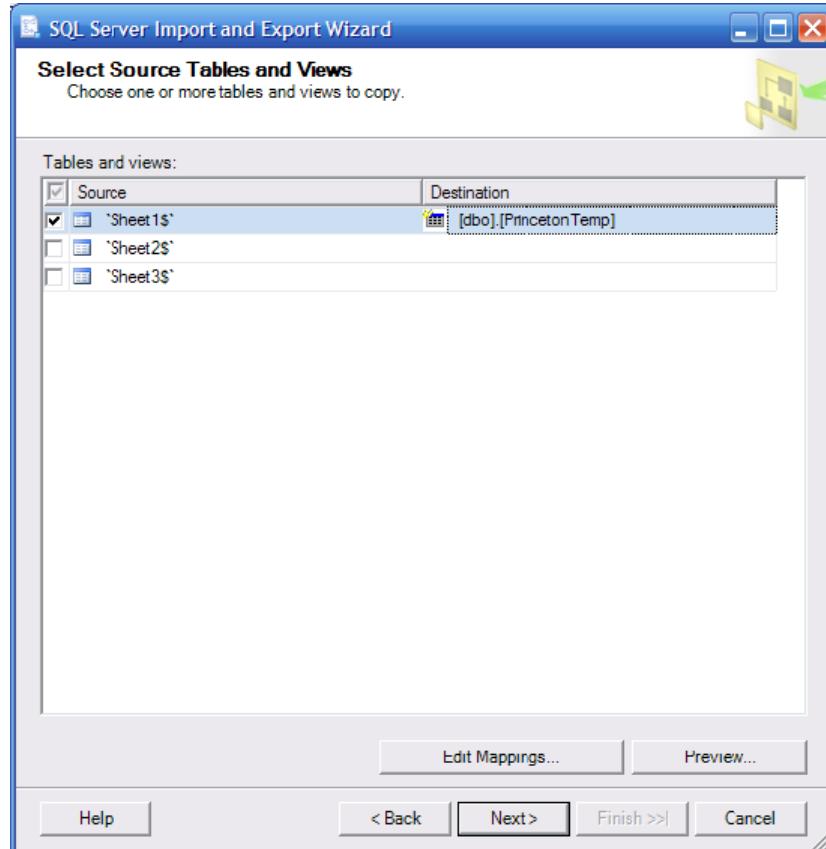


Click Next. The 'Choose the Destination' page shows up with SQL Server Native Client 10.0 as default and the resident server as Hodentek2\Mysorian. The server is configured for Windows authentication. Accept the defaults. In case your server is configured for SQL Server authentication you need to have the information ready. The database is displaying <default>. Click on the handle and choose a database from the drop-down list. Herein TestNorthwind is chosen. You can choose any database including the tempdb. Note that you can begin to create a new database as well, if you choose to do so by using the New...button.

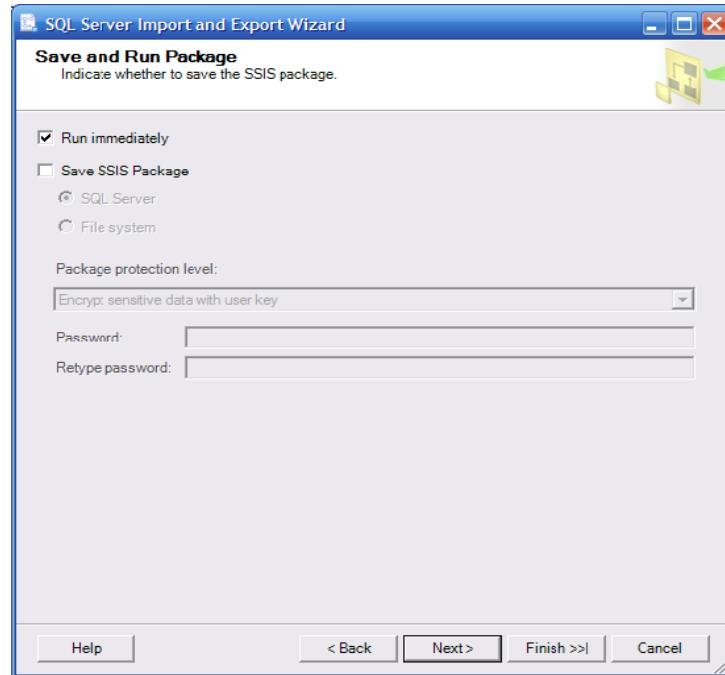
Click Next. The 'specify the Table Copy or Query' page of the wizard shows up. Since we are transferring only one table, accept the default option, 'Copy data from one or more tables or views'.



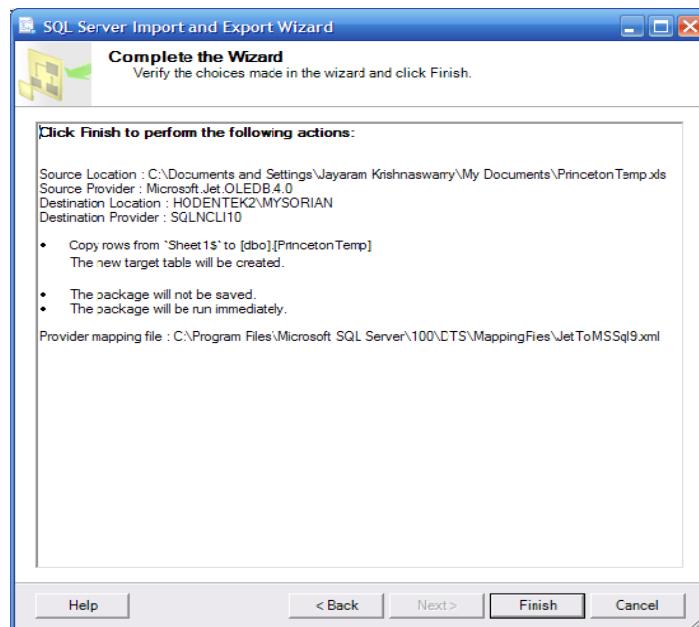
Click Next. Since sheet one has the data place check mark for 'Sheet1\$' as shown. Only Sheet1 has data in this XLS file. Modify the destination column to read dbo.PrincetonTemp instead of the default [dbo].[Sheet1\$] as shown.



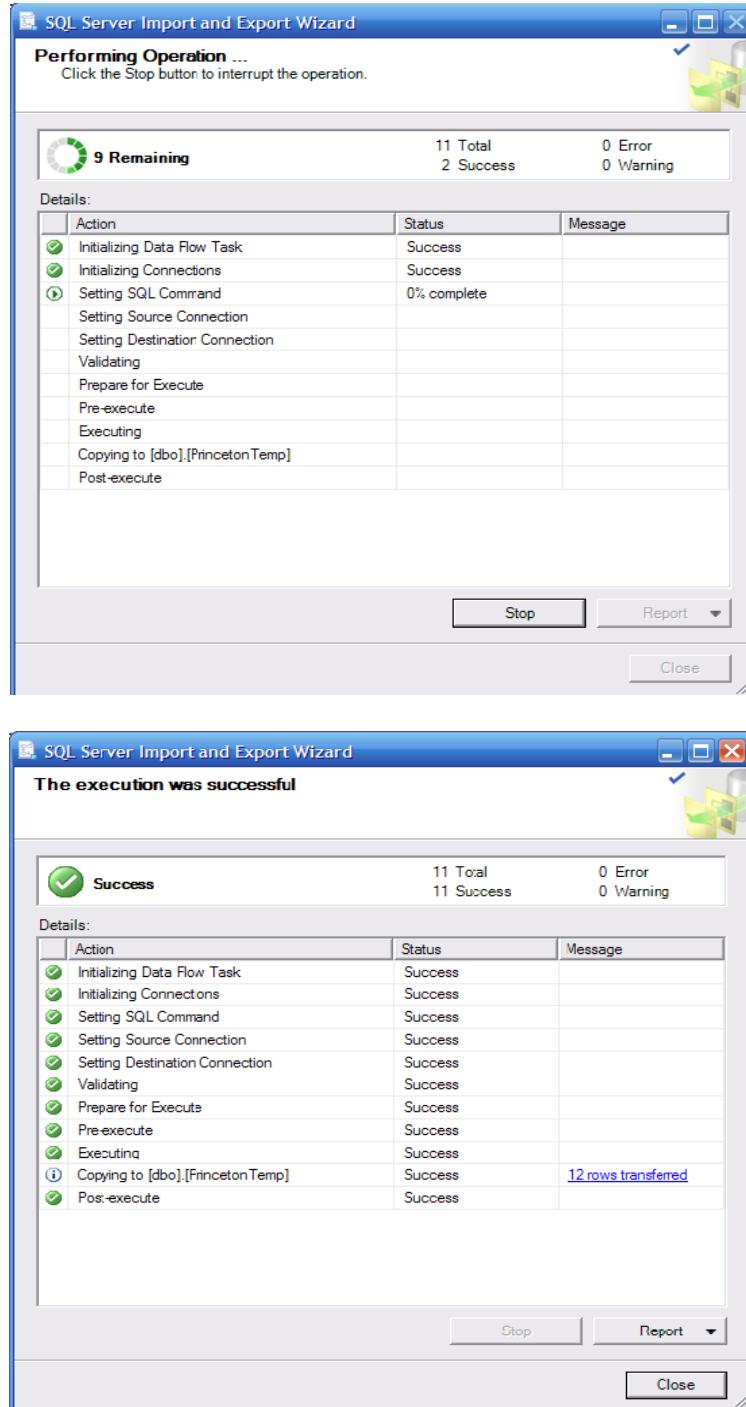
Click Next. In the 'Save and Run Package' page of the wizard accept the defaults shown. You could also save it as a package as well for later use.



Click Next. The 'Complete the Wizard' page gets displayed. Check if the information is correct (this is a summary of options you have chosen). If it is not correct you can hit the back button and move back to the pages you visited earlier in the reverse order.



Click Finish. The program starts running and you should see a progress window displaying 'Performing Operation...' as shown.

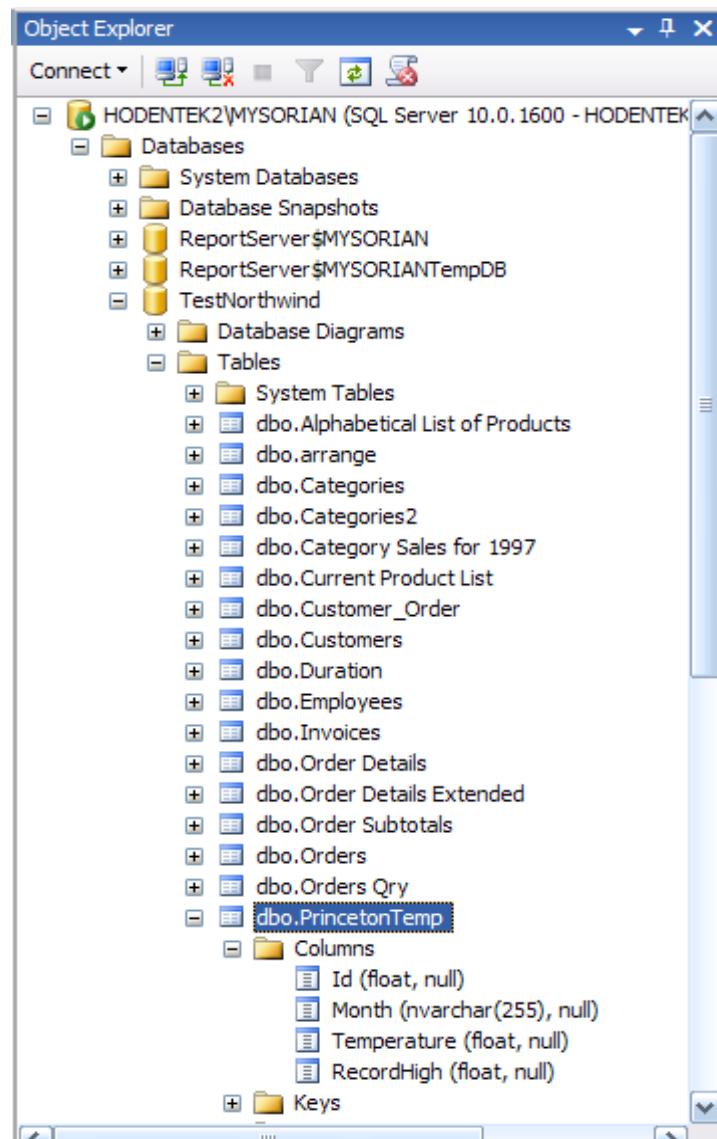


When the operation is completed you should see the following window and you can keep a copy of the report as to how the import was executed using the Report drop-down button.

The import in this case was successful as shown above. If there is an error there should be a hyperlink to the message in the Message column of the above window, presently the message is '12 rows transferred'. Close the wizard. The transfer is finished.

Verifying the import

Open the Microsoft SQL Server Management Studio and login to display the database engine using your Windows credentials. Expand the databases node and the TestNorthwind database node as shown.



Data type mismatch and the fix

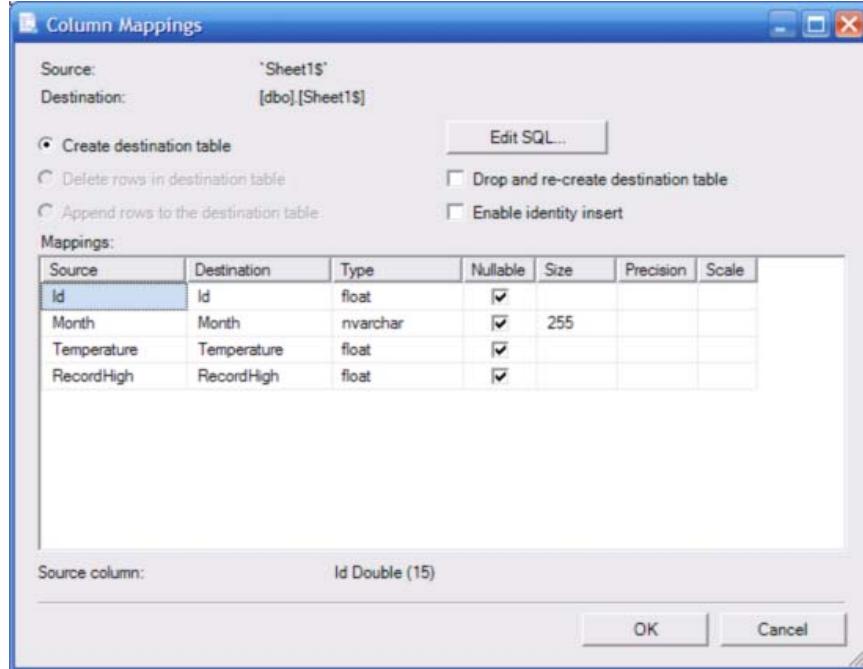
Also check if the data is brought in correctly as shown by right clicking the dbo.PrincetonTemp table and choose 'Select Top 1000 rows'. You can see that the Month names are all showing 'Null'. The 'text' data type in the XLS file became nvarchar type.

	Id	Month	Temperature	RecordHigh
1	1	NULL	40	60
2	2	NULL	32	50
3	3	NULL	43	65
4	4	NULL	50	70
5	5	NULL	53	74
6	6	NULL	60	78
7	7	NULL	68	70
8	8	NULL	71	70
9	9	NULL	60	82
10	10	NULL	55	67
11	11	NULL	45	55
12	12	NULL	40	62

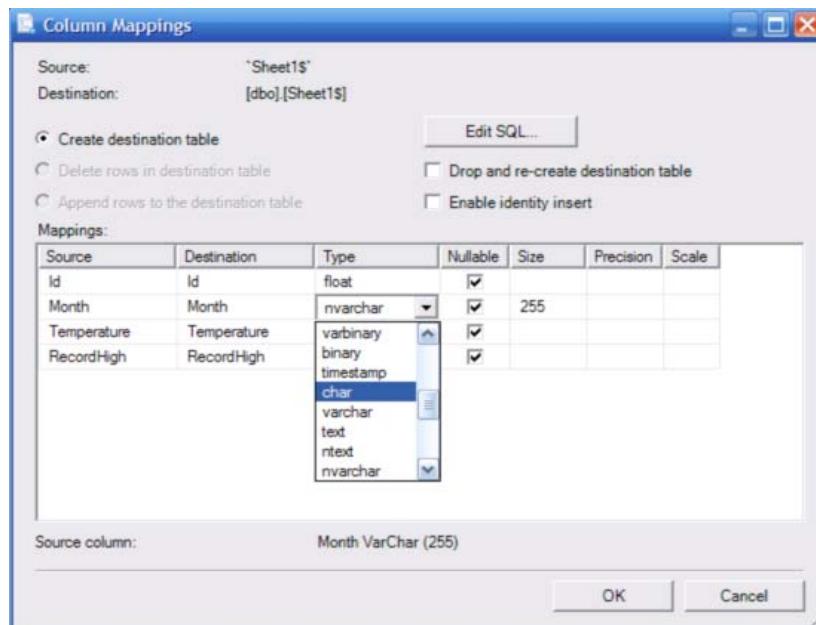
Modify the default mappings

In order to fix this, you can use either Drop table statement or right click and choose delete to delete the table from the TestNorthwind database. In the Delete Object window click OK. Refresh the Tables node by right clicking the Tables and choosing refresh. Now the imported table is gone.

Repeat the process that you did earlier and when you come to the stage shown in Figure.6 click on the table Edit Mappings...button. The Column Mappings page shows up as in the next figure

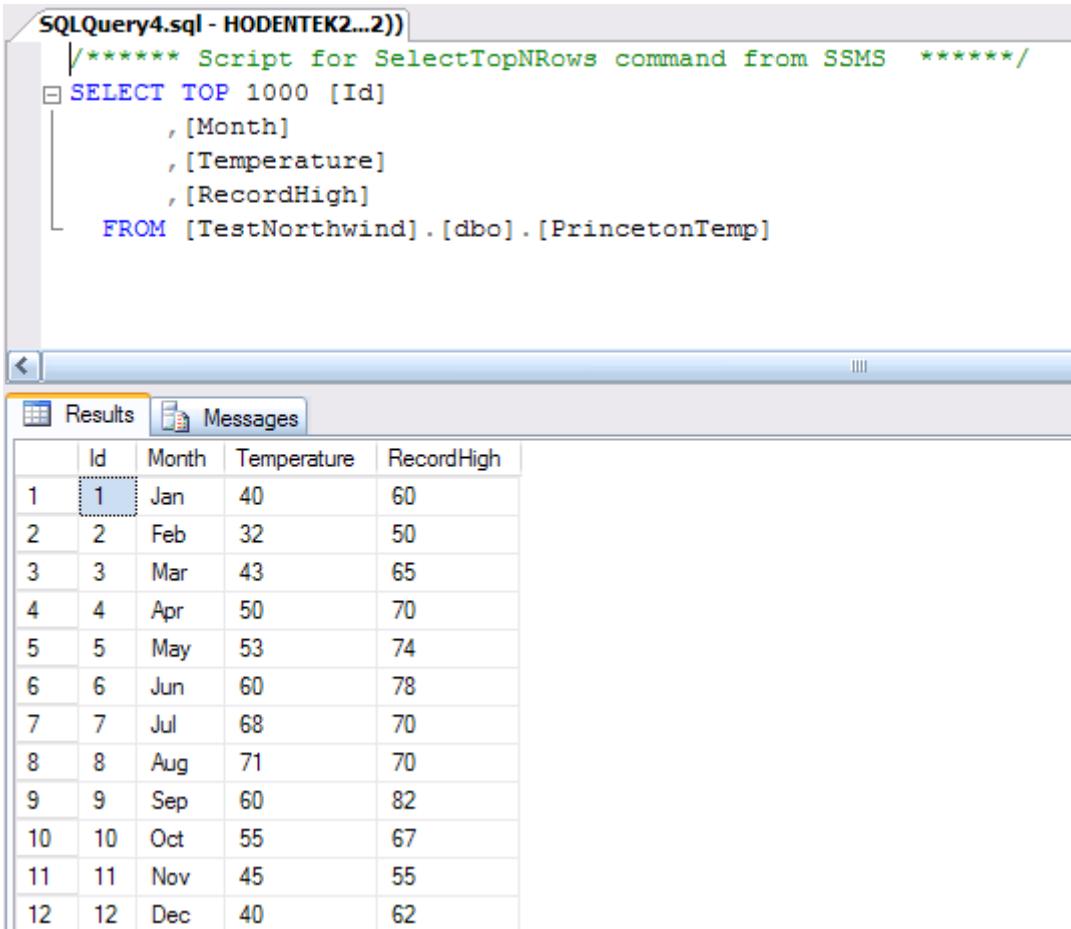


The month column data type for the destination is nvarchar (255). The Source had 'Text' as data type for this column. We need to cast it properly. Click on nvarchar in the 'Type' column and change it to 'char' as shown. Click OK. Change destination table name from [dbo].[Sheet1\$] to [dbo].[PrincetonTemp] as done previously. Click Next.



In the 'Save and Run Package' page accept defaults as previously. Click Next. The 'Complete the Wizard' page shows up. Click Finish. You get the wizard announcing 'The execution was successful'. Close the wizard.

Refresh the Tables node of the Northwind database in Management Studio. Now right click the PrincetonTemp and choose to select top 1000 rows as before. You will see that all the data in source is in the destination.



The screenshot shows the SQL Server Management Studio interface. The title bar says "SQLQuery4.sql - HODENTEK2...2)". The main pane displays a T-SQL script:

```
/* ***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP 1000 [Id]
    , [Month]
    , [Temperature]
    , [RecordHigh]
FROM [TestNorthwind].[dbo].[PrincetonTemp]
```

Below the script, the results pane is visible, showing a table with 12 rows of data:

	Id	Month	Temperature	RecordHigh
1	1	Jan	40	60
2	2	Feb	32	50
3	3	Mar	43	65
4	4	Apr	50	70
5	5	May	53	74
6	6	Jun	60	78
7	7	Jul	68	70
8	8	Aug	71	70
9	9	Sep	60	82
10	10	Oct	55	67
11	11	Nov	45	55
12	12	Dec	40	62

Case Study: Transferring Jobs and Logins using SSIS

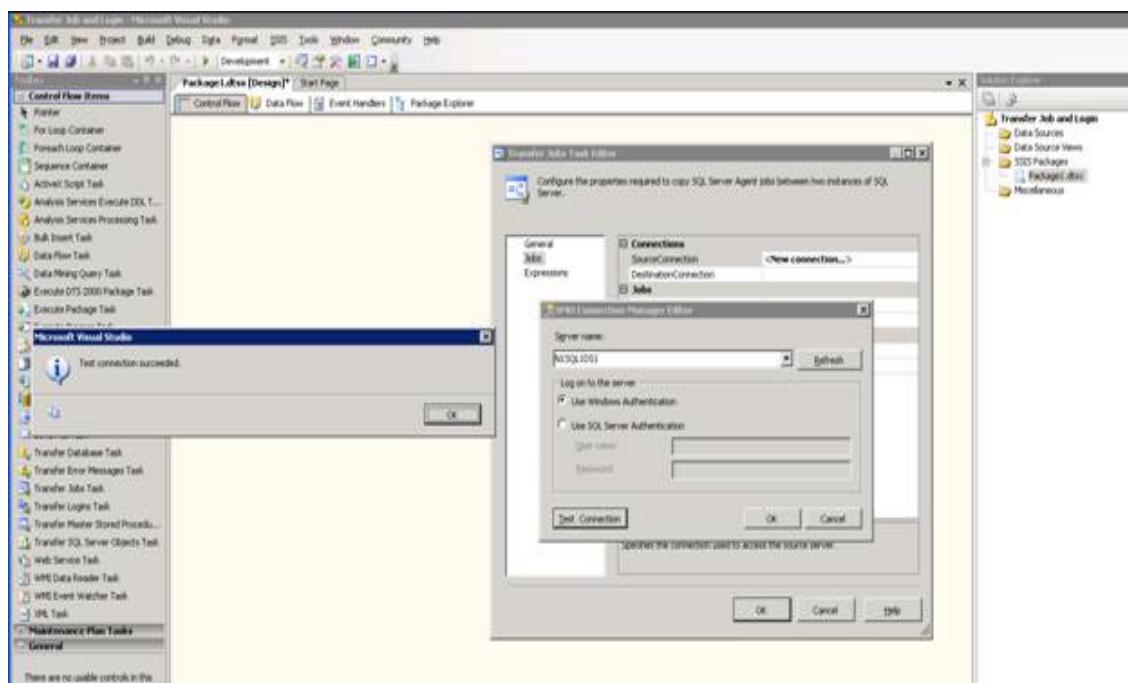
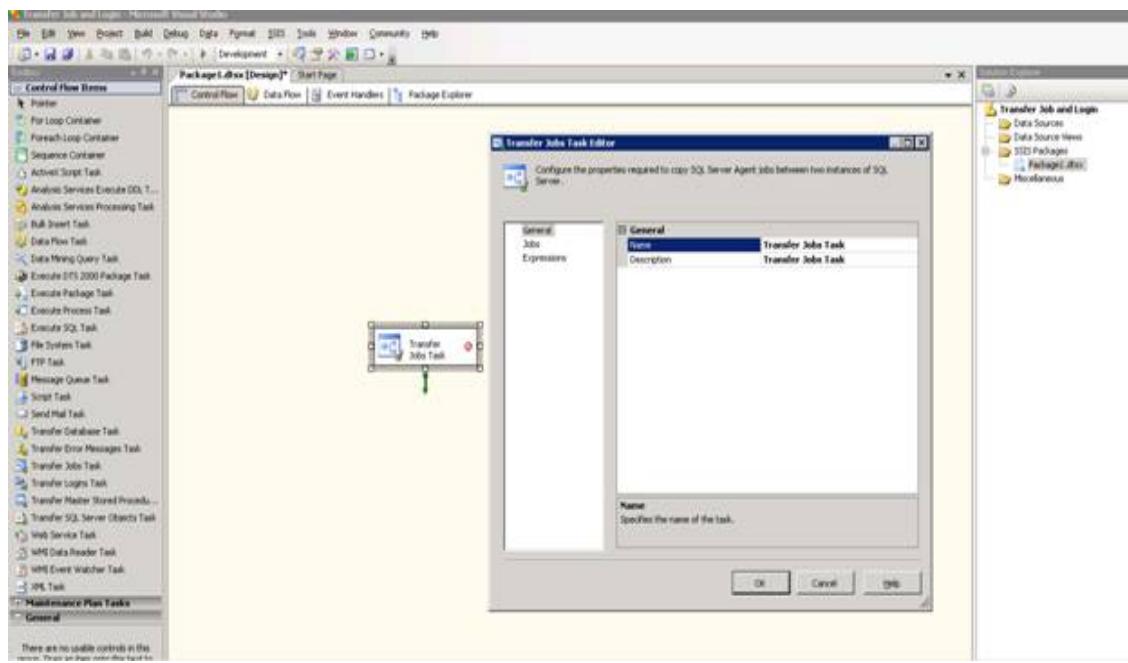
We can use SQL Server Integration Services to transfer the logins and jobs from SQL 2005 to another SQL 2005 or SQL 2008. This comes in handy when it's difficult to script each of the jobs. Firstly we need to create an SSIS package.

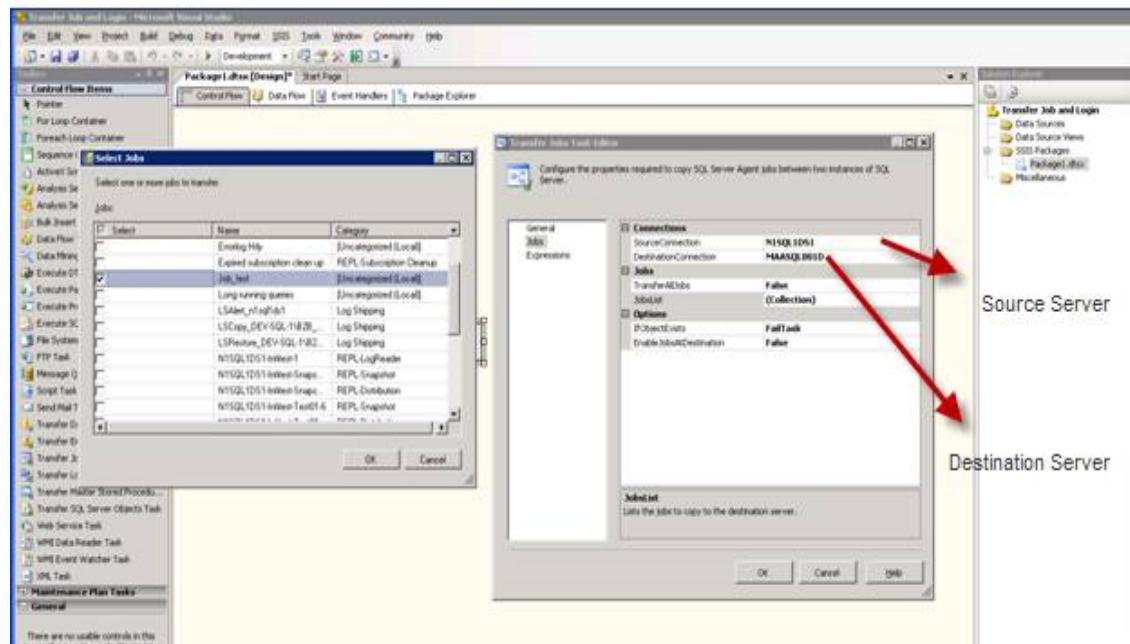
Open Business intelligence development studio – click file new project – select integration services project as the template and provide a suitable name for it.



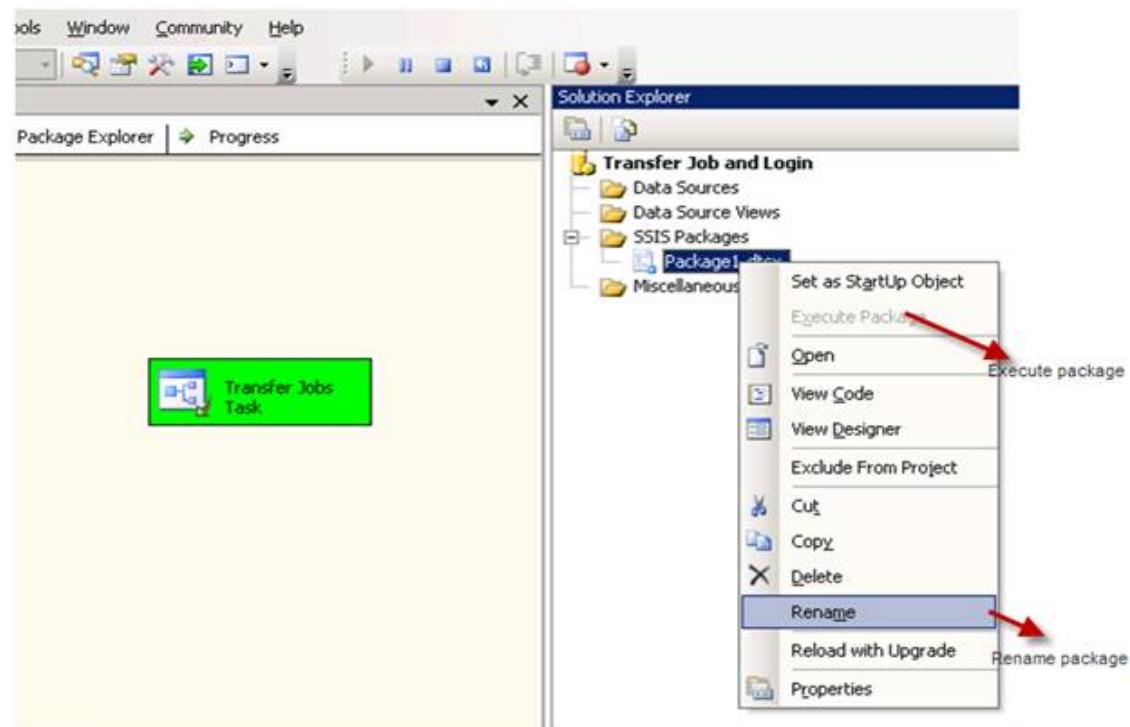
Open solution explorer using Ctrl + Alt +L or go to view and select solution explorer. Expand SSIS under solution explorer and expand SSIS packages – right click and select new SSIS package.

1. Drag and drop Transfer Job task from Control flow item into Control flow tab and double click it
2. In the general tab, give a specific name and description for the transfer job task
3. In Jobs tab, specify the source and destination server name under connections and test the connection as shown below
4. We also have an option to specify whether to transfer all the jobs or specific jobs. In my case I am setting the option as False so as to transfer only specific jobs. Then we need to select the list of jobs that needs to be transferred from the 'JobsList'
5. In the options, if the objects already exist (in our case it is the job) we need to specify what needs to be done. We can either specify it as FailTask if the job exists in destination or skip the object or overwrite it if it's already present
6. Finally we need to specify the option to enable the jobs in destination server after getting transferred

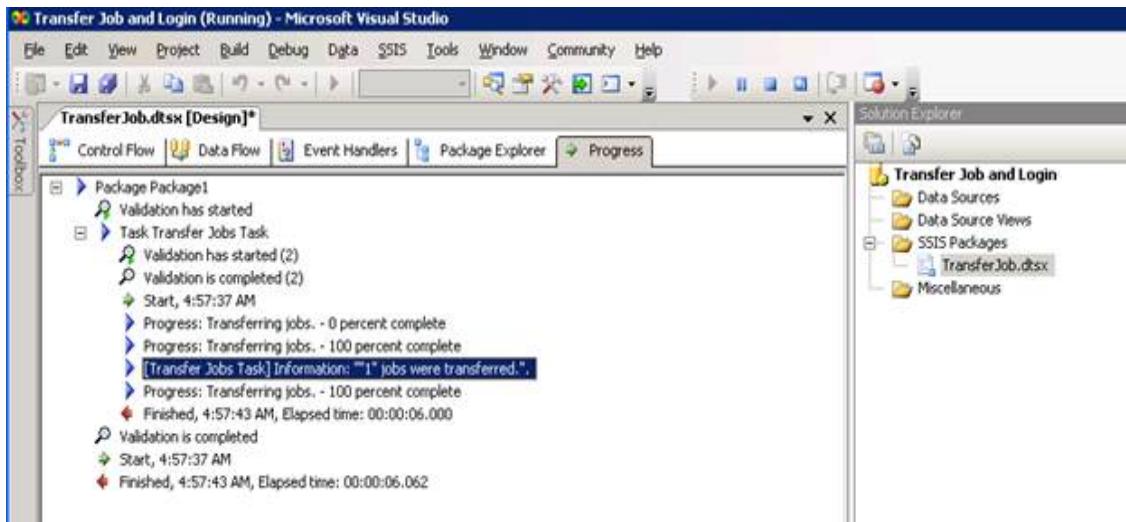




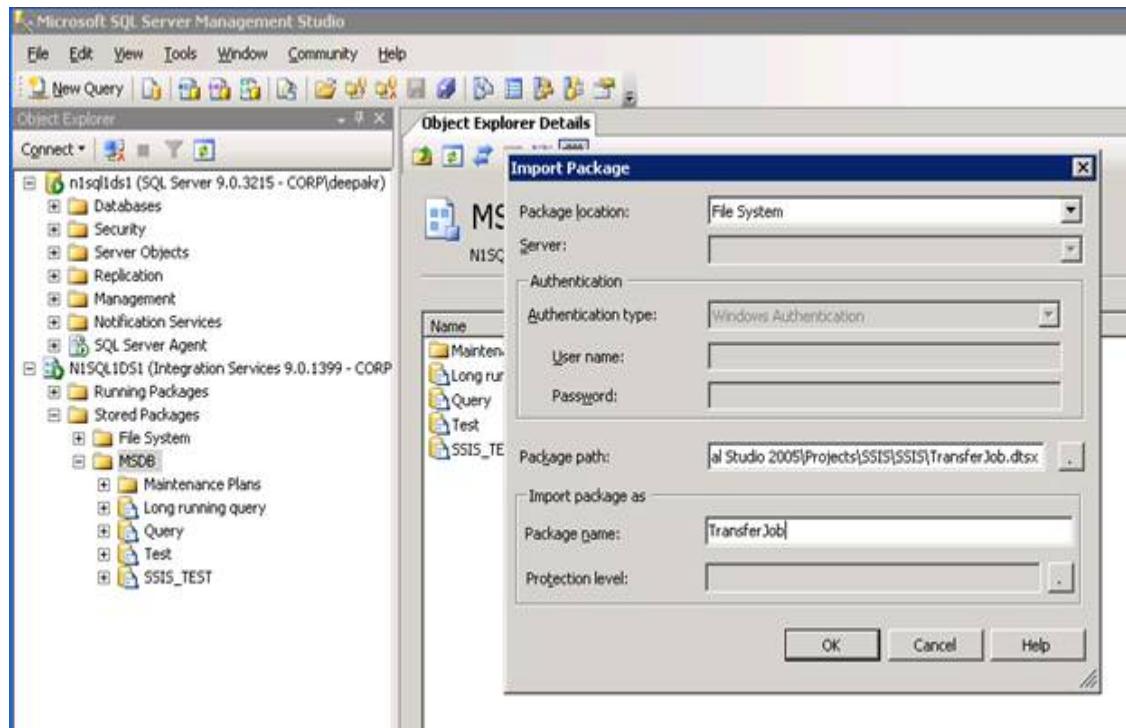
Once the above steps are completed we need to save the package (Ctrl + S) and execute it as shown below. We can view the status of the package execution under 'Progress' tab.



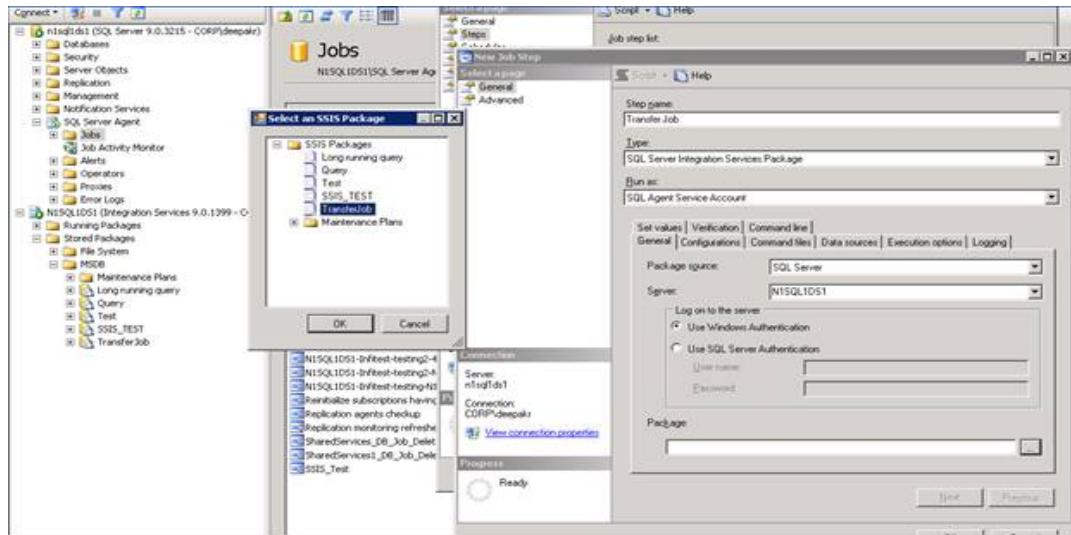
We also have an option to rename the package as indicated by the arrow marks.



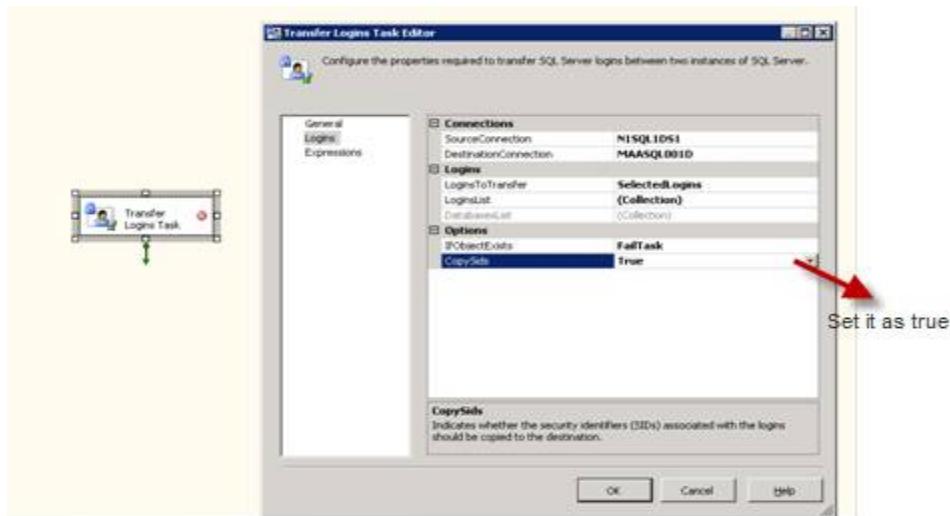
Once the package has executed successfully we can schedule the package as a job. First connect to Integration service – right click MSDB – Import package – select file system – specify the package path – and provide a name for the SSIS package.



Expand SQL Agent node and right click on Jobs – new job and select the ‘type’ as SQL Server Integration Services Package. Specify server name and package source as SQL Server. Click the ellipse button next to ‘package’ and select the package we created. Specify the schedule for the job as per your desire.



To create a package for Transfer login task, we need to create a new package and drag and drop the transfer login task from control flow items window and specify the source, destination server name. We need to choose the option to transfer all the logins or selected logins for specific databases. Also we have option to overwrite/skip/fail the package similar to the transfer job task. Finally we need to set the option ‘CopySids’ as true to transfer the security identifiers as well. We need to enable this option in for transferring logins while doing log shipping where there might be mismatched id due to restore operation.



Chapter – 11

SQL Server Security

SQL Server 2012 can be configured to work in either the Windows Authentication Mode or the SQL Server and Windows Authentication Mode, which is also frequently called Mixed Mode.

Windows Authentication Mode

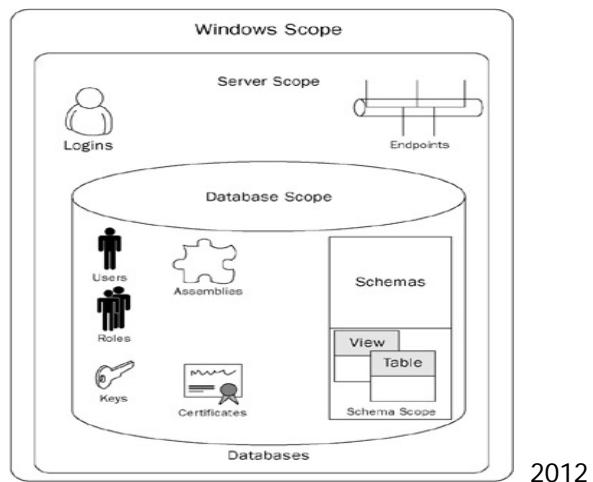
In Windows Authentication Mode only logins for valid Windows users are allowed to connect to SQL Server. In this authentication mode, SQL Server “trusts” the Windows, Windows Domain, or Active Directory security subsystem to have validated the account credentials. No SQL Server accounts are allowed to connect. They can be created, but they cannot be used for login access.

SQL Server and Windows Authentication Mode (Mixed Mode)

In SQL Server Mode and Windows Authentication Mode or Mixed Mode, valid Windows accounts and standard SQL Server logins are permitted to connect to the server. SQL Server logins are validated by supplying a username and password. Windows accounts are still trusted by SQL Server. The chief advantage of Mixed Mode is the ability of non-Windows accounts (such as UNIX) or Internet clients to connect to SQL Server.

Security Architecture:

There are different levels in your security hierarchy. The below figure outlines the different levels of security you need to manage. At the Windows scope, you create Windows users and groups, manage the files and services needed by the SQL Server, as well as the behavior of the server itself. In the server scope, you manage logins, endpoints, and databases. In the database scope, you work with users, keys, certificates, roles, assemblies, and other objects. Also in this scope are schemas, which contain your next set of securables. Finally, within the schema scope, you have data types, XML schema collections, and objects. These objects include your tables, views, stored procedures, and more.



Microsoft SQL Server 2012 includes a number of server-level roles that are available to simplify management (and the delegation of management) for SQL logins. These are often referred to as fixed 2012 because membership is the only thing you can really change about these roles. The fixed 2012 are designed to allow you to automatically assign a common set of permissions to a login, based upon the purpose of the role.

Role	Description
sysadmin	Members have full administrative access to the SQL Server, and can perform any action. By default, this includes the BUILTIN\Administrators group.
serveradmin	Members can change server-wide configurations and shut down the server.
securityadmin	Members can manage SQL logins, including changing and resetting passwords as needed, as well as managing GRANT, REVOKE, and DENY permissions at the server and database levels.
dbcreator	Members can create, drop, alter, and restore any database for the server.
diskadmin	Members can manage disk files for the server and all databases.
processadmin	Members can manage and terminate processes on the SQL Server.
setupadmin	Members can add and remove linked servers.
bulkadmin	Members of this role can execute the BULK INSERT statement for any database on the server.

Server Securables

Dozens of instance-wide privileges can be granted at the instance level. These include connecting and managing the various endpoints within the SQL Server instance, managing the logins within the SQL Server instance, various instance-wide settings, the AlwaysOn Availability Groups, and the user-defined server roles that were introduced in SQL Server 2012.

The biggest difference between instance-wide privileges and database-wide privileges is that instance wide privileges are granted directly to the login, whereas database-wide privileges are granted to users, and these users are then mapped to logins.

Now take a look at the different endpoint privileges available to the various objects within the scope of the SQL Server instance.

Endpoints

You can manage five privileges for each endpoint within the SQL Server instance:

1. **Alter:** The Alter right enables the login that has the privilege to make configuration changes to the endpoint.
2. **Connect:** The Connect privilege enables the user that has the privilege to connect to the endpoint; by default all logins are granted the privilege to connect to the default endpoints.
3. **Control:** The Control privilege grants the other four privileges.
4. **Take Ownership:** The Take Ownership privilege grants the login the ability to become the owner of the endpoint.

5. View Definition: The View Definition privilege grants the login the ability to view the configuration of the endpoint without being able to make changes to the endpoint.

Logins

You can manage four privileges for each login within the SQL Server instance:

1. **Alter:** The Alter privilege enables the login that has been given that right to make changes to the second login that the right was granted to. For example, if there were two logins named login1 and login2, login1 can be granted the ability to alter login2. Altering a login gives the granted login the ability to change the password, default database, default language, and so on of the grantee login.
2. **Control:** The Control privilege grants the granted user the other three privileges to the grantee login. For example, if there were two logins named login1 and login2, login1 can be granted the ability to control login2.
3. **Impersonate:** The Impersonate privilege grants the granted user the ability to use the EXECUTE AS syntax specifying the grantee login the ability to execute code as the grantee login.
4. **View Definition:** The View Definition privilege grants the granted user the ability to view the configuration of the grantee login.

Availability Groups

Availability Groups have four rights that can be granted to user-defined server roles:

1. **Alter:** The Alter privilege enables the user that has been assigned the privilege to make changes to the AlwaysOn Availability Group.
2. **Control:** The Control privilege grants the other three privileges to the user.
3. **Take Ownership:** The Take Ownership privilege enables the user who has been assigned the privilege the ability to change the ownership of the availability group.
4. **View Definition:** The View Definition privilege enables the user who has been granted the right the ability to view the definition of the availability group.

User Defined Server Roles

SQL Server 2012 introduces user defined server roles. User defined server roles are similar to fixed server roles except that they are created by the SQL Server administrator and not by Microsoft. The user defined server roles can be made members of any other server role, either fixed or user defined. Any server-wide right that can be granted to a login can be granted to a user defined server role.

Four privileges can be granted to a login for each user defined server role:

1. Alter: The Alter privilege grants the login the privilege to alter the user-defined server role. This includes adding other logins as members of the fixed server roles.

2. Control: The Control privilege grants the other three privileges.
3. Take Ownership: The Take Ownership privilege grants the login the ability to set himself as the owner of the user-defined server role.
4. View Definition: The View Definition privilege grants the login the ability to view the user defined server role without having the ability to alter the user-defined server role.

Instance-Wide Settings

PRIVILEGE NAME	PRIVILEGE DEFINITION
Administrator bulk options	Enables the user to bulk insert data into the SQL Server instance using the BULK INSERT statement, the bcp command-line application, and the OPENROWSET(BULK) operation.
Alter any availability group	Grants the user the right to alter or failover any Always On availability group. By granting this privilege, the login is also granted to the Create Availability Group privilege.
Alter any connection	Grants the user the right to kill any user connection.
Alter any credential	Grants the user the right to alter any credential within the database instance.
Alter any database	Grants the user the right to change the database options for any database within the database instance. By having this right granted, the user is also granted the Create Any Database privilege.
Alter any endpoint	Grants the user the right to alter any endpoint that has been created on the SQL Server instance. By having this right granted, the user is also granted the Create Any Endpoint privilege.
Alter any event notification	Grants the user the right to alter any event notification that has been created within the SQL Server instance. By having this right granted, the login is also granted the Create Trace Event Notification privilege.
Alter any linked server	Grants the user the right to alter any linked server within the SQL Server instance.
Alter any login	Grants the user the right to alter any login within the instance.
Alter any server audit	Grants the user the right to change any server audit specification.
Alter any server role	Grants the user the right to change the user-defined server roles within the SQL Server instance.
Alter resources	Grants the user the right to change system resources.
Alter server state	Grants the user the right to change the server state. By having this right granted, the login is also granted the View Server State right.

PRIVILEGE NAME	PRIVILEGE DEFINITION
Alter settings	Grants the user the right to change instance-wide settings.
Alter trace	Grants the user the right to change other user's profiler and server side traces.
Authenticate server	Grants the user the right to authenticate against the SQL Server instance.
Connect SQL	Grants the user the right to connect to the SQL Server instance.
Control server	Grants a superset of instance level rights: Administrator bulk options, Alter Any Availability Group, Alter Any Connection, Alter Any Credential, Alter Any Database, Alter Any Endpoint, Alter Any Event Notification, Alter Any Linked Server, Alter Any Login, Alter Any Server Audit, Alter Any Server Role, Alter Resources, Alter Server State, Alter Settings, Alter Trace, Authenticate Server, Connect SQL, External Access Assembly, Shutdown, Unsafe Assembly, and View Any Definition.
Create any database	Enables the user to create a new database or to restore a database from backup.
Create availability group	Enables the user to create a new Always On availability group.
Create DDL event notification	Grants the user the privilege to create a DDL trigger.
Create endpoint	Grants the user the privilege to create a SQL Server endpoint.
Create server role	Grants the user the privilege to create a user defined server role.
Create trace event notification	Grants the user the privilege to create a trace event notification.
External access assembly	Grants the user the privilege to create an assembly that requires the external access setting.
Shutdown	Grants the user the privilege to shut down the SQL Server instance by using the SHUTDOWN T-SQL statement.
Unsafe assembly	Grants the user the privilege to create an assembly that requires the unsafe setting.
View any database	Grants the user the privilege to view the definition of any database within the SQL Server instance.
View any definition	Grants the user the privilege to view the definition of any object within the SQL Server instance. By granting this right, the login is also granted the View Any Database privilege.
View server state	Grants the user the privilege to view the server state objects. These server state objects include the SQL Servers dynamic management views and functions.

Database Securables:

Objects of various kinds exist within each SQL Server database, all of which have their own permissions. These permissions grant specific users rights to those objects so that the users can perform the functions needed to complete their tasks. It is a best practice to grant the users who will be using the database the minimum permissions needed to complete her job. This is done so that the users don't have rights to objects or data within the database that they don't need. An added benefit of this practice is to prevent someone who breaks into the database from gaining access to more secure data.

Database Permissions

Permissions can be granted against the database itself. Some of these rights are specific to the database level while some cascade down to objects within the database such as tables, views, and stored procedures.

Tables and Views

When dealing with tables and views, ten different rights can be granted to specific users or user defined database roles.

RIGHT	DEFINITION
Alter	Grants the user the ability to change the schema of the object.
Control	Grants the user all other rights on the object.
Delete	Grants the user the ability to delete data from the object.
Insert	Grants the user the ability to insert data into the table.
References	Grants the user the ability to create a foreign key on the table. This right does not apply to views.
Select	Grants the user the ability to select the data from the object.
Take Ownership	Grants the user the ability to change the ownership of the object.
Update	Grants the user the ability to change data within the table.
View Change Tracking	Grants the user the ability to view the change tracking information for the object in question.
View Definition	Grants the user the ability to view the schema design of the object.

Stored Procedures and Functions

Stored procedures, functions, and most other database objects within SQL Server contain only five permissions that can be granted to users or roles.

PERMISSION	DEFINITION
Alter	Enables the user to change the schema of the database object the right was granted to
Control	Grants the user the other rights to the object
Execute	Enables the user to execute the object
Take Ownership	Enables the user to change the owner of the object
View Definition	Enables the user to view the schema of the object without having the ability to change the object

Creating Logins in Management Studio

To create logins from Management Studio, follow these steps:

1. From the Object Explorer, expand your server.
2. Expand the Security folder.
3. Right-click Logins and select New Login.
4. In the New Login dialog box (see Figure), either type the Login name you want to add, or click the Search button to browse for a Windows account.

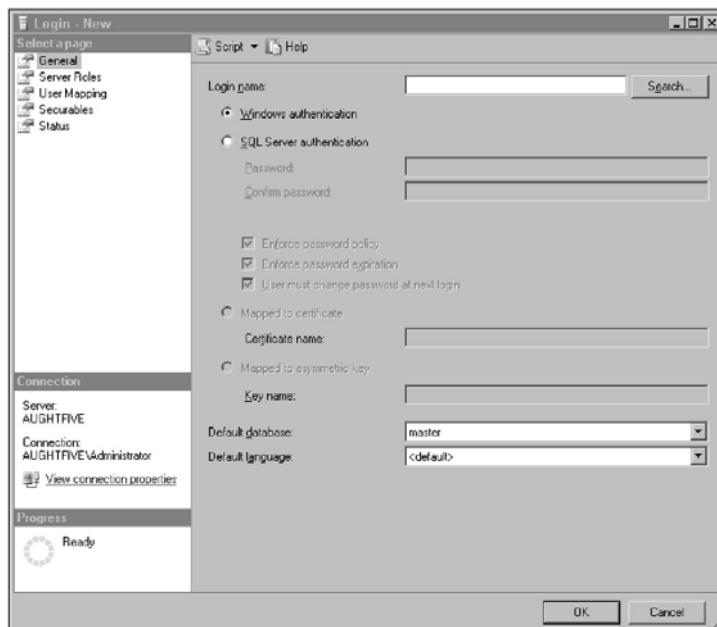


Figure: New Login dialog box

5. If you are creating a SQL Login, select the “SQL Server authentication” radio button.

6. Also, when you select "SQL Server authentication," you can choose to not enforce the password policies.
7. You may also want to change the user's default database and language.

Credentials

Microsoft SQL Server 2008/2012 includes a new feature for mapping SQL Server logins to external Windows accounts. This can be extremely useful if you need to allow SQL Server logins to interact with the resources outside the scope of the SQL Server itself (such as a linked server or a local file system). They can also be used with assemblies that are configured for EXTERNAL_ACCESS.

Credentials can be configured as a one-to-one mapping, or a many-to-one mapping, allowing multiple SQL Server logins to use one shared Windows account for external access. Logins, however, can only be associated with one credential at a time.

Creating a New Credential

To create a new credential, follow these steps:

1. In Object Explorer, expand your server.
2. Expand the Security folder.
3. Right-click Credentials and select New Credential.
4. Type a name for the credential (see [Figure](#)).

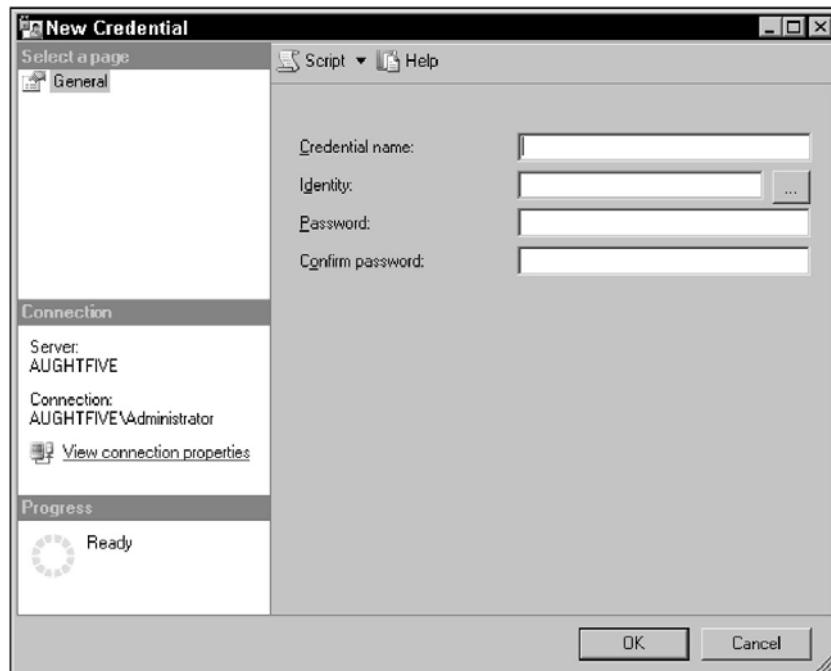


Figure: New Credential properties screen

5. Either type the name of a Windows account, or click the “...” button to browse for an account.
6. Enter the password for the account.
7. Re-enter the password to confirm.
8. Click OK.

View	Description
sys.server_principals	Returns information about all server-level principals.
sys.sql_logins	Returns information about SQL Server logins.
sys.server_role_members	Returns the role ID and member ID for each member of a server role.

Fixed Database Roles

Every SQL database has a list of fixed database roles that allow you to delegate permissions to users as necessary. As with the fixed 2012, membership is the only thing you can change about these roles. It is important to know how and when to use these roles.

The following table shows the fixed database roles.

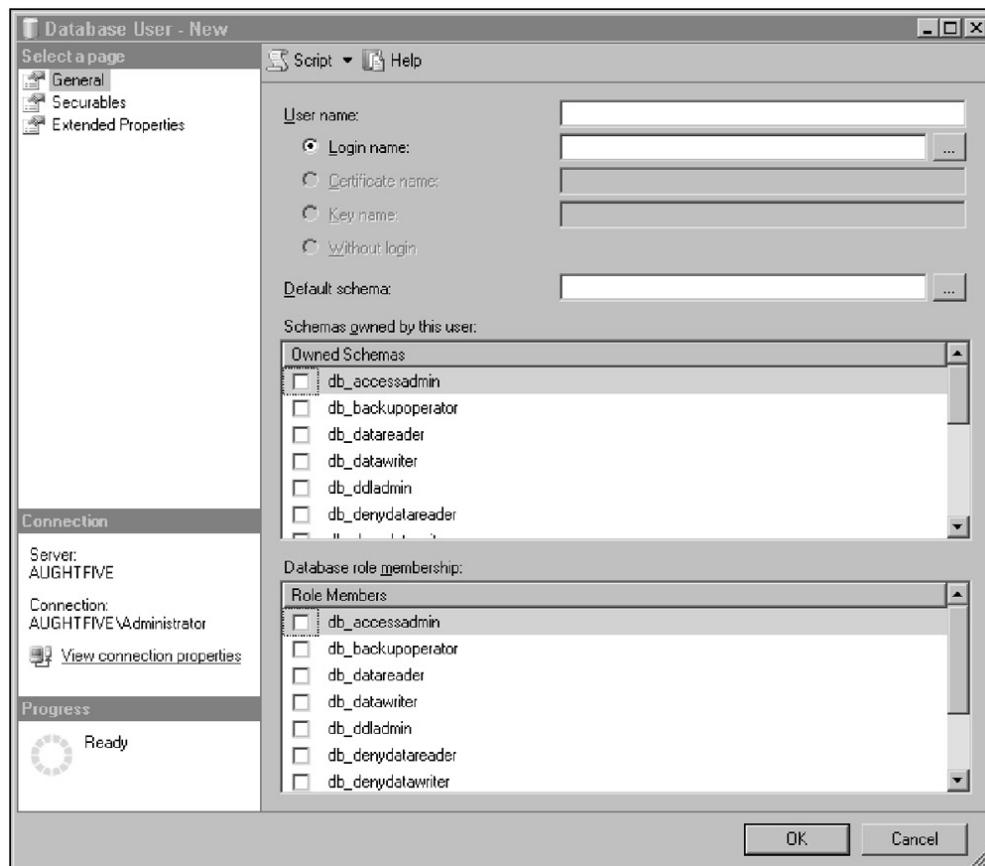
Role	Description
db_accessadmin	This role can add or remove access for Windows logins, Windows groups, and SQL Server logins.
db_backupoperator	This role has the right to back up the database.
db_datareader	Members of this role can read data from all user tables.
db_datawriter	Members of this role can write data from all user tables.
db_ddladmin	This role can execute data definition language (DDL) statements for any object in the database.
db_denydatareader	This role is explicitly excluded from being able to read from any user table with the database.
db_denydatawriter	This role is explicitly excluded from being able to write to any table in the database.
db_owner	Members of this role can perform any activity within the database. New to SQL Server 2012 is the ability for this role to drop the database from the server. The dbo user is automatically a member of this role.
db_securityadmin	This role can manage permissions and role membership within the database.
public	Membership in the public role is automatic. Permissions that apply to the public role apply to everyone who accesses the database.

Note that the fixed database roles include db_denydatareader and db_denydatawriter. These roles explicitly deny read or write access to user tables in the database, and should be used sparingly. Deny permissions are authoritative and cannot be overridden.

Database Users

Database users are another component of the security model employed by Microsoft SQL Server 2012. Users are granted access to database securables, either directly or through membership in one or more database roles. Users are also associated with ownership of objects such as tables, views, and stored procedures.

When a login is created, unless it is a member of a fixed server role with database administrative privileges, that login has no explicit permissions within the various databases attached to the server. When this happens, the login is associated with the guest database user, and inherits the permissions of that user account.



Permissions

Permissions are at the heart of security in SQL Server 2012. In the previous section, you looked at the different types of objects that can be created to help manage security by identifying to whom you can grant access. In this section, you look at permissions that can be applied to the different resources in SQL Server.

To begin with, you should understand there are essentially three permission states that exist: GRANT, GRANT_W_GRANT, and DENY. In addition, when a principal does not have an explicit

permission defined, the permission is considered “revoked.” The following table shows the different permission states.

Permission	Description
GRANT	This state means that you have been given the right to perform this action, or interact with this resource based on what the actual permission is.
GRANT_W_GRANT	Not only can you perform this action, but you also have the right to give others the ability to perform this action.
DENY	You cannot perform this action. This is also known as an “explicit deny,” because nothing will allow you to perform this action.
REVOKE	This is not really a permission state as much as it is the absence of a permission state. Revoked permissions will not show up in a sysprotects table or sys.sysprotects view, and are considered an “implicit deny.” The idea is that if you haven’t been granted this permission, either directly or through membership in a role with that permission, it is safe to assume you shouldn’t be doing that. Therefore, you will not be doing that.

To control permission states, you can use the Object Explorer or Transact-SQL. The three commands that you can use to control permission states are GRANT, REVOKE, and DENY, which are described in the following table.

Command	Description
GRANT	This command allows you to grant the right to perform and action or interact with an object in a specific way. The GRANT statement includes the WITH GRANT OPTION option, which also allows the grantee the ability to become a grantor of this permission.
REVOKE	This command removes any explicit permission granted to the grantee, either grant or deny. Revoked permissions will remove the ability to perform that task. Remember that if the user is a member of another role, they may still have the ability to perform the action, unless an explicit deny is specified.
DENY	This command creates an entry that will prevent the user from performing the action. Denied permissions cannot be overridden by grant permissions.

What is Contained Database?

Before going to explain this I need to give you some knowledge on the keywords used.

- Uncontained – An user entity that crosses beyond application boundary. In other words you can access resources outside your application boundary
- Contained – An user entity that resides within an application boundary. In other words you can't access resources outside your application boundary

- Partial Contained database – It's a contained database which also allows you to access the objects outside
- Full Contained database – Full containment won't allow you to access the objects outside the application boundary.

As of now SQL Server Denali aka SQL Server 2012 supports only partial contained database. Database movement made easy with the help of contained database, since the users are created within the database, credentials are stored in the db and there is no need to map SID's as no dependency on server logins. It resolves another problem; if your database collation is different than the server collation then you will be facing problem as tempdb will fetch the collation from server level. In earlier version we will overcome this with COLLATE parameter however it's no longer required as here tempdb will be using the calling database collation.

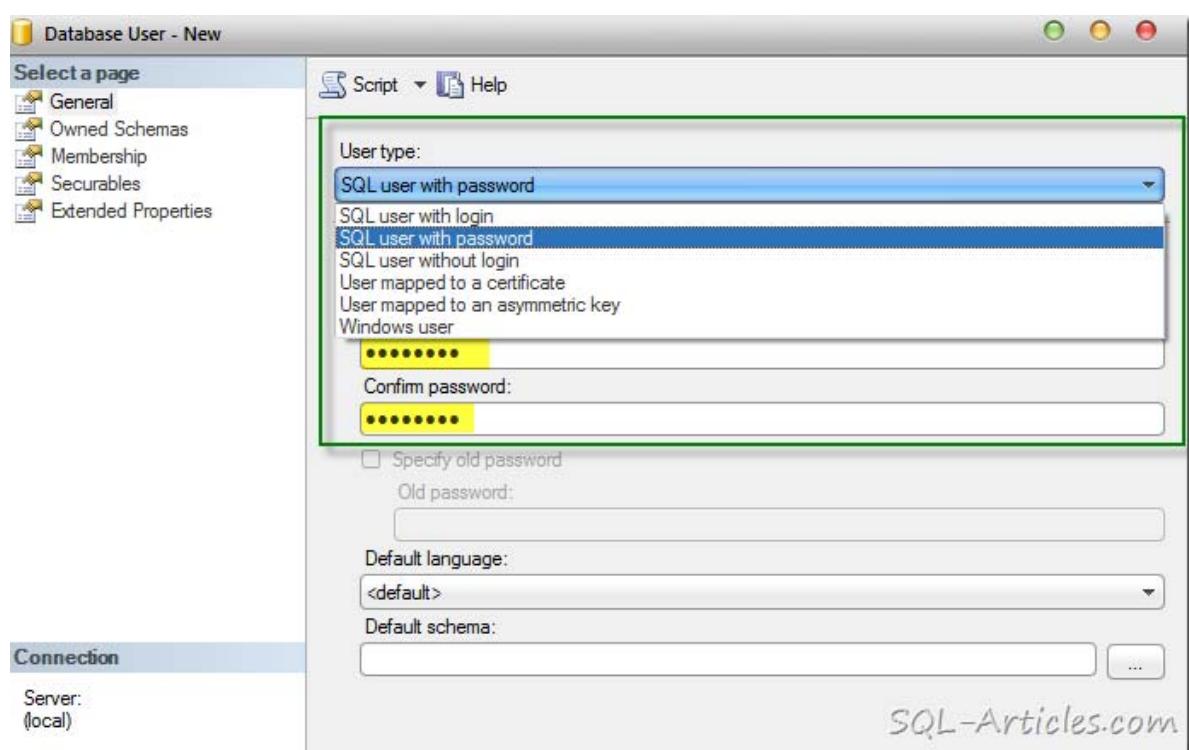
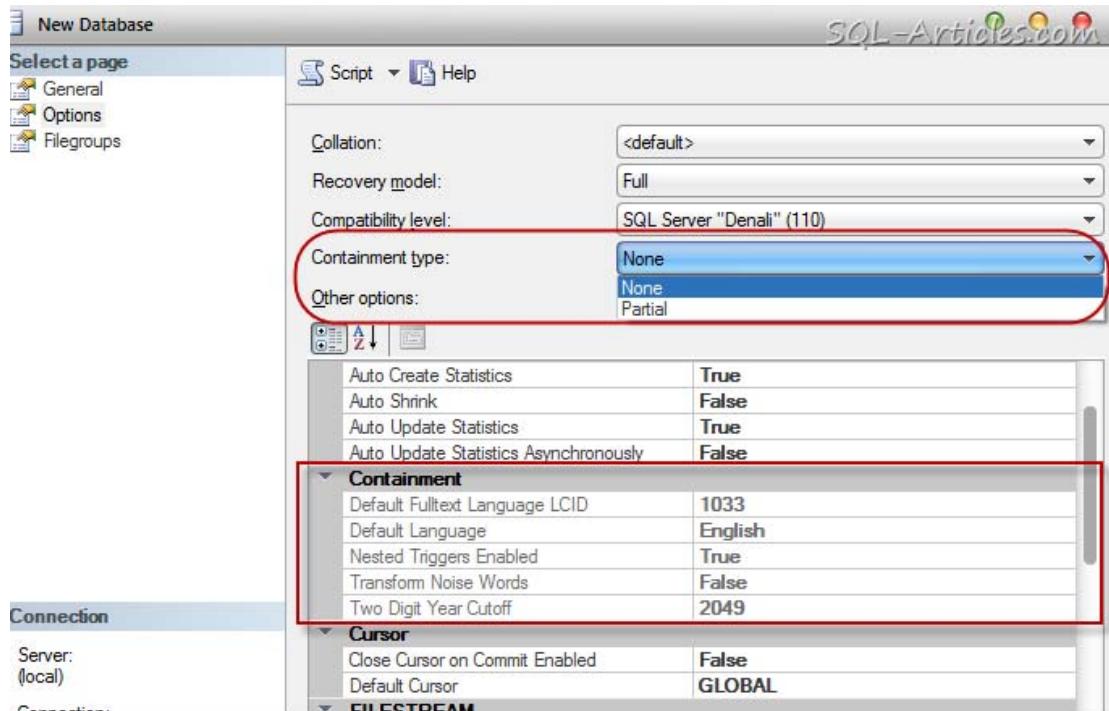
How to Create Contained Database

Contained database feature is not enabled by default . You need to enable this feature before you using it. Use the below code to enable this feature

```
EXEC SP_CONFIGURE 'contained database authentication',1  
RECONFIGURE  
GO
```

There is a new parameter in CREATE database command, it's called CONTAINMENT. As of now this parameter allows NONE and PARTIAL as values. You need to specify this parameter during database creation. In GUI if you go to options tab you need to specify Containment parameter.

Through GUI



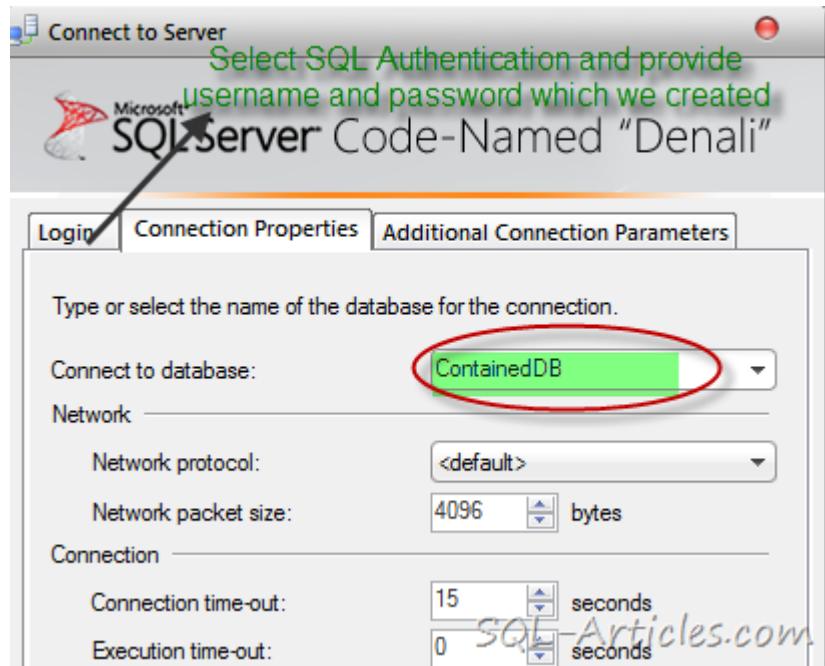
Through T-SQL

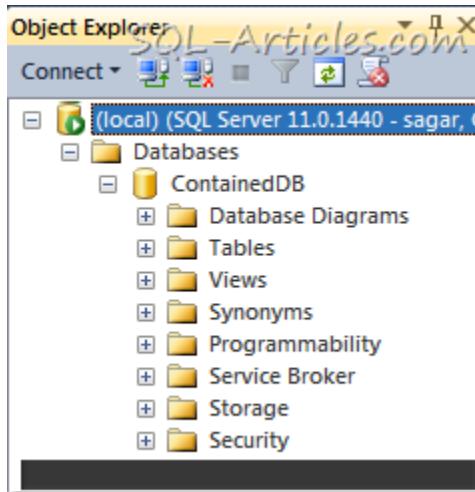
```
CREATE DATABASE [ContainedDB] CONTAINMENT = PARTIAL  
GO
```

Now lets connect to the server with contained database user. You need to specify the database name in the connection string or else you wont be able to connect to the server, you will be thrown login failed error message like below. The user won't have access outside the database entity, you will be able to do the work within database only.

Error: 18456, Severity: 14, State: 5.

Login failed for user 'sagar'. Reason: Could not find a login matching the name provided.





If you see the image above the user wont be able to do anything outside the contained database.

Will I be able to connect as contained SQL User if the server is configured as windows authentication?

No, you won't be able to connect as contained SQL User if the server is configured as windows authentication, you will get the below error Error: 18456, Severity: 14, State: 58.

Best Practices on Security

1. Ensure the physical security of each SQL Server, preventing any unauthorized users to physically accessing your servers.
2. Only install required network libraries and network protocols on your SQL Server instances.
3. Minimize the number of sysadmins allowed to access SQL Server.
4. As a DBA, log on with sysadmin privileges only when needed. Create separate accounts for DBAs to access SQL Server when sysadmin privileges are not needed.
5. Assign the SA account a very obscure password, and never use it to log onto SQL Server. Use a Windows Authentication account to access SQL Server as a sysadmin instead.
6. Give users the least amount of permissions they need to perform their job.
7. Use stored procedures or views to allow users to access data instead of letting them directly access tables.
8. When possible, use Windows Authentication logins instead of SQL Server logins.
9. Use strong passwords for all SQL Server login accounts.
10. Don't grant permissions to the public database role.

11. Remove user login IDs who no longer need access to SQL Server.
12. Remove the guest user account from each user database.
13. Disable cross database ownership chaining if not required.
14. Never grant permission to the xp_cmdshell to non-sysadmins.
15. Remove sample databases from all production SQL Server instances.
16. Use Windows Global Groups, or SQL Server Roles to manage groups of users that need similar permissions.
17. Avoid creating network shares on any SQL Server.
18. Turn on login auditing so you can see who has succeeded, and failed, to login.
19. Don't use the SA account, or login IDs who are members of the Sysadmin group, as accounts used to access SQL Server from applications.
20. Ensure that your SQL Servers are behind a firewall and are not exposed directly to the Internet.
21. Remove the BUILTIN/Administrators group to prevent local server administrators from being able to access SQL Server.
22. Run each separate SQL Server service under a different Windows domain account.
23. Only give SQL Server service accounts the minimum rights and permissions needed to run the service. In most cases, local administrator rights are not required, and domain administrator rights are never needed. SQL Server setup will automatically configure service accounts with the necessary permissions for them to run correctly, you don't have to do anything.
24. When using distributed queries, use linked servers instead of remote servers.
25. Do not browse the web from a SQL Server.
26. Instead of installing virus protection on a SQL Server, perform virus scans from a remote server during a part of the day when user activity is less.
27. Add operating system and SQL Server service packs and hot fixes soon after they are released and tested, as they often include security enhancements.
28. Encrypt all SQL Server backups with a third-party backup tool, such as SQL Backup Pro.
29. Only enable C2 auditing or Common Criteria compliance if required.
30. Consider running a SQL Server security scanner against your SQL servers to identify security holes.
31. Consider adding a certificate to your SQL Server instances and enable SSL or IPSEC for connections to clients.
32. If using SQL Server 2005, enable password policy checking.
33. If using SQL Server 2005, implement database encryption to protect confidential data.
34. If using SQL Server 2005, don't use the SQL Server Surface Area Configuration tool to unlock features you don't absolutely need.
35. If using SQL Server 2005 and you create endpoints, only grant CONNECT permissions to the logins that need access to them. Explicitly deny CONNECT permissions to endpoints that are not needed by users.

Problem/Case Study:

1. How to grant column level permissions on another schema?

Case study 2: How to Recover "SA" password when you forget it. - SQL 2005/2008

What to do if you forgot SA password in SQL Server 2005/2008?

In general if you forgot SA password or if the SA account is disabled these are the options to login into SQL Server 2005 and reset or enable SA.

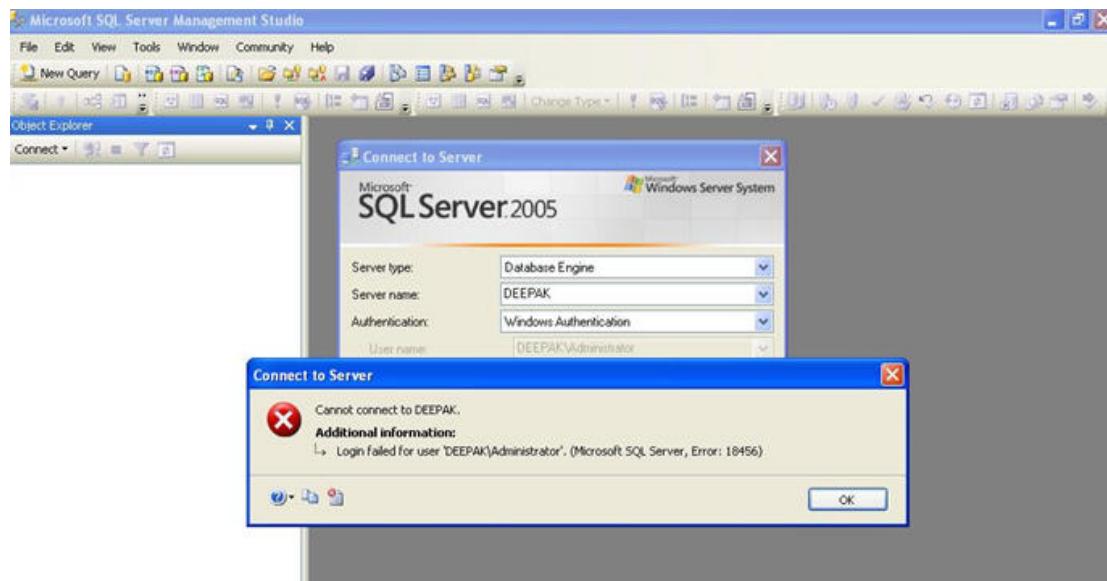
1. If Builtin\Administrator is present in SQL Server, you can login with an ID which is member of Administrators group and reset the SA password.
2. Or else if you have some other ID which is having sysadmin privilege in SQL level(this also SQLservice account), you can login with that and reset SA.

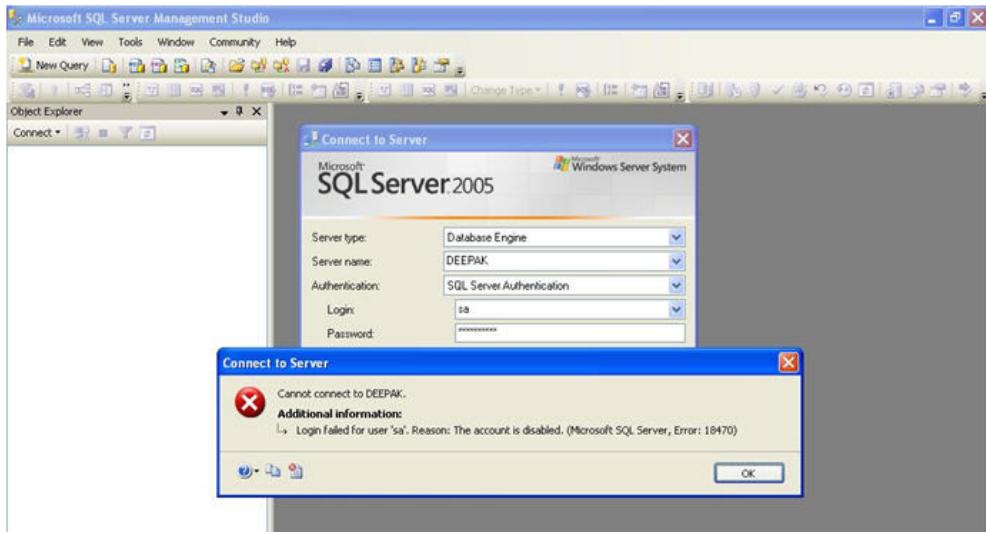
But in case if you have the following scenario where,

1. You have disabled SA (or) forgotten SA password
2. You followed the best security practice and removed Builtin\administrator from SQL Server

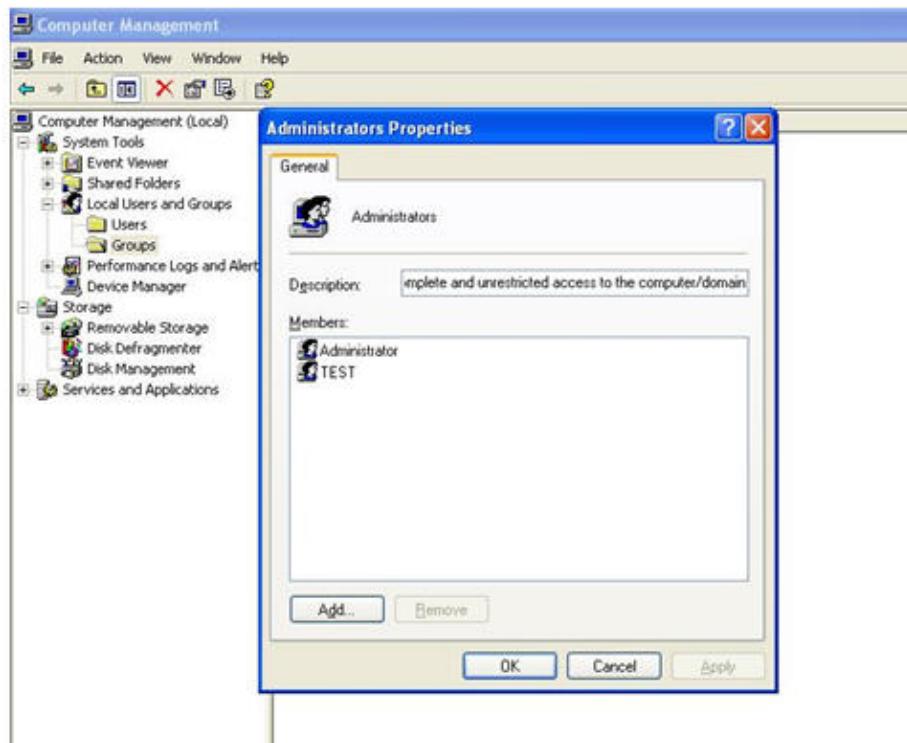
So you cannot login with a sysadmin ID into SQL Server, and you start thinking about uninstalling SQL Server 2005. No need to perform any uninstall and reinstall in such scenarios in SQL Server 2005 as you have this option where the Members of Windows Administrative groups have sysadmin privilege in SQL Server if you start SQL Server 2005 in Single user mode.

Consider this example, where I have disabled SA and removed Builtin\Administrator from SQL Server 2005. Refer the below screenshots which displays the error message I get when I login Administrator and SA,





Refer the below screenshot which shows the members of windows administrator group,

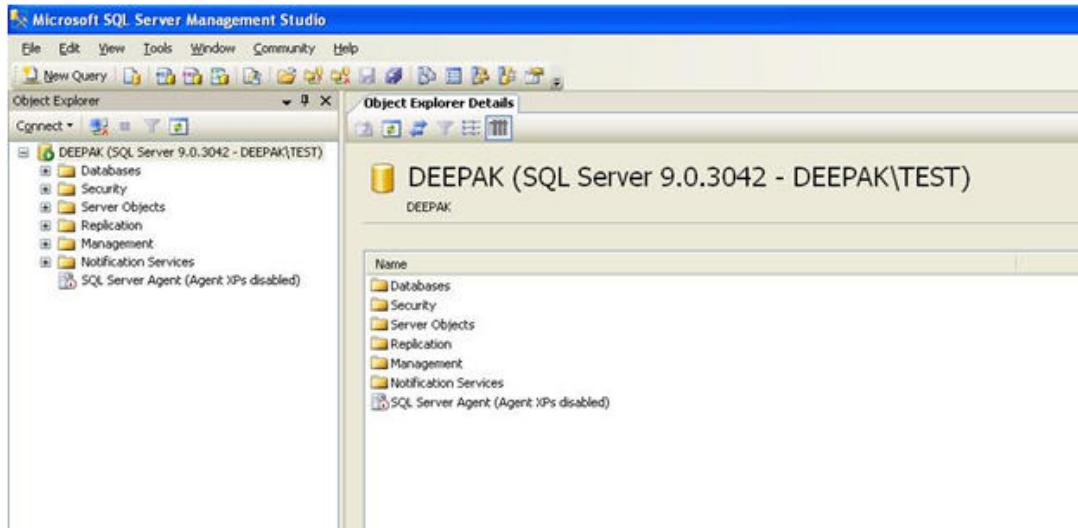


I perform the following steps,

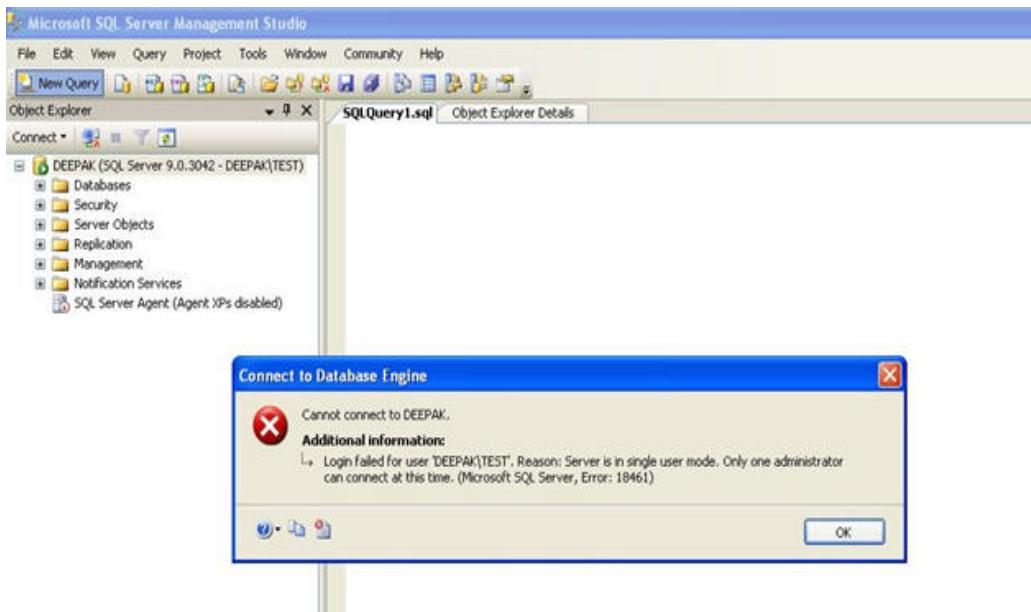
1. Create account "test" at OS level with admin privileges. Login with the ID Test @OS Level
2. Stop SQL Server 2005 using the command, **NET STOP MSSQLSERVER**

3. Start SQL Server 2005 in Single-User mode using the command, **NET START MSSQLSERVER /m**

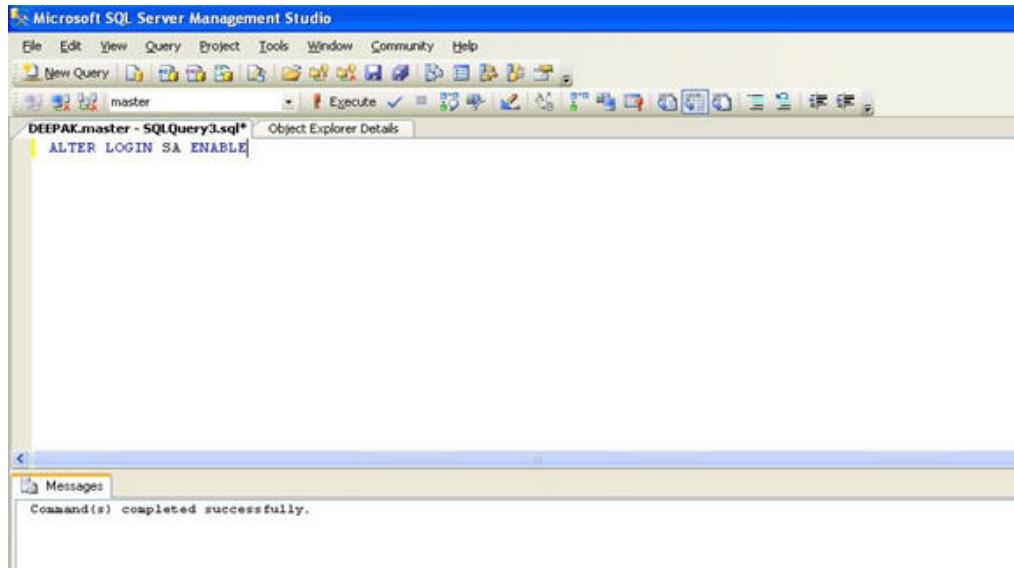
4. Login into SQL Server 2005 using the ID Test as shown in the below screenshot,



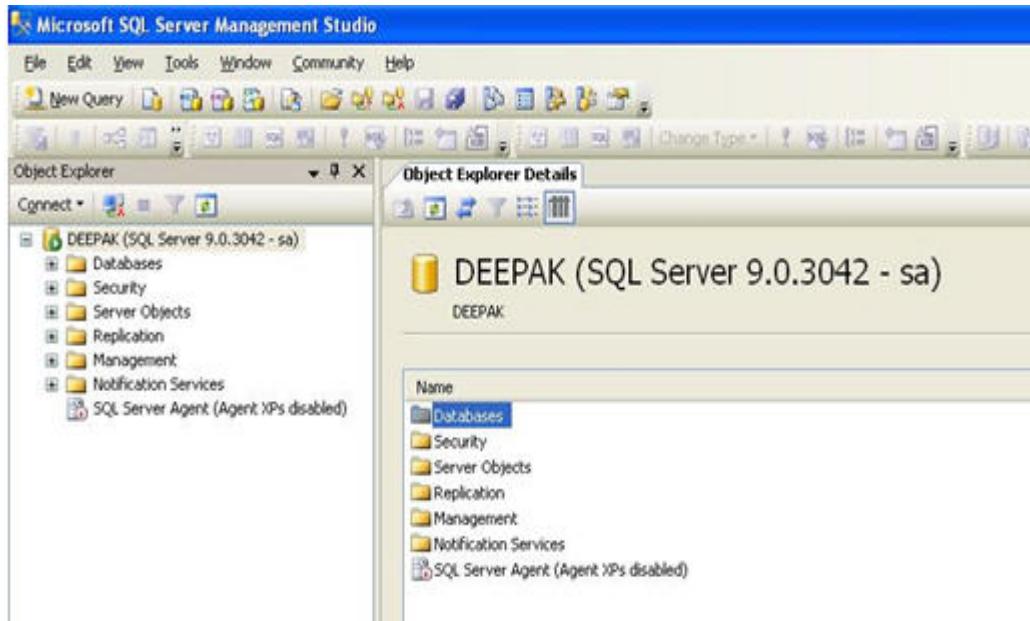
5. Since SQL Server is started in Single-User mode it will allow only one connection and hence you will get the following error if you click the "New Query"



6. Disconnect and close the Object Explorer and then connect using "New Query" you will be able to connect as shown below, and then enable SA login using the command ALTER Login SA enable



8. Now you need to Stop SQL Server and start it normally using the command, **NET START MSSQLSERVER** and connect using SA or the new login you created and proceed as shown below,



Case study 3: How to grant single stored procedure execute permissions to user

```
exec sp_addrole 'procrun'  
go  
exec sp_addrolemember 'procname','user'  
go  
grant execute on procname to procrun
```

Case study 4: How to grant all stored procedures execute permissions to an account.

Solution:

```
/* Create a new role for executing stored procedures */  
CREATE ROLE db_executor  
  
/* Grant stored procedure execute rights to the role */  
GRANT EXECUTE TO db_executor  
  
/* Add a user to the db_executor role */  
EXEC sp_addrolemember 'db_executor', 'AccountName'
```

Case study 5: Recover access to a SQL Server instance

Starting with SQL Server 2008, the local Administrators group is no longer added by default during SQL Server setup; you even have to use a manual step to add the current user as a local administrator. This means that it is possible, especially if you don't use mixed authentication (or have forgotten the sa password), that you can be completely locked out of your own SQL Server instance. I've seen cases where an employee has moved on, but their Windows account, being the only one with Administrator privileges for SQL Server, had been completely obliterated from the system. Of course that person was the only one who knew the sa password as well, and being a local admin or even a domain admin might not help you.

The typical workaround I have seen employed is to restart SQL Server in single user mode. However, this approach requires at least some downtime; in some systems, this would be unacceptable. And depending on what needs to be managed on the server, it might not be feasible to wait for a scheduled maintenance window.

There is a very painless way to solve this problem without any downtime: [PsExec](#). While it wasn't one of its primary design goals, PsExec allows you to run programs as the NT AUTHORITY\SYSTEM account, which - unlike "regular" Administrator accounts - has inherent access to SQL Server.

You can download PsExec from the below link:

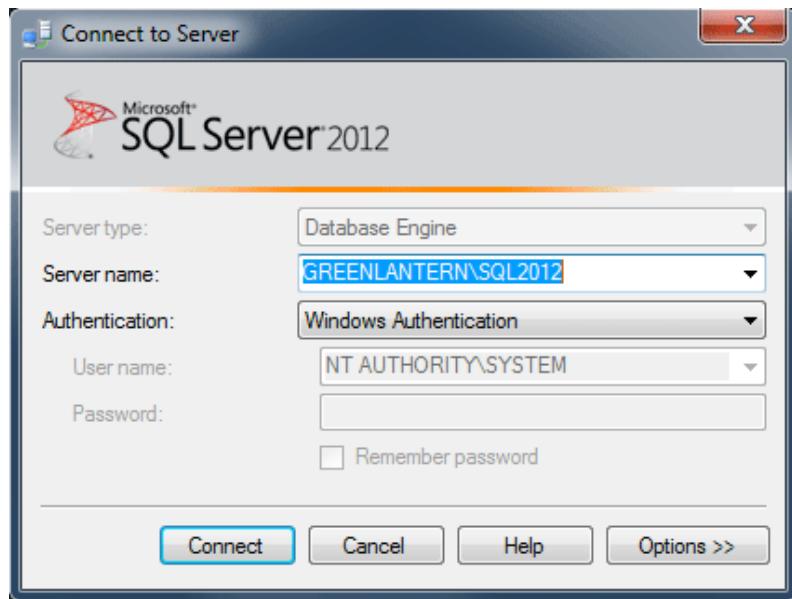
<http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>

The process to get back up and running is quite simple. Log in to the server where SQL Server is running, as an account with local Administrator privileges. Download and extract PsExec.exe. Start an elevated command prompt (Shift + Right-click, "Run as Administrator"). Run the following command, adjusting for your actual path to Management Studio, which may be different:

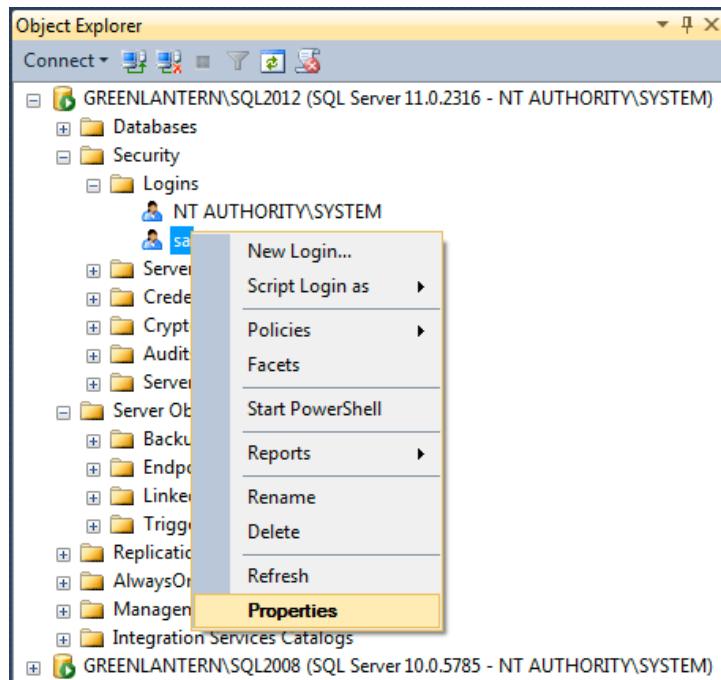
```
PsExec -s -i "C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\Ssms.exe"
```

This command tells PsExec to run SSMS interactively (-i) and as the system account (-s).

You will get an error message if you are not an Administrator. You will need to agree to the license terms in order to proceed. When Management Studio launches, it will prompt you to connect to a server. You will notice that the authentication mode is Windows Authentication, and the username is hard-coded as NT AUTHORITY\SYSTEM:



Once you connect, in Object Explorer, you will see that you are connected to the server as NT AUTHORITY\SYSTEM:



Now, you can go in to Security > Logins and add your account as a sysadmin, add other admin accounts, update the sa password, and do anything else you need to do to make your instance manageable. As you can see, I was able to connect in this way to both SQL Server 2008 and SQL Server 2012 instances from an instance of Management Studio 2012. I also confirmed that this process works when connecting to a SQL Server 2008 instance using the 2008 version of SSMS. In both cases, I was logged in to Windows as a local administrator, but the account had no explicit access to either SQL Server instance.

You can connect to additional instances on the local server using the Connect > Database Engine dropdown in Object Explorer, or by right-clicking a query window and choosing Connection > Change Connection.

- You must be a local Administrator to masquerade as NTAUTHORITY\SYSTEM.
- You may need to disable UAC.
- Leave the command prompt running in the background until you're done with all of your changes - if you inadvertently Ctrl+C from within the command prompt, SSMS will vanish.

Chapter – 12

Automating Administrative Tasks

SQL Server Agent

This section examines how to automate tasks on the SQL Server using the Microsoft SQL Server Agent Service. The SQL Agent service runs as Windows service that is dependent on the SQL Service. Each instance of SQL will have its own Agent service to manage jobs, schedules, operators, and alerts. You learn about the essential components of the Agent service for single and multiple server management configurations.

The primary purpose of the SQL Server Agent is to make your job easier. In a perfect world, you could configure your servers, let them run, and never worry about losing data or the database going offline. But, as is too often the case, this isn't a perfect world. And because you can't realistically monitor every server every minute of every day, you can use the SQL Server Agent to leverage against what you can't do.

The SQL Server Agent service is not available in SQL Server 2005 Express Edition.

You did not configure the Agent to start automatically; you'll need to know how to start it manually. There are actually four different ways you can start and stop the SQL Server Agent service. One way is to use the

NET START command from a command prompt:

```
NET START SQLSERVERAGENT
```

To stop the service, use the NET STOP command:

```
NET STOP SQLSERVERAGENT
```

Agent Security

When planning to use the SQL Server Agent service, or allowing other users to access it, you need to ensure that appropriate access is granted. By default, only members of the sysadmin fixed server role have complete access to the Agent service. In the msdb database, additional roles are created with varying levels of rights and permissions, but these roles are empty until a user is explicitly added to these roles. In this section, you learn about each of these roles and the permissions assigned to them.

SQLAgentUserRole

The SQLAgentUserRole is the most limited of the three Agent roles. Users who are members of this role have the ability to create new jobs and schedules, and can manage only those jobs and schedules they create.

SQLAgentReaderRole

As with SQLAgentUserRole, SQLAgentReaderRole can enable users to create local jobs and schedules, and manage only those that they create. In addition to these permissions, they can also view the properties of other local jobs, as well as multi-server jobs.

SQLAgentOperatorRole

Members of this role can create local jobs, as well as manage and modify jobs they own. They can also view and delete the job history information for all local jobs. To a limited extent, they can also enable or disable jobs and schedules owned by other users.

There are four basic components of SQL Server Agent, each of which the following sections discuss:

Jobs: Defines the work to be done

Schedules: Defines when the job will be executed.

Operators: Lists the people who can be notified for job status and alerts

Alerts: Enables you to set up an automatic response or notification when an event occurs

Jobs

A great reason to use SQL Server Agent is to create tasks you can schedule to complete work automatically, such as backing up a database. A SQL Server Agent job contains the definition of the work to be done. The job itself doesn't do the work but is a container for the job steps, which is where the work is done. A job has a name, a description, an owner, a category, and a job can be enabled or disabled. Jobs can be run in several ways:

- By attaching the job to one or more schedules
- In response to one or more alerts
- By executing sp_start_job
- Manually via SQL Server Management Studio

Job Steps

A job consists of one or more job steps. The job steps are where the work is actually done. Each job step has a name and a type. Be sure to give your jobs and job steps good descriptive names that can be useful when they appear in error and logging messages. You can create a number of different types of job steps.

There is some control of flow related to job steps as well. You may specify an action for when the step succeeds and when the step fails. These actions can be one of the following:

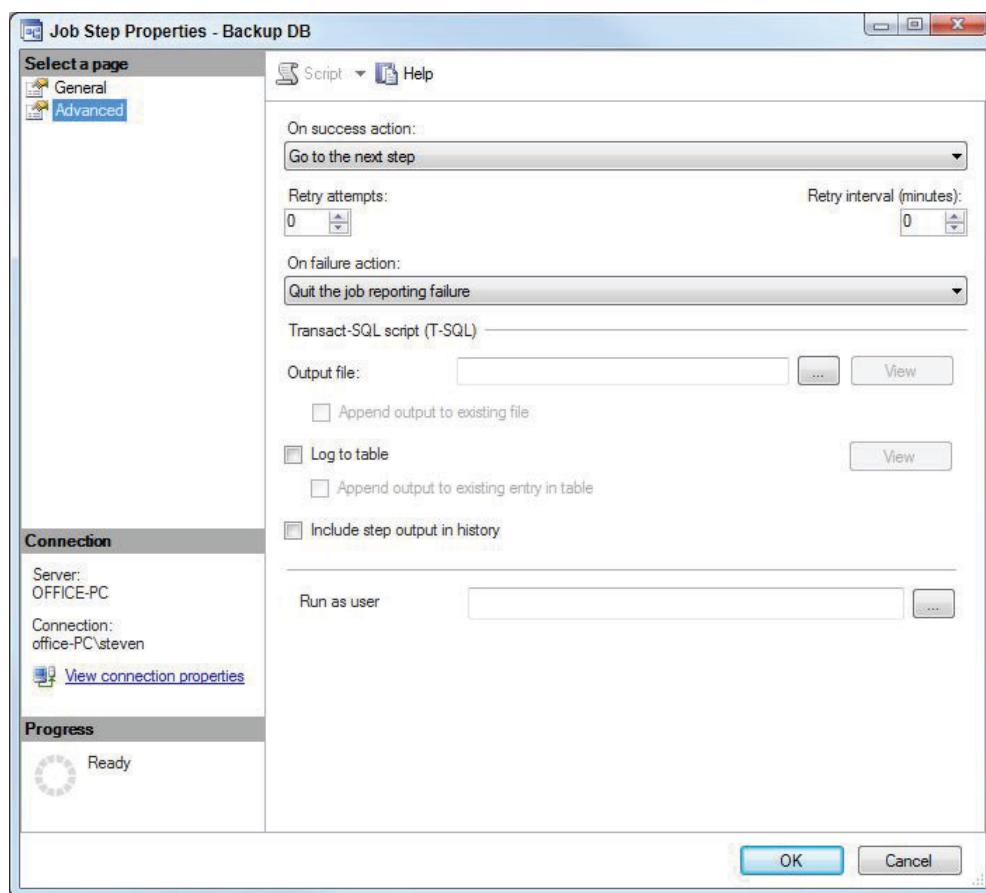
- Quit the job, indicating success.
- Quit the job, with failure.
- Go to another job step.

You may also require that the job step be retried before the job step fails. You may specify the number of retry attempts and the retry interval, in minutes. Once you set these guidelines, a job step will then be retried the number of times you specify in the Retry Attempts field before it executes the On Failure control of flow. If the Retry Interval in Minutes field has been set, the

step waits for the specified time period before retrying. This can be useful when there are dependencies between jobs. For example, you may have a job step that does a bulk insert from a text file. The text file is placed into the proper directory by some other process, which may run late. You could create a VBScript job step that checks for the presence of the input file. To test for the file every 10 minutes for 30 minutes, you would set the retry attempts to 3 and the Retry Interval to 10.

Job Step Logging

Each time a job is run, job history is created. Job history tells you when the job started, when it completed, and if it was successful. Each job step may be configured for logging and history as well. All the logging setup for a job step is on the Advanced Tab of the job step properties. The Advanced Tab of the Job Step Properties is shown in Figure and the key options that affect logging are discussed in the following list.



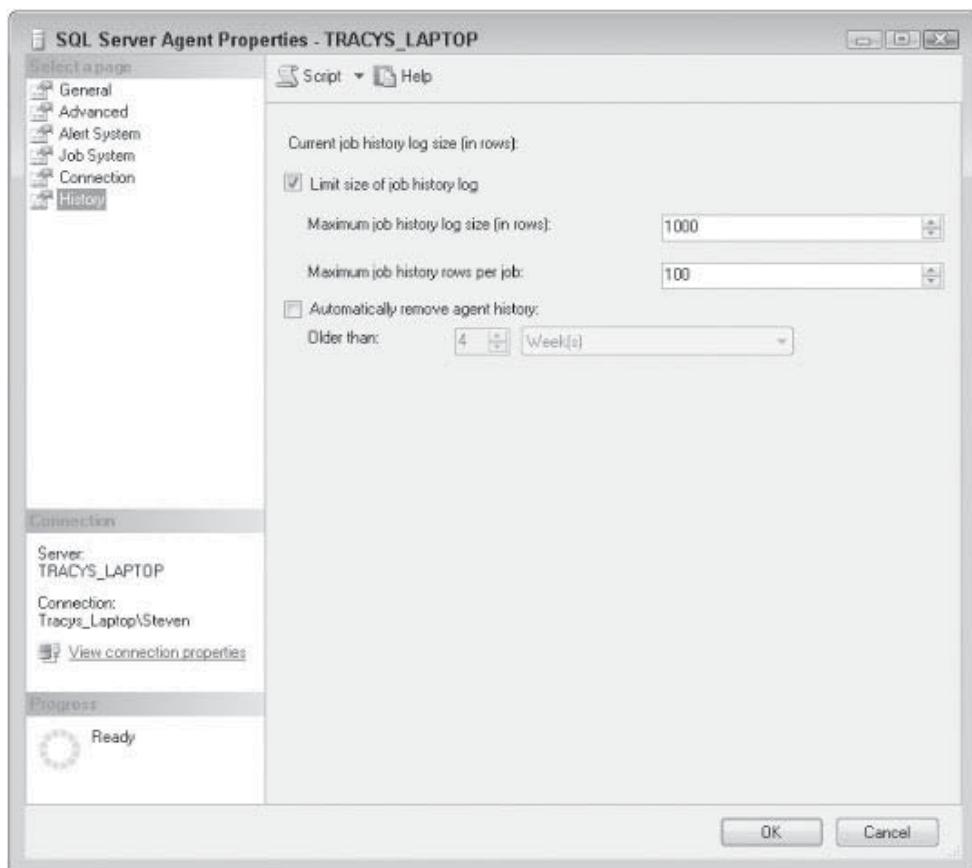
Output to File: Job steps executed by sysadmin role members may also have the job step history written to a file. To do so, enter the filename in the Output File text box. Check the

Append Output to Existing File check box if you do not want to overwrite the file. Job steps executed by others can log to only dbo.sysjobstepslogs in msdb.

Log to table: You may also choose to have the information logged to dbo.sysjobstepslogs in msdb. To log to this table, check the Log to Table check box. To include step history from multiple job runs, also check the Append Output to Existing Entry in Table. Otherwise, you see only the most recent history.

Include step output in history: To append the job step history to the job history, check the Include the step output in history check box.

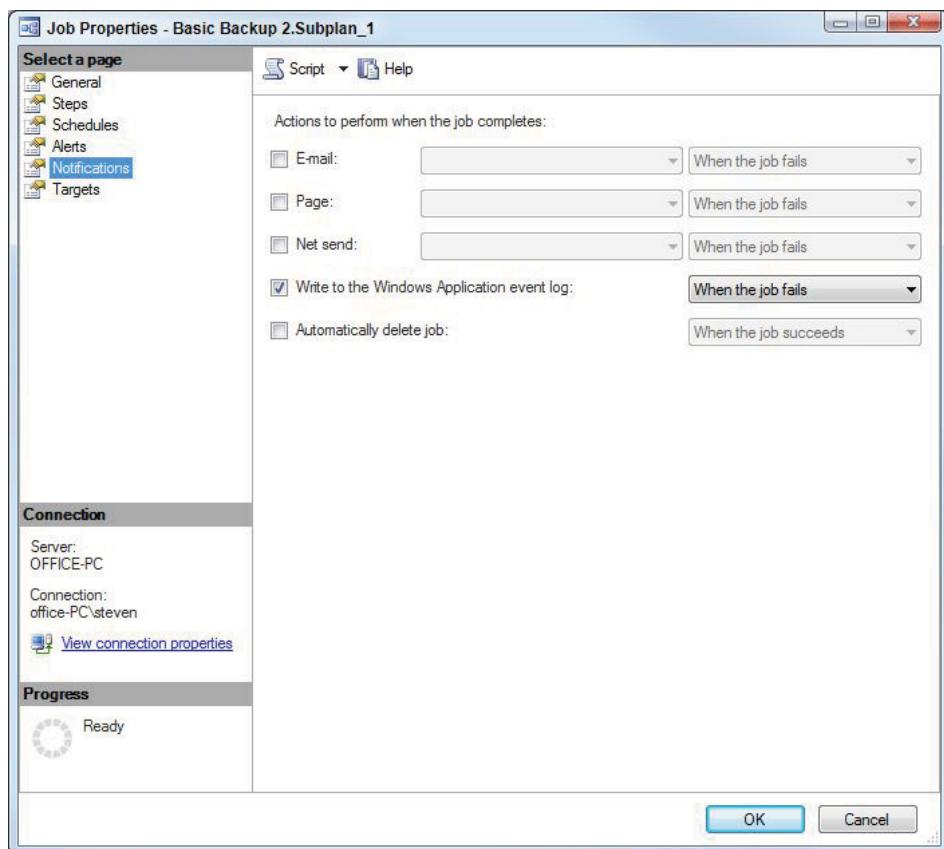
By default, SQL Server stores only 1,000 rows in its Job History Log and a maximum of 100 for any one job. The Job History Log is a rolling log, so the oldest records are deleted to make room for newer job history. If you have a lot of jobs, or jobs that run frequently, the Job History Log can soon become full and start deleting old records. If you need to change the size of the log, you can do so under the SQL Server Agent properties,



Job Notifications:

You can configure SQL Server Agent to notify you when a job completes, succeeds, or fails. To do so, follow these steps:

- 1.** In the Job Properties dialog, choose Notifications to see the dialog box shown in Figure
- 2.** A job can send a notification via e-mail, pager, and Net Send. A job can also write to the Windows Application Event Log. As Figure shows, there is a line in the dialog box for each of these delivery methods. Place a check beside the delivery method you want; you may choose multiple methods.



- 3.** Click the drop-down menu for each option and choose an operator to notify. An operator enables you to define the e-mail address for the delivery. (Operator setup is described later in this chapter.)
- 4.** Choose the event that should trigger the notification. It can be when the job completes, when the job fails, or when the job succeeds. You may not want to be notified at all for some jobs such as routine maintenance like Index Maintenance. However, for mission-critical jobs, you might want to be e-mailed always when the job completes, and perhaps paged and notified

through Net Send if the job fails, so you can know immediately. Examples of more critical jobs where you do want notification might be Backups and DBCC CHECKDB.

Schedules

One of the advantages of SQL Server Agent is that you can schedule your jobs. You can schedule a job to run at any of these times:

- When SQL Server Agent starts
- Once, at a specified date and time
- On a recurring basis
- When the CPU utilization of your server is idle

Operators

An operator is a SQL Server Agent object that contains a friendly name and some contact information. Operators can be notified on completion of SQL Server Agent jobs and when alerts occur. (Alerts are covered in the next section.) You may want to notify operators who can fix problems related to jobs and alerts, so they may go about their business to support the business. You may also want to automatically notify management when mission-critical events occur, such as failure of the payroll cycle.

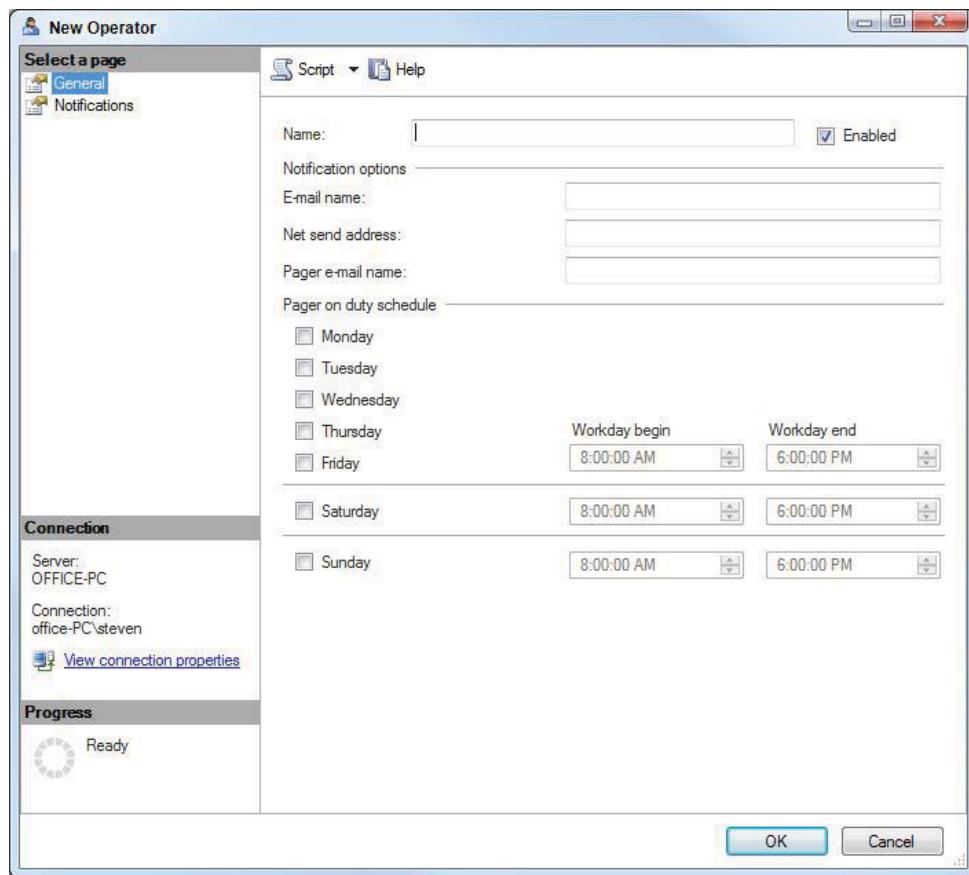
You should define operators before you begin defining alerts. This enables you to choose the operators you want to notify as you are defining the alert, saving you some time. To create a new operator, follow these steps:

1. Expand the SQL Server Agent Node in the Object Explorer in SQL Server Management Studio.
2. From there, right-click Operators and select New Operator. The New Operator dialog shown in Figure appears, and here you can create a new operator. The operator name must be unique and fewer than 128 characters.

Operator Notifications

Jobs enable you to notify a single operator for three different send types:

E-mail: To use e-mail or pager notifications, Database Mail must be set up and enabled, and SQL Server Agent must be configured. For e-mail notifications, you can provide an e-mail address. You may provide multiple e-mail addresses separated by semicolons. This could also be an e-mail group defined within your e-mail system. If you want to notify many people, it is better to define an e-mail group in your e-mail system. This enables you to change the list of people notified without having to change every job.



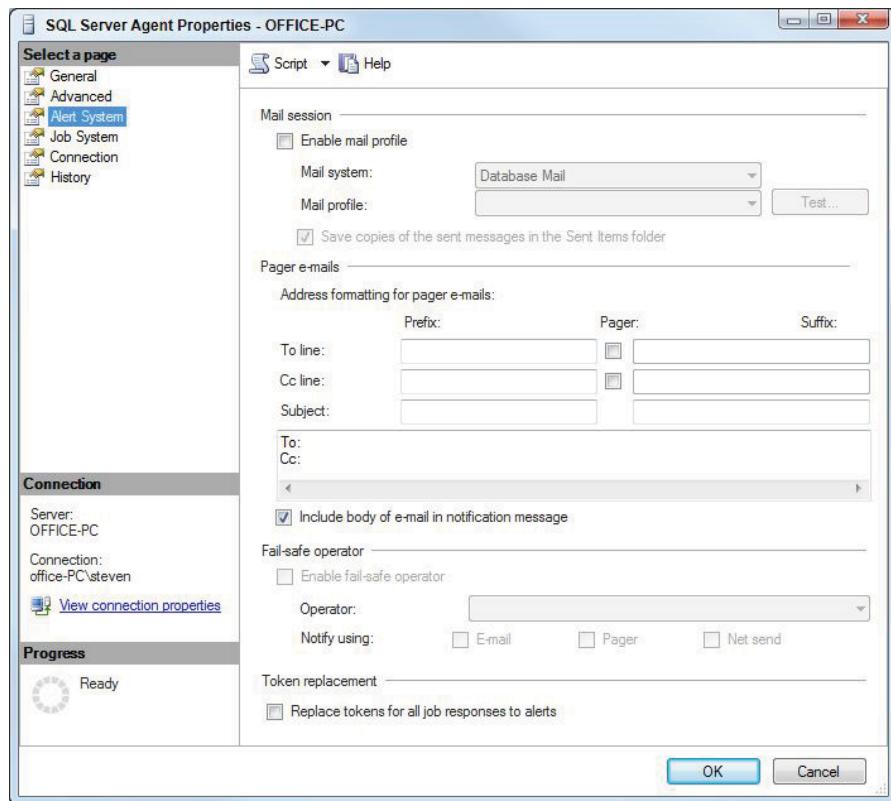
Failsafe Operator

What happens if an alert occurs and no operators are on duty, according to their pager on-duty schedule? Unless you specify a failsafe operator, no one would be notified. The failsafe operator is a security measure that enables an alert notification (not job notification) to be delivered for pager notifications (not e-mail or Net Send) that could not be sent. Failures to send pager notifications include the following:

- None of the specified operators are on duty.
- SQL Server Agent cannot access the appropriate tables in msdb.

To designate an operator as the Failsafe Operator, perform the following steps:

- Select the properties of SQL Server Agent.
- Select the Alert system tab as shown in Figure
- In the Fail-safe operator section select enable fail-safe operator.



Alerts

An *alert* is an automated response to an event. An *event* can be any of the following:

- SQL Server event
- SQL Server performance condition
- Windows Management Instrumentation (WMI) event

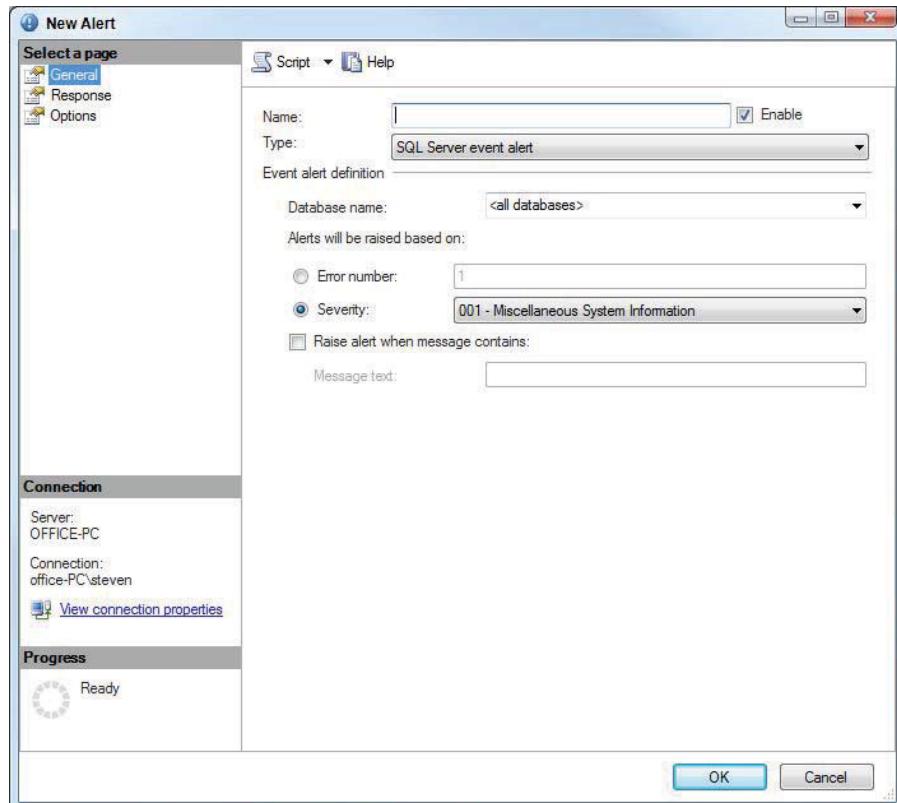
An alert can be created as a response to any of the events of these types. The following responses can be triggered as the result of an event alert:

- Start a SQL Server Agent job
- Notify one or more operators

To create an alert, follow these steps:

1. Open the New Alert dialog (see Figure) by selecting New Alert from the context menu on the Alerts Node under the SQL Server Agent node in SQL Server Management Studio.

- When you create an alert, you give it a name. Ensure that this name tells you something about what is going on; it will be included in all messages. Names such as Log Full Alert or Severity 18 Alert on Production might be useful.



- Then choose the event type on which the alert is based (refer to Figure). SQL Server events and SQL Server performance condition events are covered in this section; WMI events are outside the scope of this book and not addressed here.

Maintenance Plans

Maintenance plans can be created using the Maintenance Plan Wizard or using the design surface. The Wizard is useful if the DBA wants to create a basic maintenance plan. If he intends to create enhanced work flow then, it is advisable to use the design surface. Maintenance Plans are displayed only to users connected using the Windows Authentication.

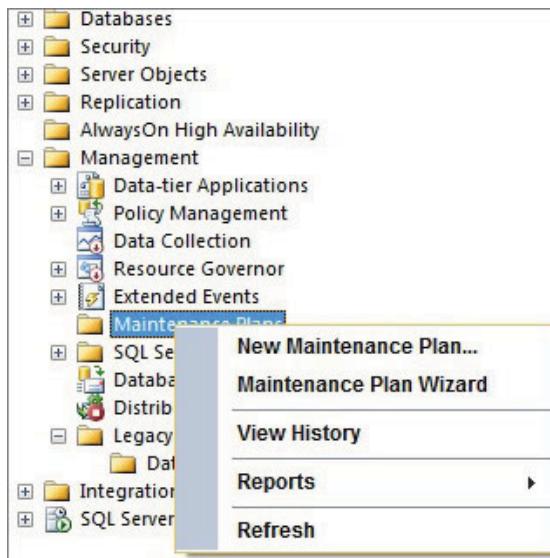
The Maintenance Plan Wizard helps set up the core maintenance tasks for optimum performance of the SQL Server components.

Maintenance Plan Wizard

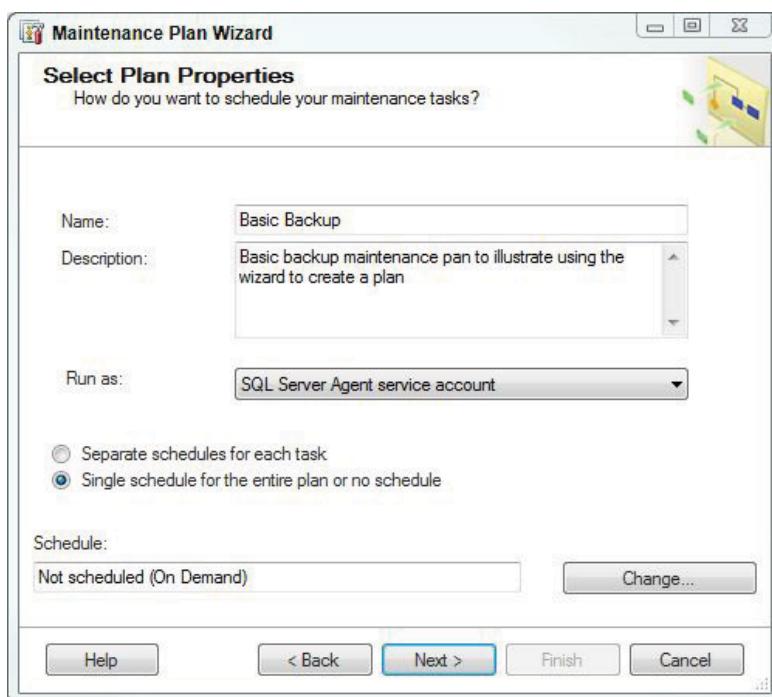
This section walks you through the steps to create a backup using the Maintenance Plan Wizard:

- First, launch the wizard, which lives on the context menu on the Maintenance Plans node in the Object Explorer in SQL Server Management Studio. Select the Maintenance Plans Wizard

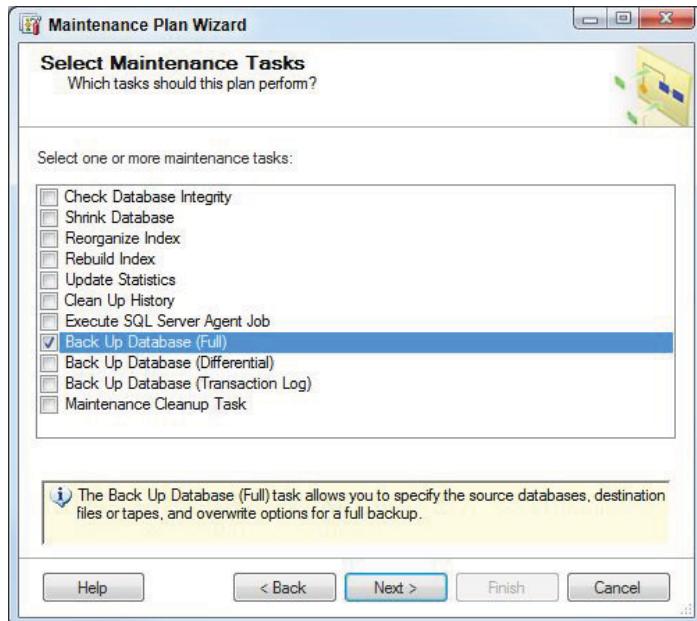
menu item to launch the first page of the wizard. Figure shows the menu selection to start the Maintenance Plan Wizard. You can opt to not show this page again and then select next. This brings up the Select Plan Properties page, where you can set some of the Plan options.



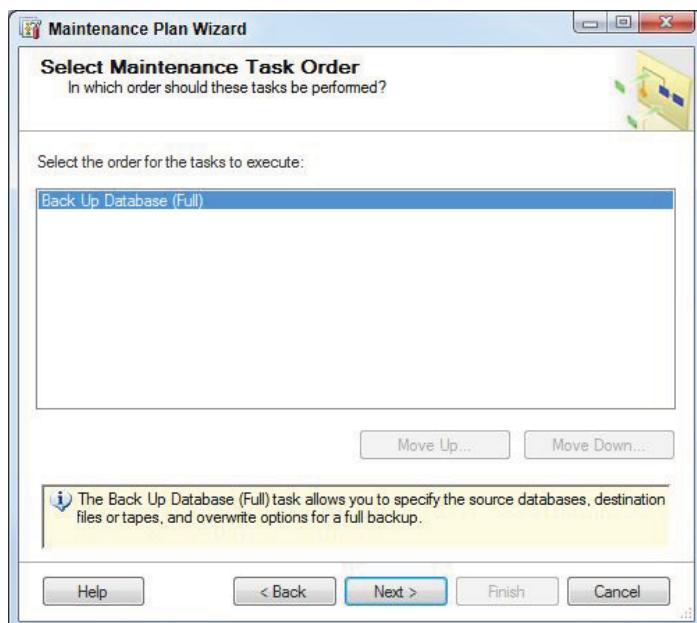
2. On this page, as shown in Figure, specify a name and description for the plan and select the scheduling options.



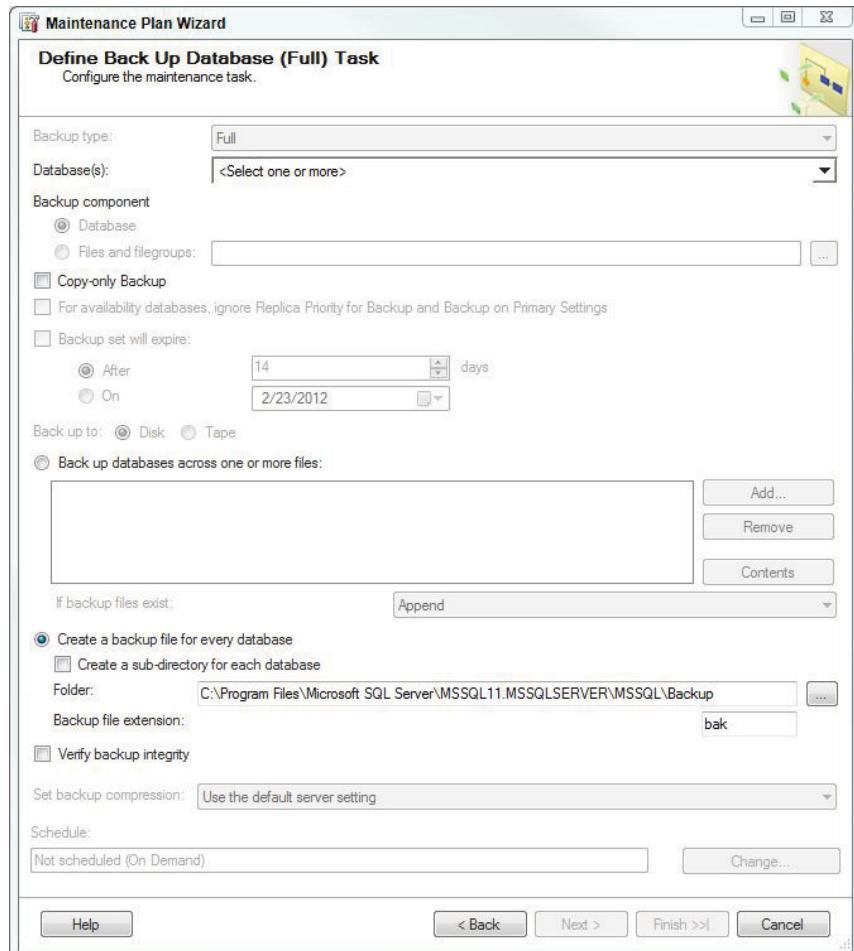
- 3.** Select Next to move to the Select Maintenance Tasks screen where you can choose the tasks you want the plan to perform. For this example select the Back Up Database (Full) option, as shown in Figure.



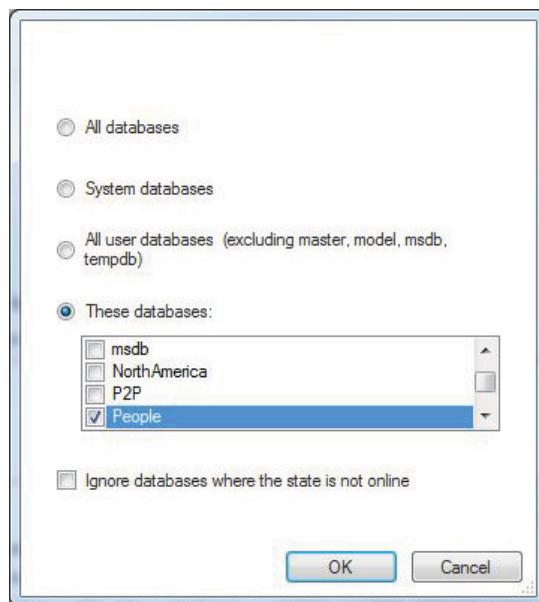
- 4.** Select Next to move to the Select Maintenance Task Order screen, as shown in Figure. If you selected multiple tasks on the previous page, you can reorder them here to run in the order you want. In this example you have only a single task, so click Next.



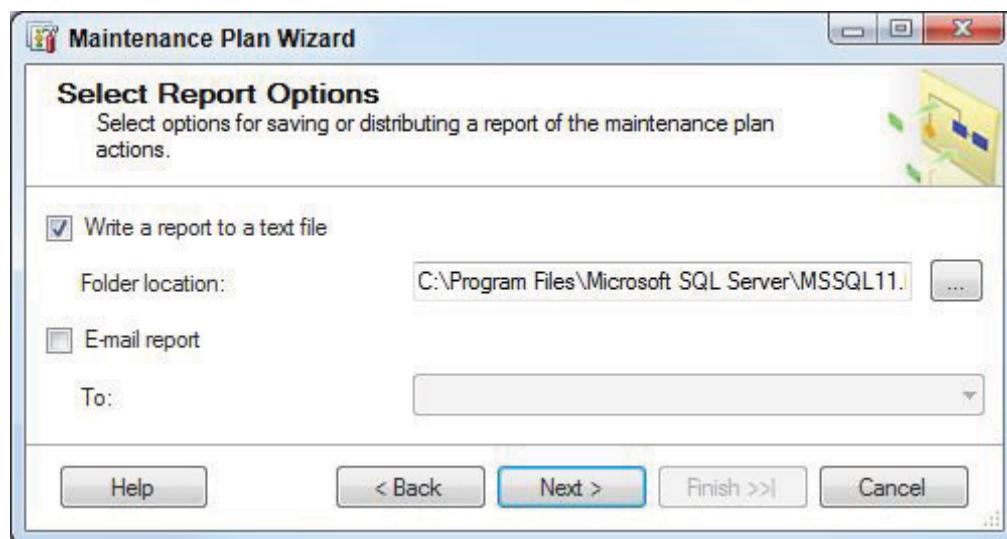
5. The next page is the Define Back Up Database (Full) Task screen, as shown in Figure. On this page select the details for the backup task. If you selected a different task on the Select Maintenance Tasks screen, you need to supply the details for that task. In the case of multiple tasks, this step presents a separate page for each task you selected in your plan.



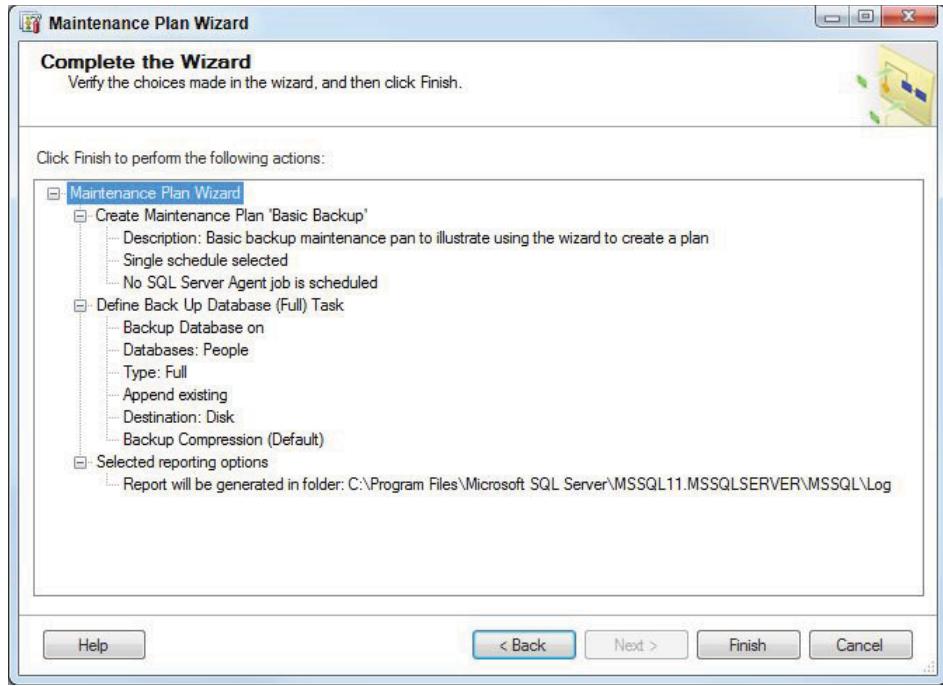
6. Figure shows the dialog where you can select the databases you want to back up. This figure shows just one database to back up.



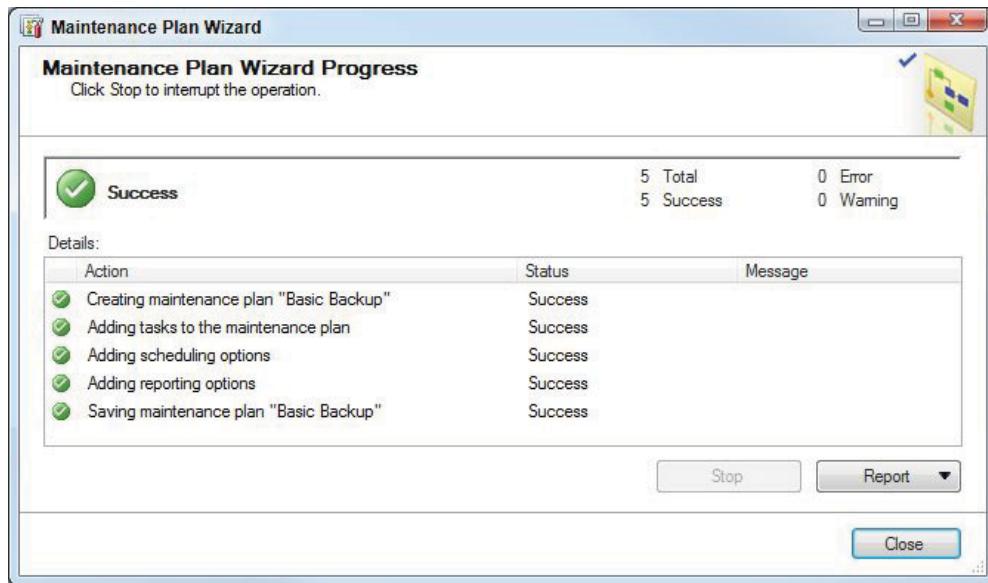
7. On the next page (shown in Figure), select the reporting options for the plan: write a log to a specific location, send an e-mail, or both.



8. Select Next to go to the final page of the wizard, where you can confirm your selections (see Figure).



9. Click Finish to create your plan. While the plan is being created, a status page will show you the progress on each step of the plan's creation, as shown in Figure.



The new plan now displays in the Object Explorer under the Maintenance Plans node and can be manually run by using the menu from that node.

You should have noticed along the way that the Maintenance Plan Wizard can perform only a limited number of tasks, but these are some of the most important routine maintenance activities on the server. Using this wizard enables you to automate many of the essential tasks needed on a SQL Server.

To explore more details about the plan you just created, look at the job created for this plan in the SQL Server Agent node, under the Jobs node. The job will be named *<Your plan name>.subplan_1*, so in this example the job name is Basic Backup.Subplan_1.

Best Practices on Job Maintenance

1. Avoid overlapping jobs on the same SQL Server instance. Ideally, each job should run separately at different times.
2. When creating jobs, be sure to include error trapping, log job activity, and set up alerts so you know instantly when a job fails.
3. Create a special SQL Server login account whose sole purpose is to run jobs, and assign it to all jobs.
4. If your jobs include Transact-SQL code, ensure that it is optimized to run efficiently.
5. Periodically (daily, weekly, or monthly) perform a database reorganization on all the indexes on all the tables in all your database. This will rebuild the indexes so that the data is no longer logically fragmented. Fragmented data can cause SQL Server to perform unnecessary data reads, slowing down SQL Server's performance. Reindexing tables will also update column statistics.
6. Don't reindex your tables when your database is in active production, as it can lock resources and cause your users performance problems. Reindexing should be scheduled during down times, or during light use of the databases.
7. At least every two weeks, run DBCC CHECKDB on all your databases to verify database integrity.
8. Avoid running most DBCC commands during busy times of the day. These commands are often I/O intensive and can reduce performance of the SQL Server, negatively affecting users.
9. If you rarely restart the mssqlserver service, you may find that the current SQL Server log gets very large and takes a long time to load and view. You can truncate (essentially create a new log) the current server log by running DBCC ERRORLOG. Set this up as a weekly job.
10. Script all jobs and store these scripts in a secure area so they can be used if you need to rebuild the servers.

Chapter – 13

Monitoring SQL Server

One of the primary responsibilities of the database administrator is the ongoing monitoring of SQL Server performance. Much of this monitoring can be automated, but for the most part, the monitoring results must be interpreted and acted upon in a systematic approach by the DBA. The monitoring job never ends, and it can become quite complex. Knowing what to monitor, when to monitor, and what constitutes acceptable and unacceptable behavior can become a full-time job. Making things worse is the fact that each SQL Server installation is different, making a global recommendation about what indicators identify unacceptable and acceptable performance very difficult.

In this we will learn various tools used to monitor SQL Server and provides guidelines on how to use these tools to identify areas for optimization. Monitoring SQL Server can be a challenging process. SQL Server interacts heavily with every operating system subsystem. Some applications rely heavily on RAM, whereas others are CPU- or disk-intensive. SQL Server can be all three at the same time.

Tools and Techniques for Monitoring

Management Studio

DBAs spend a lot of time in SQL Server Management Studio. This tool enables you to perform most of your management tasks and to run queries. SQL Server 2012 uses a version of Visual Studio 2010 as its shell. Because this is a professional-level book, it won't go into every aspect of Management Studio, but instead covers some of the more common and advanced features that you might like to use for administration.

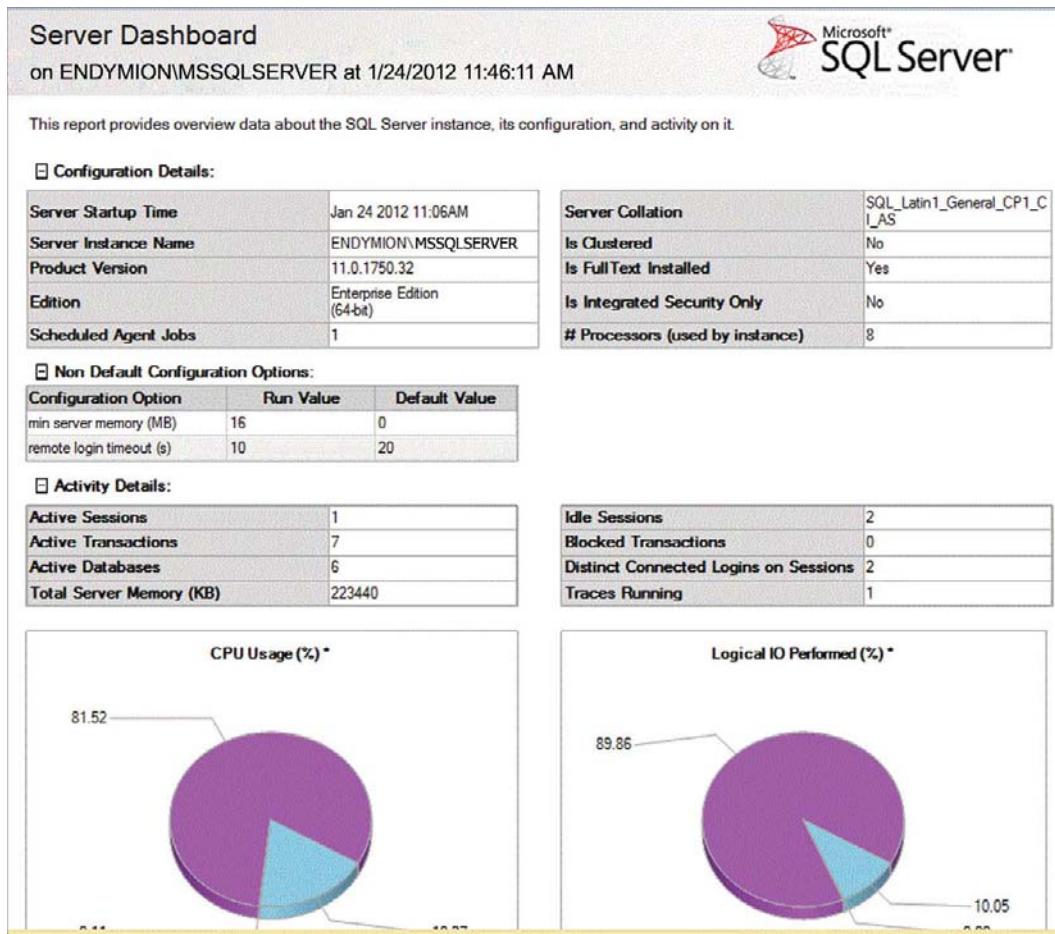
Reports

One of the most impressive features of the SQL Server management environment is the integrated reports available in each area of administration. These standard reports are provided for the server instances, databases, logins, and Management tree item. Each runs as a Reporting Services report inside of SQL Server Management Studio. Server-level reports give you information about the instance of SQL Server and the operating system. Database-level reports drill into information about each database. You must have access to each database you want to report on, or your login must have enough rights to run the server-level report. You also have the capability to write custom reports and attach them to many other nodes in the Object Explorer window.

Server Reports

You can access server-level reports from the Object Explorer window in Management Studio by right-clicking an instance of SQL Server and selecting Reports from the menu. A report favorite at the server level is the Server Dashboard, which is shown in Figure. The Server Dashboard report gives you a wealth of information about your SQL Server 2012 instance, including:

- The edition and version of SQL Server you run
- Anything for that instance not configured to the default SQL Server settings
- The I/O and CPU statistics by type of activity (for example, ad hoc queries, Reporting Services, and so on)
- High-level configuration information such as whether the instance is clustered



Most of the statistical information includes only data gathered since the last time you started SQL Server. For example, the Server Dashboard provides a few graphs that show CPU usage by type of query. This graph is not historical; it shows you the CPU usage only for the period of

time that SQL Server has been online. Use caution when extrapolating information from this aspect of the Server Dashboard. Always keep in mind that what you see is a time-sensitive snapshot of server performance, not its entire history.

Database Reports

Database reports operate much like server-level reports. Select them by right-clicking the database name in the Object Explorer window in Management Studio. With these reports, you can see information that pertains to the database you selected. For example, you can see all the transactions currently running against a database, users being blocked, or disk utilization for a given database, as shown in Figure

Disk Usage
[AdventureWorks]
on ENDYMION\MSSQLSERVER at 1/24/2012 11:57:05 AM

This report provides overview of the utilization of disk space within the Database.

Total Space Usage: 182.88 MB
Data Files Space Usage: 176.75 MB
Transaction Log Space Usage: 6.13 MB

Data Files Space Usage (%)

Index	495.14
Unallocated	17.11
Data	2.93
Unused	-415.18

Transaction Log Space Usage (%)

Used	40.8
Unused	59.2

No entry found for autogrow/autoshrink event for AdventureWorks database in the trace log.

Disk Space Used by Data Files

Filegroup Name	Logical File Name	Physical File Name	Space Reserved	Space Used
PRIMARY	AdventureWorks_Data	C:\Database\MSSQLSERVER\Data\AdventureWorks_Data.mdf	176.75 MB	146.63 MB

Using sp_configure

sp_configure is a stored procedure that enables you to change many of the configuration options in SQL Server. Some of the more commonly adjusted settings include:

- Cost threshold for parallelism: Use to mitigate parallelism on lower costing transactions to reduce the need to modify the max degree of parallelism.

- Max degree of parallelism: For OLTP environments, this is usually set to the number of available sockets.
- CLR enabled: Enables CLR procedures to execute.
- Blocked processes threshold: Use to set time threshold before blocked process reports are generated.

When you run the stored procedure with no parameters, it shows you the options and their current settings. By default, only the basic, or commonly used, settings are returned, of which there are 15. To see additional options, you need to configure the instance to display advanced options. You can do so by running `sp_configure`, as shown here:

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

```
RECONFIGURE;
```

The change does not take effect until the `RECONFIGURE` command is issued. SQL Server does a check for invalid or not recommended settings when you use `sp_configure`. If you have provided a value that fails any of these checks, SQL Server warns you with the following message:

Msg 5807, Level 16, State 1, Line 1

Recovery intervals above 60 minutes not recommended. Use the `RECONFIGURE WITH OVERRIDE` statement to force this configuration.

Note: *Issuing the RECONFIGURE command results in a complete flush of the plan cache. This means that nothing remains in cache, and all batches are recompiled when submitted for the first time afterward. Be cautious of using sp_configure in a production environment and make sure that you understand the impact of the cache flush on your database.*

This gives you an opportunity to reconsider what may have been a bad choice, such as setting a memory option to more memory than exists on the box. The following code shows you how to issue the override for the setting:

```
EXEC sp_configure 'recovery interval', 90;
```

```
RECONFIGURE WITH OVERRIDE;
```

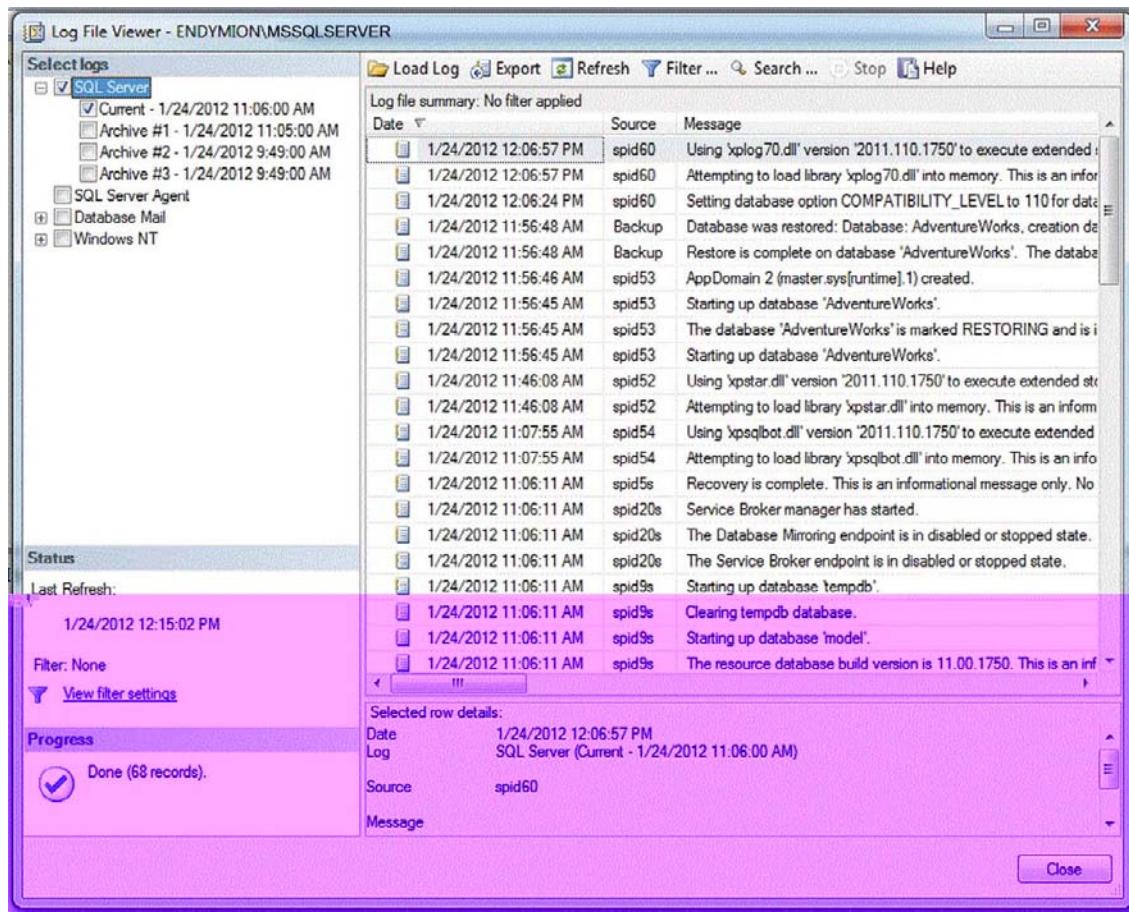
```
GO
```

Error Logs

As you probably have already experienced, when something goes wrong with an application, one of the factors that you must consider is the database. It is up to the DBA to support the troubleshooting effort and to confirm that the database isn't the source of the problem. The

first thing the DBA typically does is connect to the server and look at the SQL Server instance error logs and then the Windows event logs.

In SQL Server 2012, you can quickly look through the logs in a consolidated manner using Management Studio. To view the logs, right-click SQL Server Logs under the Management tree, and select View → SQL Server and Windows Log. This opens the Log File Viewer screen. From this screen, you can check and uncheck log files that you want to bring into the view. You can consolidate logs from SQL Server, Agent, and the Windows Event Files, as shown in Figure

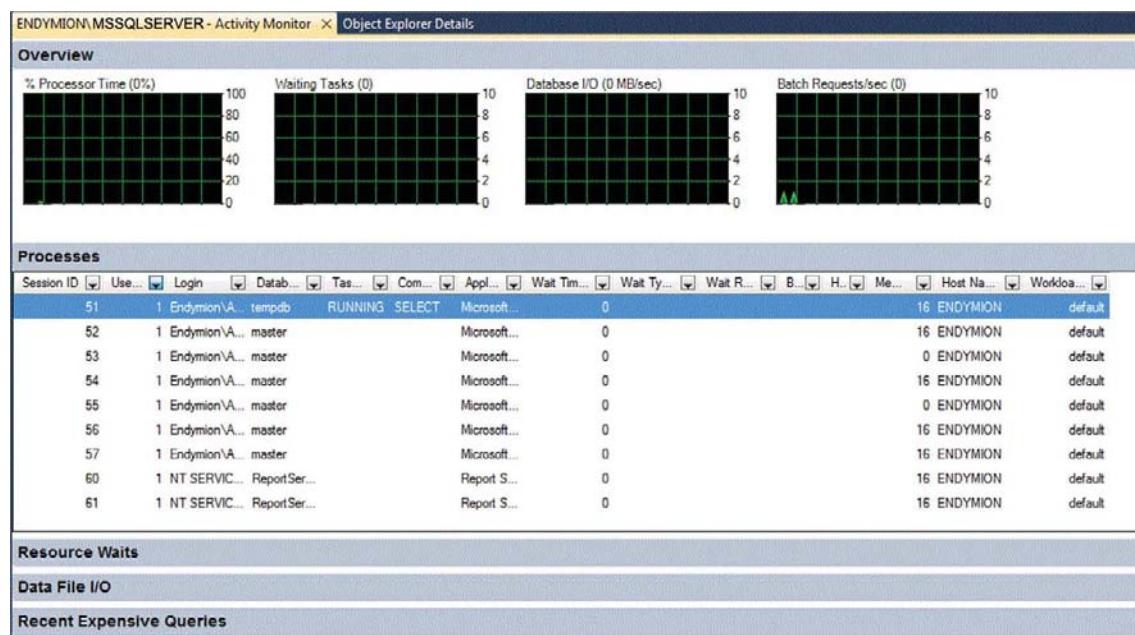


In some situations, you may want to merge the logs from several machines into a single view to determine what's causing an application problem. To do this, click the Load Log button, and browse to your .LOG file. That file could be a Windows error log that has been output to .LOG format or a SQL log from a different instance. For example, you can use this to consolidate all the SQL logs from every instance on a single server to give you a holistic view of all the physical machines problems.

Activity Monitor – SSMS 2012

The Activity Monitor gives you a view of current connections on an instance. You can use the monitor to determine whether you have any processes blocking other processes. To open the Activity Monitor in Management Studio, right-click the Server in the Object Explorer, and then select Activity Monitor.

The tool is a comprehensive way to view who connects to your machine and what they do. The top section shows four graphs (Show Processor Time, Waiting Tasks, Database I/O, and Batch Requests/Sec) that are commonly used performance counters for the server. There are four lists under the graphs: Processes, Resource Waits, Data File I/O, and Recent Expensive Queries. In all these lists, you can also apply filters to show only certain hosts, logins, or connections using greater than a given number of resources. You can also sort by a given column by clicking the column header. On the Process Info page (shown in Figure), you can see each login connecting to your machine (also called a Server Process ID, or SPID). It's easy to miss how much information is in this window. You can slide left to right to see loads of important data about each connection. When debugging, most of the following columns are useful:



Session ID: The unique number assigned to a process connected to SQL Server. This is also called a SPID. An icon next to the number represents what happens in the connection. If you see an hourglass, you can quickly tell that the process is waiting on or is being blocked by another connection.

User Process Flag: Indicates whether internal SQL Server processes are connected. These processes are filtered out by default. You can change the value to see the SQL Server internal processes by clicking the drop-down and selecting the appropriate value.

Login: The login to which the process is tied.

Database: The current database context for the connection.

Task State: Indicates whether the user is active or sleeping.

- **Done:** Completed.
- **Pending:** The process is waiting for a worker thread.
- **Runnable:** The process has previously been active, has a connection, but has no work to do.
- **Running:** The process is currently performing work.
- **Suspended:** The process has work to do, but it has been stopped. You can find additional information about why the process is suspended in the Wait Type column.

Command: Shows the type of command currently being executed. For example, you may see SELECT, DBCC, INSERT, or AWAITING COMMAND here, to name a few. This won't show you the actual query that the user executes, but it does highlight what type of activity is being run on your server. To see the actual command, select a row in this table, right-click, and choose Details.

Application: The application that is connecting to your instance. This can be set by the developer in the connection string.

Wait Time (ms): If the process is blocked or waiting for another process to complete, this indicates how long the process has been waiting, in milliseconds; it can have a value of 0 if the process is not waiting.

Wait Type: Indicates the event you are waiting on.

Wait Resource: The text representation of the resource you are waiting on.

Blocked By: The Session ID (SPID) that is blocking this connection.

Head Blocker: A value of 1 means the Blocked By Session ID is the head of the blocking chain; otherwise it's 0.

Memory Use (KB): The current amount of memory used by this connection, represented by the number of kilobytes in the Procedure cache attributed to this connection. This was reported in pages prior to SQL Server 2008.

Host: The login's workstation or server name. This is a useful item, but in some cases you may have a Web server connecting to your SQL Server, which may make this less important.

Workload Group: The name of the Resource Governor Workload group for this query.

Another important function you can perform with the help of Activity Monitor is identifying a locking condition. The following steps explain how to set up a blocked transaction and how to use activity monitor to resolve this issue.

1. Run the following query in one query window while connected to the AdventureWorks database, and ensure you back up the AdventureWorks database before performing these steps:

```
BEGIN TRAN
```

```
DELETE FROM Production.ProductCostHistory
```

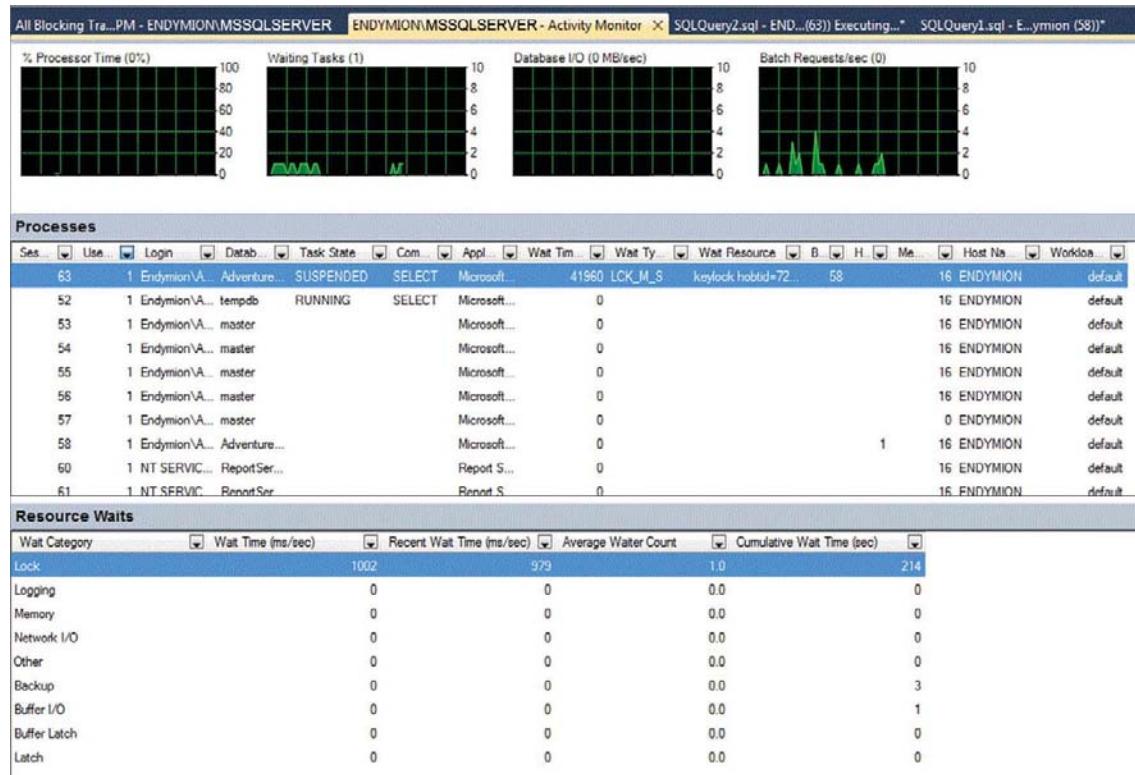
WHERE ProductID = 707;

This query was intentionally not committed. In other words, there is a BEGIN TRAN command but no ROLLBACK or COMMIT command. This means that the rows deleted from Production.ProductCostHistory are still locked exclusively.

2. Next, without closing the first window, open a new query window, and run the following query:

```
SELECT * FROM Production.ProductCostHistory;
```

This query should hang up and wait on the DELETE from Production.ProductCostHistory because the first transaction locks row one. Do not close either window. At the top of each query window, your session ID displays in parentheses, and at the bottom your login displays. If you cannot see the SPID in the tab at the top, hover your mouse over the tab, and a small window pops up showing the entire tab title, which includes the SPID. While the query windows are open, go ahead and explore the Activity Monitor to see what these connections look like (Figure).



3. Open the Activity Monitor and note that one connection has a task state of Suspended. This is the query that is trying to do the SELECT. You can confirm this by comparing the session ID of the suspended session with the session ID of your query. Your wait type will be LCK_M_S, which means you are waiting on a shared lock for reading. If you hover the mouse over the Wait Resource column value, you can see more detailed information about the locks, including the object IDs of the resources. In the Blocked By column, you can also see the session ID of

the process that blocks you, and it should match the SPID of your first query. You have the option to kill the blocking process; to do so, select the row for the blocking process, right-click, and choose Kill Process.

4. While the locks still exist, take a look at a standard blocking report that comes with SQL Server 2012. In Object Explorer, right-click on your server name, select Reports, Standard Reports, and choose Activity -All Blocking Transactions. You can see the report shown in Figure

The screenshot shows the 'All Blocking Transactions' report for the AdventureWorks database on ENDYMION\MSSQLSERVER at 1/24/2012 12:26:09 PM. The report title is 'All Blocking Transactions [AdventureWorks] on ENDYMION\MSSQLSERVER at 1/24/2012 12:26:09 PM'. It includes a Microsoft SQL Server logo. The report details a blocking transaction (Session 55821) that is directly blocking another transaction (Session 62885). The blocked SQL statement is 'SELECT * FROM Production.ProductCostHistory;'. The report also includes notes about transactions spanning multiple databases and being distributed.

Transaction ID	# Directly Blocked Transactions	# Indirectly Blocked Transactions	Transaction Name	State	Transaction Type	Start Time	Resource Type	Session ID	Blocking SQL Statement
55821	1	0	user_transaction	Active	Full Transaction	1/24/2012 12:21:01 PM	KEY	58	...

Blocking SQL Statement								
Direct/Indirect	Blocked Transaction ID	Blocked Transaction Name	State	Transaction Type	Start Time	Resource Type	Session ID	Blocked SQL Statement
Direct	62885*	SELECT	Active	Read Only	1/24/2012 12:24:21 PM	KEY	63	SELECT * FROM Production.ProductCostHistory;

* The transactions span multiple databases on the same instance.
* The transactions are distributed transactions.

5. Now, back at the Activity Monitor, hover over the Wait Resource column to see the mode=X. This mode means that there is an exclusive lock on that resource. An exclusive lock means that no one else is allowed to read the data cleanly. If you see a request mode of S, then SQL Server has granted a shared lock, which in most situations is harmless, and others are allowed to see the same data. To request a dirty read of the uncommitted data, add a WITH (NOLOCK) clause like so:

```
SELECT *
FROM Production.ProductCostHistory
WITH (NOLOCK);
```

6. This query returns the data. Before you leave this section, execute the following SQL in the query window that contains the DELETE statement:

What are the permissions required to use Activity Monitor?

If you are connecting to SQL 2005 or SQL 2008 then the user should have VIEW SERVER STATE permission. If you are connecting to SQL 2000 then the user should have select permission on sysprocesses and syslocks tables in master table. By default public will have the privilege.

MONITORING PROCESSES IN T-SQL

You can also monitor the activity of your server via T-SQL. Generally, DBAs prefer this as a quick way to troubleshoot long-running queries or users who complain about slow performance. DBAs typically prefer T-SQL because the information you can retrieve is more flexible than the Activity Monitor.

sp_who and sp_who2

The `sp_who` stored procedure returns what is connecting to your instance, much like the Activity Monitor. You'll probably prefer the undocumented `sp_who2` stored procedure, though, which gives you more verbose information about each process. Whichever stored procedure you use, they both accept the same input parameters. For the purpose of this discussion, we go into more detail about `sp_who2`. Just keep in mind that `sp_who` shows a subset of the information.

To see all the connections to your server, run `sp_who2` without any parameters. This displays the same type of information in the Activity Monitor. You can also pass in the parameter of 'active' to see only the active connections to your server, like so:

```
sp_who2 'active';
```

Additionally, you can pass in the SPID, as shown here, to see the details about an individual process:

```
sp_who2 55;
```

sys.dm_exec_connections

The `sys.dm_exec_connections` dynamic management view (DMV) gives you even more information to help you troubleshoot the Database Engine of SQL Server. This DMV returns a row per session in SQL Server. Because it's a DMV, it displays as a table and enables you to write sophisticated queries against the view to filter out what you don't care about, as shown in the following query, which shows only user connections that have performed a write operation:

```
SELECT * FROM  
sys.dm_exec_sessions  
WHERE is_user_process = 1  
AND writes > 0;
```

In addition to the information shown in the methods described earlier to view processes, this DMV indicates how many rows the user has retrieved since opening the connection, and the number of reads, writes, and logical reads. You can also see in this view the settings for each connection and what the last error was, if any.

Case study: How to create new error log or recycle error logs in sql server.

SQL Server: Every time you restart the SQL Server a new error log will be created. The old error will be renamed as ERRORLOG.1. In most of the production environment the SQL Server will be restarted rarely and you will notice a large ERRORLOG.

In order to recycle or create a new error log you no need to restart the SQL Server. You can recycle the error log using below command.

```
EXEC sp_cycle_errorlog  
GO
```

you can limit the number of error log files before they are recycled.

1. In Object Explorer, Click the plus sign to expand Management.
2. Right-click on the SQL Server Logs folder and select Configure.
3. In the Configure SQL Server Error Logs dialog box, you can enter maximum number of Error logs.

SQL Server Agent also has an error and it will be recreated every time you restart SQL Server Agent. The new error log will be generated as SQLAGENT.OUT and old error log will be renamed as SQLAGENT.1...etc. The SQL Server Agent can maintain up to 9 SQL Server Agent Error logs.

You can recycle the SQL Server Agent Error log using below

```
EXEC sp_cycle_agent_errorlog  
GO
```

Best Practices on Monitoring

1. Check OS Event Logs, SQL Server Logs, and Security Logs for unusual events.
2. Verify that all scheduled jobs have run successfully.
3. Confirm that backups have been made and successfully saved to a secure location.
4. Monitor disk space to ensure your SQL Servers won't run out of disk space.
5. Throughout the day, periodically monitor performance using both System Monitor and Profiler.
6. Use Enterprise Manager/Management Studio to monitor and identify blocking issues.
7. Keep a log of any changes you make to servers, including documentation of any performance issues you identify and correct.
8. Create SQL Server alerts to notify you of potential problems, and have them emailed to you. Take actions as needed.
9. Run the SQL Server Best Practices Analyzer on each of your server's instances on a periodic basis.

Chapter – 14

Transaction Log Architecture

1. Transaction Log Logical Architecture

The SQL Server transaction log operates logically as if the transaction log is a string of log records. Each log record is identified by a log sequence number (LSN). Each new log record is written to the logical end of the log with an LSN that is higher than the LSN of the record before it.

Log records are stored in a serial sequence as they are created. Each log record contains the ID of the transaction that it belongs to. For each transaction, all log records associated with the transaction are individually linked in a chain using backward pointers that speed the rollback of the transaction.

Many types of operations are recorded in the transaction log. These operations include:

- The start and end of each transaction.
- Every data modification (insert, update, or delete). This includes changes by system stored procedures or data definition language (DDL) statements to any table, including system tables.
- Every extent and page allocation or de-allocation.
- Creating or dropping a table or index.

Rollback operations are also logged. Each transaction reserves space on the transaction log to make sure that enough log space exists to support a rollback that is caused by either an explicit rollback statement or if an error is encountered. The amount of space reserved depends on the operations performed in the transaction, but generally is equal to the amount of space used to log each operation. This reserved space is freed when the transaction is completed.

The section of the log file from the first log record that must be present for a successful database-wide rollback to the last-written log record is called the active part of the log, or the *active log*. This is the section of the log required to do a full recovery of the database. No part of the active log can ever be truncated.

2. Transaction Log Physical Architecture

The transaction log is used to guarantee the data integrity of the database and for data recovery. The topics in this section provide the information about the physical architecture of the transaction log. Understanding the physical architecture can improve your effectiveness in managing transaction logs.

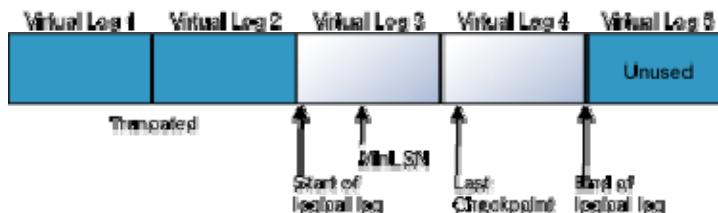
The transaction log in a database maps over one or more physical files. Conceptually, the log file is a string of log records. Physically, the sequence of log records is stored efficiently in the set of physical files that implement the transaction log.

The SQL Server Database Engine divides each physical log file internally into a number of virtual log files. Virtual log files have no fixed size, and there is no fixed number of virtual log files for a physical log file. The Database Engine chooses the size of the virtual log files

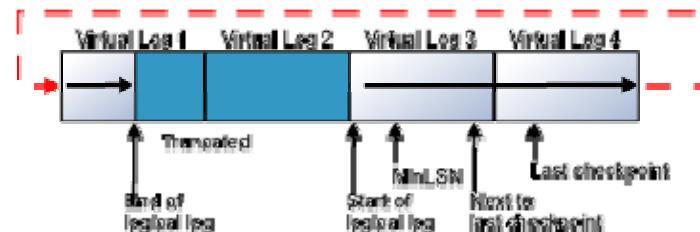
dynamically while it is creating or extending log files. The Database Engine tries to maintain a small number of virtual files. The size of the virtual files after a log file has been extended is the sum of the size of the existing log and the size of the new file increment. The size or number of virtual log files cannot be configured or set by administrators.

The only time virtual log files affect system performance is if the log files are defined by small *size* and *growth_increment* values. If these log files grow to a large size because of many small increments, they will have lots of virtual log files. This can slow down database startup and also log backup and restore operations. We recommend that you assign log files a *size* value close to the final size required, and also have a relatively large *growth_increment* value.

The transaction log is a wrap-around file. For example, consider a database with one physical log file divided into four virtual log files. When the database is created, the logical log file begins at the start of the physical log file. New log records are added at the end of the logical log and expand toward the end of the physical log. Log truncation frees any virtual logs whose records all appear in front of the minimum recovery log sequence number (MinLSN). The *MinLSN* is the log sequence number of the oldest log record that is required for a successful database-wide rollback. The transaction log in the example database would look similar to the one in the following illustration.



When the end of the logical log reaches the end of the physical log file, the new log records wrap around to the start of the physical log file.



This cycle repeats endlessly, as long as the end of the logical log never reaches the beginning of the logical log. If the old log records are truncated frequently enough to always leave sufficient room for all the new log records created through the next checkpoint, the log never fills. However, if the end of the logical log does reach the start of the logical log, one of two things occurs:

- If the FILEGROWTH setting is enabled for the log and space is available on the disk, the file is extended by the amount specified in *growth_increment* and the new log records are added to the extension. For more information about the FILEGROWTH setting, see ALTER DATABASE (Transact-SQL).

- If the FILEGROWTH setting is not enabled, or the disk that is holding the log file has less free space than the amount specified in *growth_increment*, an 9002 error is generated.

If the log contains multiple physical log files, the logical log will move through all the physical log files before it wraps back to the start of the first physical log file

3. Checkpoint Operation

Checkpoints flush dirty data pages from the buffer cache of the current database to disk. This minimizes the active portion of the log that must be processed during a full recovery of a database. During a full recovery, the following types of actions are performed:

- The log records of modifications not flushed to disk before the system stopped are rolled forward.
- All modifications associated with incomplete transactions, such as transactions for which there is no COMMIT or ROLLBACK log record, are rolled back.
- A checkpoint performs the following processes in the database:
- Writes a record to the log file, marking the start of the checkpoint.
- Stores information recorded for the checkpoint in a chain of checkpoint log records. One piece of information recorded in the checkpoint is the log sequence number (LSN) of the first log record that must be present for a successful database-wide rollback. This LSN is called the Minimum Recovery LSN (MinLSN). The MinLSN is the minimum of the:
 - LSN of the start of the checkpoint.
 - LSN of the start of the oldest active transaction.

The checkpoint records also contain a list of all the active transactions that have modified the database.

- If the database uses the simple recovery model, marks for reuse the space that precedes the MinLSN.
- Writes all dirty log and data pages to disk.
- Writes a record marking the end of the checkpoint to the log file.
- Writes the LSN of the start of this chain to the database boot page.

Activities That Cause a Checkpoint

Checkpoints occur in the following situations:

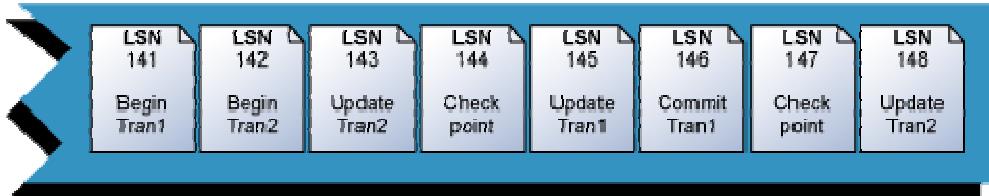
- A CHECKPOINT statement is explicitly executed. A checkpoint occurs in the current database for the connection.
- A minimally logged operation is performed in the database; for example, a bulk-copy operation is performed on a database that is using the Bulk-Logged recovery model.
- Database files have been added or removed by using ALTER DATABASE.

- An instance of SQL Server is stopped by a SHUTDOWN statement or by stopping the SQL Server (MSSQLSERVER) service. Either action causes a checkpoint in each database in the instance of SQL Server.
- An instance of SQL Server periodically generates automatic checkpoints in each database to reduce the time that the instance would take to recover the database.
- A database backup is taken.
- An activity requiring a database shutdown is performed. For example, AUTO_CLOSE is ON and the last user connection to the database is closed, or a database option change is made that requires a restart of the database.

Active Log

The section of the log file from the MinLSN to the last-written log record is called the active portion of the log, or the *active log*. This is the section of the log required to do a full recovery of the database. No part of the active log can ever be truncated. All log records must be truncated from the parts of the log before the MinLSN.

The following illustration shows a simplified version of the end-of-a-transaction log with two active transactions. Checkpoint records have been compacted to a single record.



LSN 148 is the last record in the transaction log. At the time that the recorded checkpoint at LSN 147 was processed, Tran 1 had been committed and Tran 2 was the only active transaction. That makes the first log record for Tran 2 the oldest log record for a transaction active at the time of the last checkpoint. This makes LSN 142, the Begin transaction record for Tran 2, the MinLSN.

Long-Running Transactions

The active log must include every part of all uncommitted transactions. An application that starts a transaction and does not commit it or roll it back prevents the Database Engine from advancing the MinLSN. This can cause two types of problems:

- If the system is shut down after the transaction has performed many uncommitted modifications, the recovery phase of the subsequent restart can take much longer than the time specified in the **recovery interval** option.
- The log might grow very large, because the log cannot be truncated past the MinLSN. This occurs even if the database is using the simple recovery model, in which the transaction log is generally truncated on each automatic checkpoint.

4. Write-Ahead Transaction Log

SQL Server uses a write-ahead log (WAL), which guarantees that no data modifications are written to disk before the associated log record is written to disk.

To understand how the write-ahead log works, it is important for you to know how modified data is written to disk. SQL Server maintains a buffer cache into which it reads data pages when data must be retrieved. Data modifications are not made directly to disk, but are made to the copy of the page in the buffer cache. The modification is not written to disk until a checkpoint occurs in the database, or the modification must be written to disk so the buffer can be used to hold a new page. Writing a modified data page from the buffer cache to disk is called flushing the page. A page modified in the cache, but not yet written to disk, is called a *dirty page*.

At the time a modification is made to a page in the buffer, a log record is built in the log cache that records the modification. This log record must be written to disk before the associated dirty page is flushed from the buffer cache to disk. If the dirty page is flushed before the log record is written, the dirty page creates a modification on the disk that cannot be rolled back if the server fails before the log record is written to disk. SQL Server has logic that prevents a dirty page from being flushed before the associated log record is written. Log records are written to disk when the transactions are committed.

Managing the Transaction Log

Log truncation, which is automatic under the simple recovery model, is essential to keep the log from filling. The truncation process reduces the size of the logical log file by marking as inactive the virtual log files that do not hold any part of the logical log. In some cases, however, physically shrinking or expanding the physical log file is useful.

Transaction Log Truncation

If log records were never deleted from the transaction log, it would eventually fill all the disk space that is available to the physical log files. Log truncation automatically frees space in the logical log for reuse by the transaction log.

Except when delayed for some reason, log truncation occurs automatically as follows:

- Under the simple recovery model, after a checkpoint.
- Under the full recovery model or bulk-logged recovery model, after a log backup, if a checkpoint has occurred since the previous backup

How Log Truncation Works

Truncation does not reduce the size of a physical log file. Reducing the physical size of a log file requires shrinking the file.

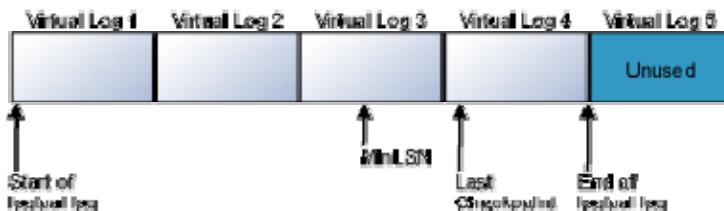
The transaction log is a wrap-around file. When the database is created, the logical log file begins at the start of the physical log file. New log records are added at the end of the logical log and expand toward the end of the physical log. The transaction log in a database maps over one or more physical files. The SQL Server Database Engine divides each physical log file internally into a number of virtual log files. Log truncation frees space in the logical log by deleting inactive virtual log files from the start of the logical log.

Virtual log files are the unit of space that can be reused. Only virtual log files that contain just inactive log records can be truncated. The active portion of the transaction log, the *active log*, cannot be truncated, because the active log is required to recover the database. The most recent checkpoint defines the active log. The log can be truncated up to that checkpoint.

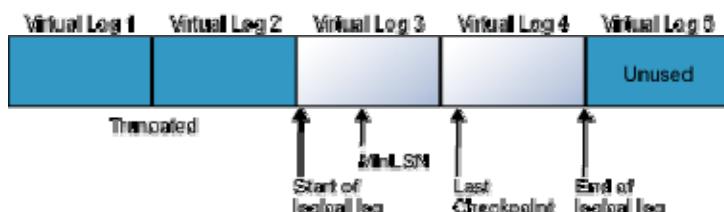
When the checkpoint is performed, the inactive portion of the transaction log is marked as reusable. Thereafter, the inactive portion can be freed by log truncation. Truncation frees the inactive virtual log files for reuse. Eventually, when a new record is written to a freed virtual log, that virtual log file becomes active again.

One piece of information recorded in a checkpoint is the log sequence number (LSN) of the first log record that must be present for a successful database-wide rollback. This LSN is called the *minimum recovery LSN (MinLSN)*. The start of the active portion of the log is the virtual log that contains the MinLSN. When a transaction log is truncated, only the log records in front of this virtual log file are freed for reuse.

The following illustrations show a transaction log before and after truncation. The first illustration shows a transaction log that has never been truncated. Currently, four virtual log files are in use by the logical log. The logical log starts at the front of the first virtual log file and ends at virtual log 4. The MinLSN record is in virtual log 3. Virtual log 1 and virtual log 2 contain only inactive log records. These records can be truncated. Virtual log 5 is still unused and is not part of the current logical log.



The second illustration shows how the log appears after being truncated. Virtual log 1 and virtual log 2 have been freed for reuse. The logical log now starts at the beginning of virtual log 3. Virtual log 5 is still unused, and it is not part of the current logical log.



Managing the Size of the Transaction Log File

Monitoring Log Space Use

You can monitor log space use by using DBCC SQLPERF (LOGSPACE). This command returns information about the amount of log space currently used and indicates when the transaction log is in need of truncation.

For information about the current size of a log file, its maximum size, and the autogrow option for the file, you can also use the **size**, **max_size**, and **growth** columns for that log file in **sys.database_files**.

Shrinking the Size of the Log File

Log truncation is essential because it frees disk space for reuse, but it does not reduce the size if the physical log file. To reduce its physical size, the log file must be shrunk to remove one or more virtual log files that do not hold any part of the logical log (that is, *inactive virtual log files*). When a transaction log file is shrunk, enough inactive virtual log files are removed from the end of the log file to reduce the log to approximately the target size.

Shrinking the Transaction Log

If you know that a transaction log file contains unused space that you will not be needing, you can reclaim the excess space by reducing the size of the transaction log. This process is known as *shrinking* the log file.

Shrinking can occur only while the database is online and, also, while at least one virtual log file is free. In some cases, shrinking the log may not be possible until after the next log truncation.

DBCC SHRINKFILE (Transact-SQL)

Shrinks the size of the specified data or log file for the current database or empties a file by moving the data from the specified file to other files in the same filegroup, allowing the file to be removed from the database. You can shrink a file to a size that is less than the size specified when it was created. This resets the minimum file size to the new value.

Shrinking a data file to a specified target size

The following example shrinks the size of a data file named DataFile1 in the UserDB user database to 7 MB.

```
USE UserDB;
GO
DBCC SHRINKFILE (DataFile1, 7);
GO
```

Chapter – 15

Backup & Restore

Introduction

Backing up a database is one of the most important things you need to do when having a database driven application. It's only all of your data in there, right? But often developers and management don't realize the importance of backups and overall proper backup strategy for the most important side of the business – data and it's consistency.

How backup works

When you run a backup database command SQL Server performs a Checkpoint on the data pages in memory. Checkpoint means that all transactionally committed dirty pages are written to disk. Dirty pages are simply changed pages in memory that haven't been written to disk yet. After this the data on the disk is backed up in one or multiple files depending on your requirements. A backup must be able to be restored to a transactionally consistent state which means that it will also contain the data from the transaction log needed to undo all of the transactions which are running while the backup is taken.

Recovery models

Full recovery model

Every DML (insert, update, delete) statement is fully logged. Also every bulk insert operation (BCP, BulkInsert, SqlBulkCopy in .Net, SELECT INTO) is fully logged for each row. This recovery model also logs every CREATE INDEX and ALTER INDEX statement which are DDL operations. This means that when you recover a transaction log that contains logged CREATE INDEX you don't have to rebuild an index since it's already built. This behavior has changed since SQL Server 2000 where only the index creation event was logged, and not the whole index. The downside of this model is that it can consume a lot of disk space very fast.

Bulk-Logged recovery model

This model differs from the Full recovery in that it doesn't log every row insert on BULK operations I mentioned in Full recovery model. SQL Server does log that the operation has been executed and information about the data page allocations. However all data can still be restored. This is handled by the Bulk Change Map (BCM). SQL Server keeps track of the changed extents under this model by setting a bit representing an extent to 1 in the BCM, if that extent has changed. That is why the BULK operations are also faster under this model than under the Full Recovery, since logging each row is much slower than just setting a bit to 1. When you take a transaction log backup, all changed extents are also backed up by reading the BCM. This means that the transaction log itself is smaller but the transaction log backup can be a lot larger than the one under Full recovery model. Note that under this model you can't do point-in-time recovery on any transaction log backup that contains a bulk-logged transaction.

Simple recovery model

Under this recovery model you can't backup a transaction log at all. An attempt to do so results in an error, since there's nothing to update. The transaction log gets truncated at

every checkpoint (writing data from a log to a disk) which happens at predetermined intervals. Also changing the database recovery model to Simple will immediately truncate the transaction log. A common misunderstanding is that nothing is being logged under this model. That is NOT TRUE. Everything is logged, you just don't have the point-in-time recovery ability. Bulk operations are minimally logged as in Bulk-Logged recovery model.

Transaction log marks

These are only available under Full and Bulk logged recovery models. Log marks are set in the transaction you want to recover to. You can do this with the begin transaction statement:

```
BEGIN TRANSACTION TranMark1 WITH MARK 'The mark description'
```

The name of the mark in the transaction log is TranMark1. After this transaction commits the mark is inserted into the logmarkhistory table in the msdb database and into the transaction logs of other related databases. Related databases are beyond the scope of this article, but simply put: the databases are related when we make related updates to them (e.g.: update to table1 in db1 has to be followed by update to table2 in db2). With log marks over multiple databases we can recover all databases to a specific related state to each other.

Types of Backups:

Full database Backup

This backs up the whole database. In order to have further differential or transaction log backups you have to create the full database backup first.

```
-- Back up the AdventureWorks as full backup  
BACKUP DATABASE AdventureWorks  
TO DISK = N'c:\AdventureWorksDiff.bak'
```

Differential database backup

Differential database backups are cumulative. This means that each differential database backup backs up the all the changes from the last Full database backup and NOT last Differential backup.

```
-- Back up the AdventureWorks as differential backup  
BACKUP DATABASE AdventureWorks  
TO DISK = N'c:\AdventureWorksDiff.bak' WITH DIFFERENTIAL
```

Transaction log backup

Transaction log backup aren't possible under the Simple Recovery model. Two transaction logs can't contain the same transactions which means that when restoring you have to restore every backup in the order they were taken

```
-- Back up the AdventureWorks transaction log  
BACKUP LOG AdventureWorks  
TO DISK = N'c:\AdventureWorksLog.bak'
```

Tail log backup

There seems to be a lot of confusion about this one since it's a new term in SQL Server 2008 (I haven't heard it being used in SS2k). A Tail log backup is the last Transaction log backup that you make prior to restoring a database. What this means is that if your db crashes for whatever reason, you have to backup your transaction log so that you can do point in time recovery. This last backup is called Tail log backup. If your data file (MDF) is unavailable you need to use WITH NO_TRUNCATE option:

```
-- Back up the AdventureWorks tail-log  
BACKUP LOG AdventureWorks  
TO DISK = N'c:\AdventureWorksTailLog.bak' WITH NO_TRUNCATE
```

If your database is in OFFLINE or EMERGENCY state then tail log backup isn't possible.

Mirrored backup

Mirrored backups simply write the backup to more than one destination. You can write up to four mirrors per media set. This increases the possibility of a successful restore if a backup media gets corrupted. Following statement gives us two backups that we can restore from:

```
-- make one backup on d: disk and a mirror on e: disk  
BACKUP DATABASE AdventureWorks  
TO DISK = 'd:\AdventureWorksMirror_1.bak'  
MIRROR  
TO DISK = 'e:\AdventureWorksMirror_2.bak'  
-- create a new mirrored backup set  
WITH FORMAT;
```

Copy-only backup

Copy-only backups are new in SQL Server 2008 and are used to create a full database or transaction log backup without breaking the log chain. A copy-only full backup can't be used as a basis for a differential backup, nor can you create a differential copy only backup.

```
-- Back up the AdventureWorks database as copy only
BACKUP DATABASE AdventureWorks
    TO DISK = N'c:\AdventureWorksDiff.bak' WITH COPY_ONLY

-- Back up the AdventureWorks transaction log as copy only
BACKUP LOG AdventureWorks
    TO DISK = N'c:\AdventureWorksLog.bak' WITH COPY_ONLY
```

Explanation:

Generally each full backup serves as a base backup for any differential or log backups taken later, so by default backups affect other backups and how they will be restored. There may be a situation where you want to take a full database backup without affecting the current backup and restore procedures. Consider the below situation where you have scheduled as a job,

Daily @07:00 – Full backup

Daily @10:00 – Differential backup (Contains all the changes made after 07:00 till 10:00)

Daily @13:00 – Differential backup (Contains all the changes made after 07:00 till 13:00)

Daily @16:00 – Differential backup (Contains all the changes made after 07:00 till 16:00)

If you take a full backup of the database manually @11:00 then the subsequent

Differential backup @13:00 (Will contain only the changes made after 11:00 till 13:00). Hence the normal backup sequence will get affected because of this full backup.

In SQL 2005 you can make use of COPY_ONLY option to avoid this. Hence if you take a full backup of your database @11:00 using Copy_Only option the next differential backup @13:00 (Will contain only the changes made after 07:00 till 13:00). Hence by using this option the backup sequence will not get affected.

You can create a copy-only backup for any type of backup:

1. Copy-only data backups (all recovery models) to create a copy-only data or differential backup, use the COPY_ONLY option in your BACKUP DATABASE statement. A data backup taken with the COPY_ONLY option cannot be used as a base backup and does not affect any existing differential backups.

```
BACKUP DATABASE DEEPAK TO DISK='D:\TEST\DEEPAK.BAK' WITH COPY_ONLY
```

2. Copy-only differential backups These are identical to regular differential backups.

```
BACKUP DATABASE DEEPAK TO DISK='D:\TEST\DEEPAK.BAK' WITH
DIFFERENTIAL,COPY_ONLY
```

- 3.Copy-only log backups to create a copy-only log backup use the COPY_ONLY option in your BACKUP LOG statement. The transaction log is not truncated by a log backup taken using the COPY_ONLY option.

```
BACKUP LOG DEEPAK TO DISK='D:\TEST\DEEPAK.BAK' WITH COPY_ONLY
```

File and file group backup

If you have your data spread across multiple files or filegroups you can take a full or differential backup of file(s) or filegroup(s):

```
-- Backup AdventureWorks_file1 file of the  
-- AdventureWorks database to disk  
-- note that AdventureWorks has to have a AdventureWorks_file1 file  
BACKUP DATABASE AdventureWorks  
FILE = 'AdventureWorks_file1'  
TO DISK = 'e:\AdventureWorks_file1.bak'  
GO  
  
-- Backup AdventureWorks_filegroup1 filegroup of the  
-- AdventureWorks database to disk  
-- note that AdventureWorks has to have a  
-- AdventureWorks_filegroup1 filegroup  
BACKUP DATABASE AdventureWorks  
FILEGROUP = 'AdventureWorks_filegroup1'  
TO DISK = 'e:\AdventureWorks_filegroup1.bak'  
GO
```

For differential backups just add WITH DIFFERENTIAL option to the upper examples.

Partial Filegroup backup

Partial filegroup backups are used to backup large databases with one or more read-only filegroups. In a way they are similar to full database backup, but by default they don't include read-only files or filegroups. This means that they contain the primary filegroup, every read/write filegroup and an optional number of read only files or filegroups.

```
-- Create a partial filegroup backup. backs up all read/write filegroup  
-- and the read only AW_ReadOnly_FileGroup1 filegroup  
BACKUP DATABASE AdventureWorks  
READ_WRITE_FILEGROUPS, FILEGROUP = 'AW_ReadOnly_FileGroup1'
```

```
TO DISK = 'e:\AdventureWorks_partial.bak'
```

For differential backups just add WITH DIFFERENTIAL option to the upper example. Note that all file and file group names are logical and not physical names.

Restore:

Recovery states:

To determine the state of the database after the store operation, you must select one of the options of the Recovery state panel.

RESTORE WITH RECOVERY

Leave the database ready for use by rolling back the uncommitted transactions. Additional transaction logs cannot be restored.

Recovers the database after restoring the final backup checked in the Select the backup sets to restore grid on the General page. This is the default option and is equivalent to specifying WITH RECOVERY in a RESTORE statement (Transact-SQL).

```
RESTORE DATABASE [AdventureWorksNew]
    FROM  DISK =
N'\\nas\Backup\L40\SQL2008\AdventureWorks_backup_200702120215.bak'
    WITH FILE = 1,
        MOVE N'AdventureWorks_Data' TO
N'C:\Data\MSSQL.1\MSSQL\Data\AdventureWorksNew_Data.mdf',
        MOVE N'AdventureWorks_Log' TO
N'C:\Data\MSSQL.1\MSSQL\Data\AdventureWorksNew_Log.ldf'
```

RESTORE WITH NORECOVERY:

Leave the database non-operational, and do not roll back the uncommitted transactions. Additional transaction logs can be restored. ---Used in Mirroring

Leaves the database in the restoring state. This allows you to restore additional backups in the current recovery path. To recover the database, you will have to perform a restore operation by using the RESTORE WITH RECOVERY option (see the preceding option).

This option is equivalent to specifying WITH NORECOVERY in a RESTORE statement.

If you select this option, the Preserve replication settings option is unavailable.

RESTORE WITH STANDBY:

Leave the database in read-only mode. Undo uncommitted transactions, but save the undo actions in a standby file so that recovery effects can be reverted. () ---Used in log-shipping

Leaves the database in a standby state, in which the database is available for limited read-only access. This option is equivalent to specifying WITH STANDBY in a RESTORE statement.

Choosing this option requires that you specify a standby file in the Standby file text box. The standby file allows the recovery effects to be undone.

Standby file : Specifies a standby file. You can browse for the standby file or enter its pathname directly in the text box.

Practical troubleshooting

Case Study 1. **Backup & Restore for SQL Server 2012**

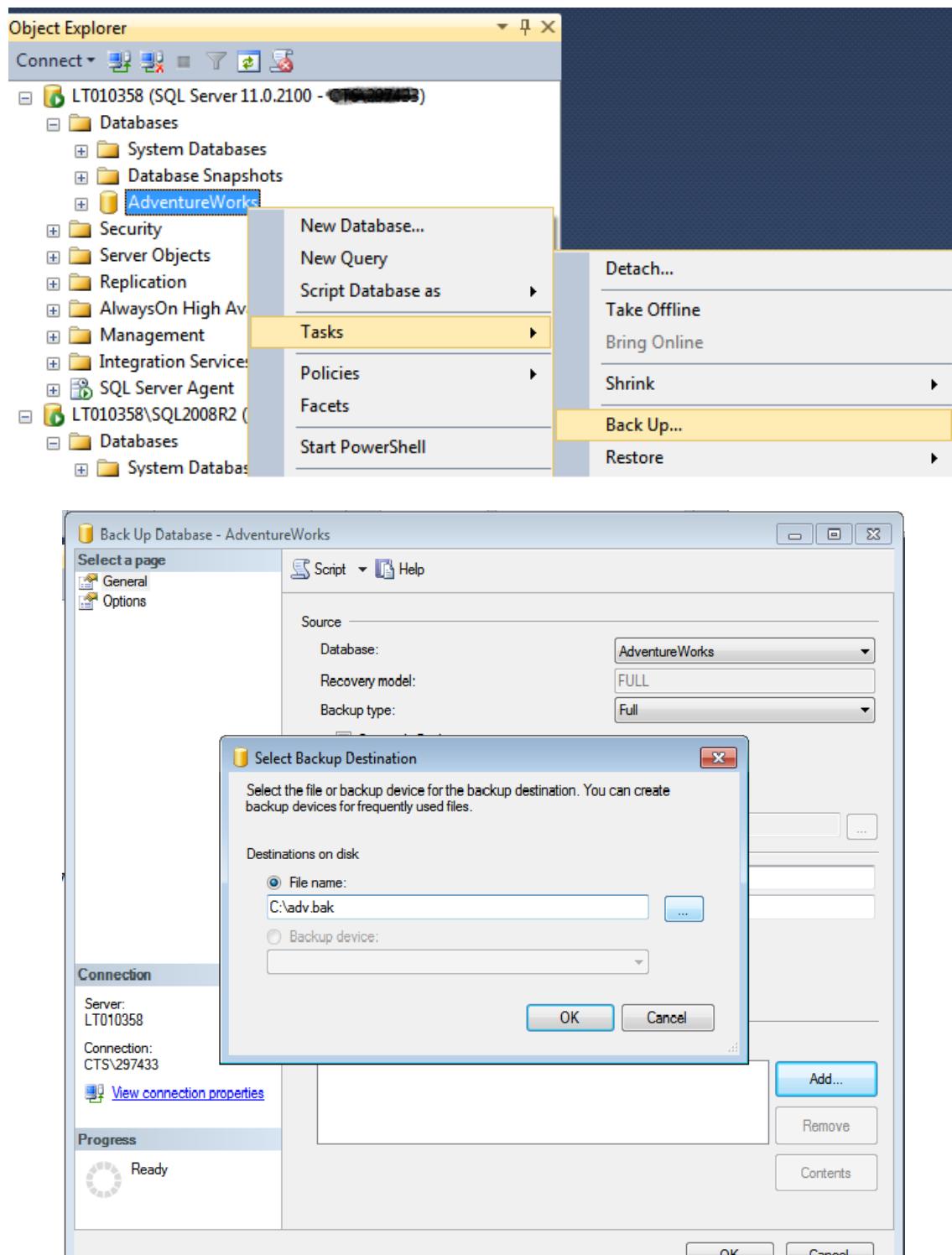
In a typical installation SQL Server stores its data in two files. One has an MDF extension and stores the data itself and the other has an LDF extension and stores the transaction log. You can configure SQL Server to have multiple data files and multiple transaction log files if you'd like but that's beyond the scope of this article. When SQL Server processes a transaction it goes through the following steps:

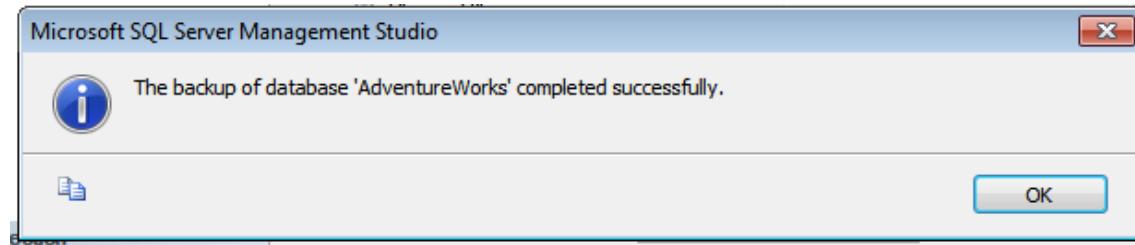
1. It writes what it's going to do to the transaction log.
2. It makes the change to the data file. This change is typically made on the in-memory copy of that portion of the data file.
3. It writes to the log that the transaction is committed.
4. The CHECKPOINT process writes the portion data file associated with the transaction to disk. This might happen anywhere from seconds to minutes after the step above.
5. It writes to the log that the transaction is "hardened".

The simplest type of backup is the Full Backup. The screen shots below are from SQL Server 2012 Management Studio (SSMS).

BACKUP:

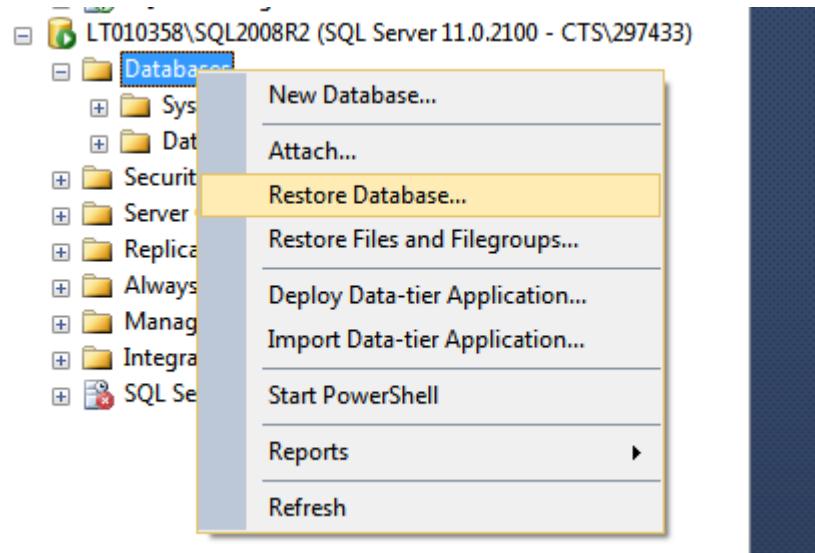
Go the primary server → goto the particular database → tasks → select the database → backup type as Full → add the backup location path → click ok



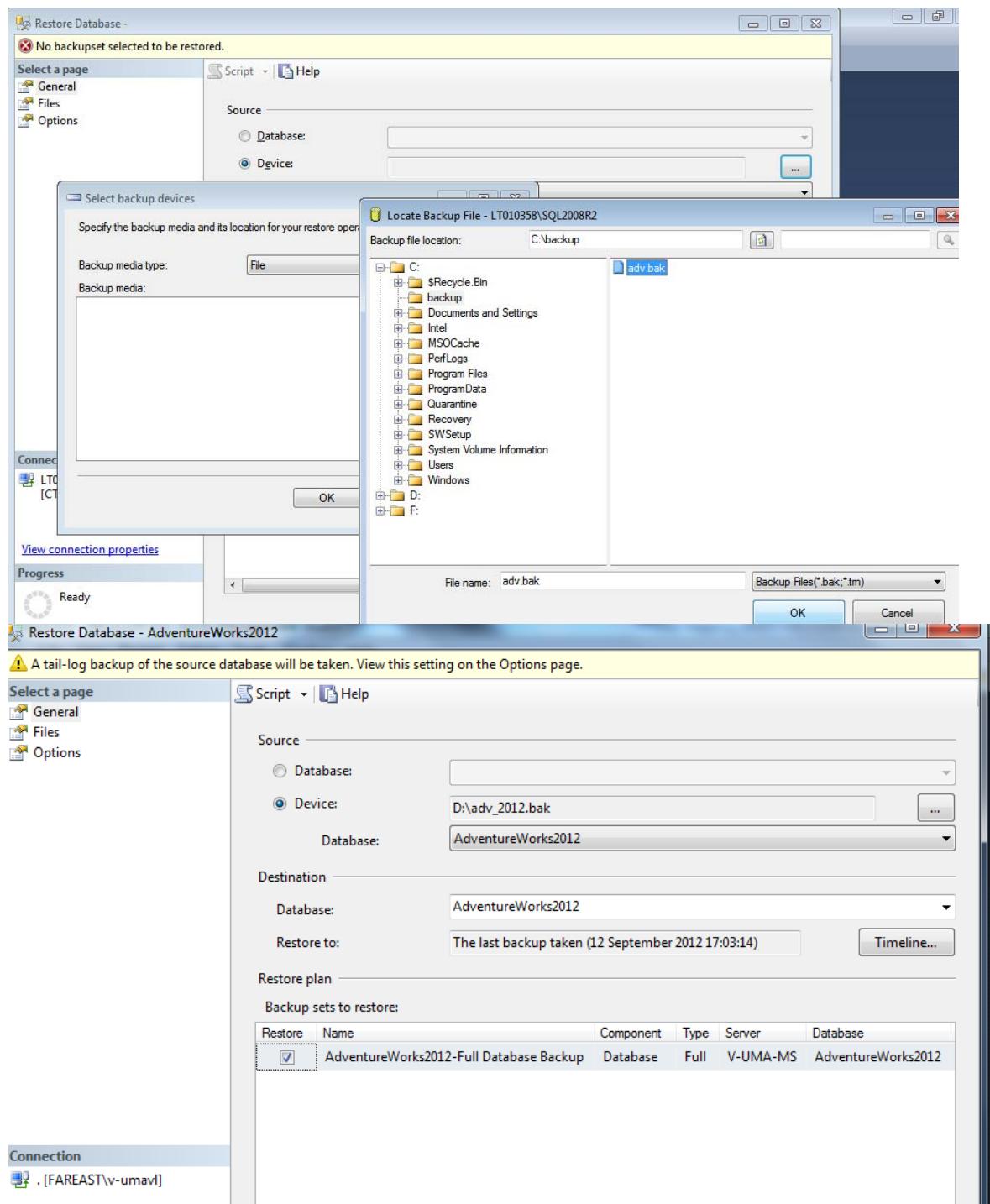


RESTORATION:

Go to the server → select databases → select restore database option → give the database name in General tab → select the from databases → give the backup file path

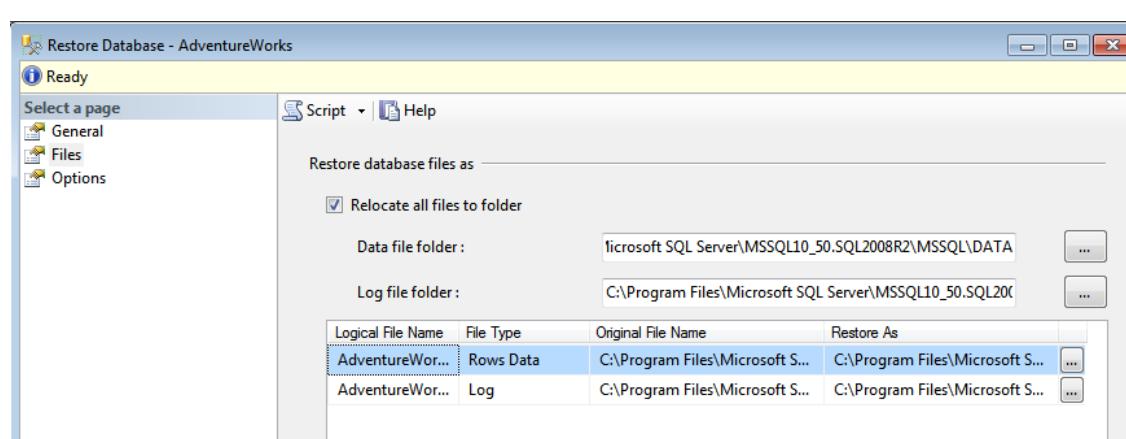
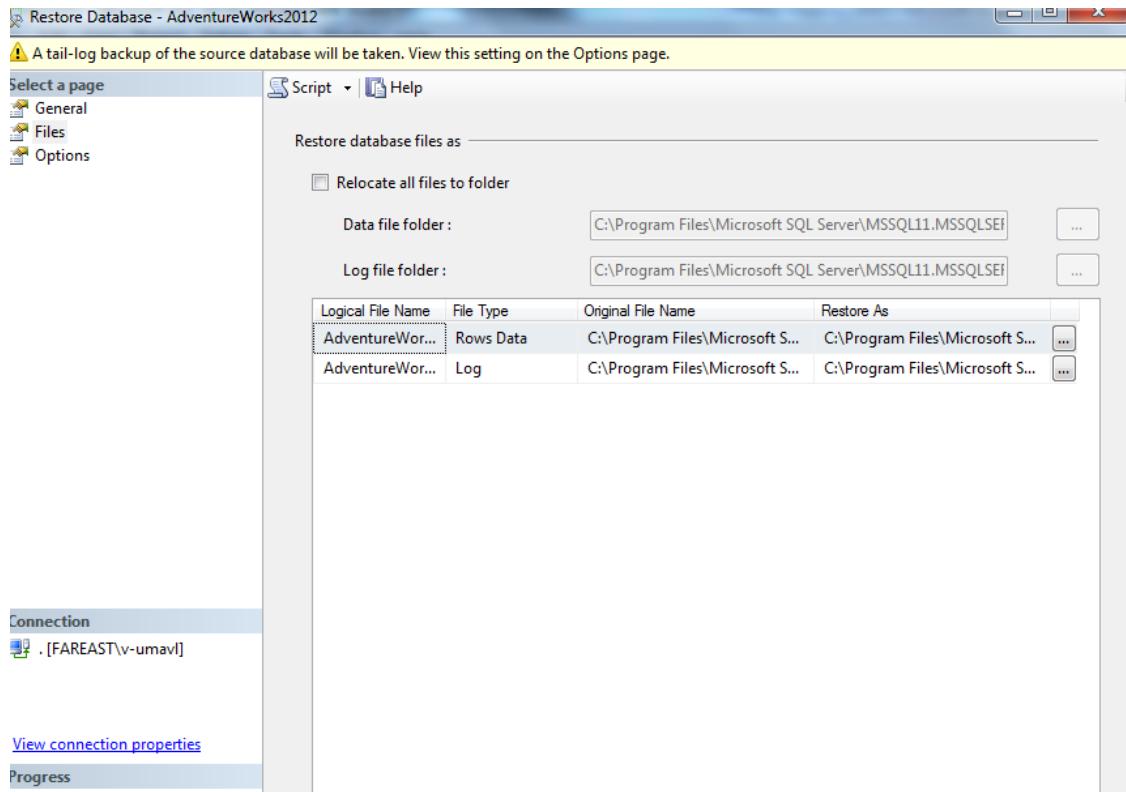


Select Device option → locate the backup location path →

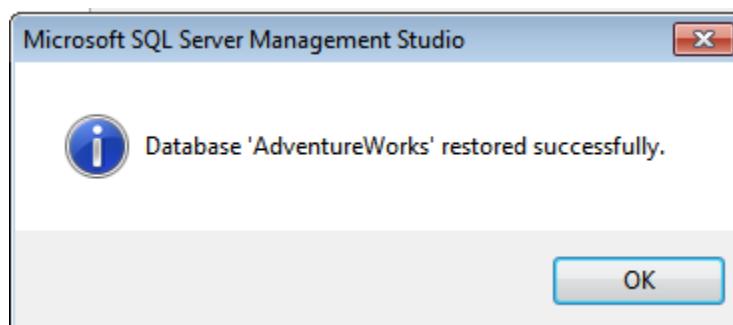
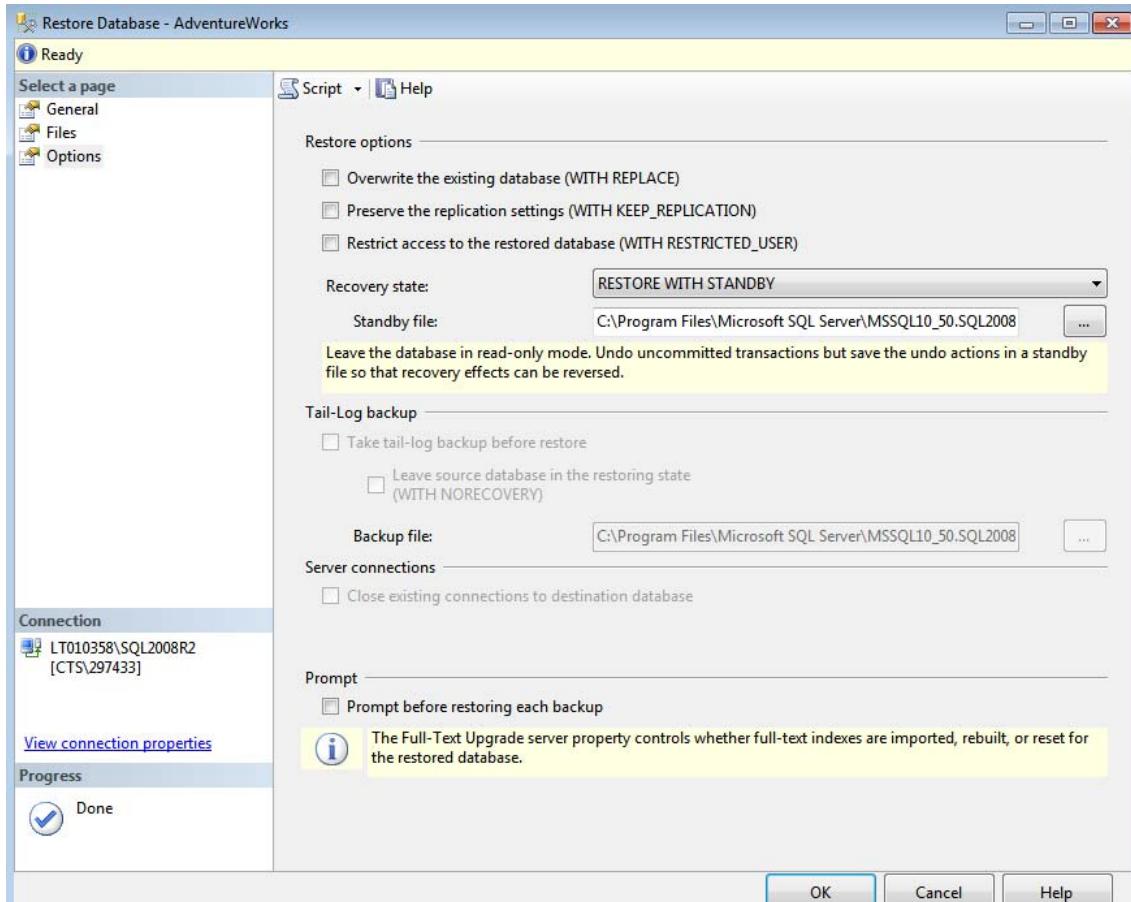


Go to files pane → give the correct data and log file path according to the that server settings
→ and click ok

Go to → option pane select the recovery option(stand by)—>click ok



Make sure that you are un-checking the tail-log checkbox.



When you create a database, SQL Server starts with a copy of the "model" database. If you set the Recovery Model of the "model" database to Simple all future databases will start out in Simple Recovery mode.

Case Study 2. How to restore a Suspect database:

If you find your Database in Suspect mode then please keep your nerve strong. Just proceed step by step what I am written bellow. I think you will get out of this trouble. SQL Server 2008 introduced a new DB Status called Emergency. This mode can change the DB from Suspect mode to Emergency mode, so that you can retrieve the data in read only mode. The steps are... After executing the script given below you will get back your Database in operational mode.

```
Exec Sp_resetstatus 'Yourdbname'  
Alter database yourdbname set emergency  
Dbcc checkdb(yourdbname)  
Alter database yourdbname set single_user with rollback immediate  
Dbcc checkdb ('yourdbname ', repair _allow_data_loss)  
ALTER DATABASE yourDBname SET MULTI_USER
```

Case Study 3. Backup and restore of the Resource database:

The **_Resource database** (shortly referred to as **RDB**) is a hidden, read-only database that contains all the system objects that are included with SQL Server 2005. This is the reason why it does not appear in SQL Server Management Studio. It contains all the system objects that ship with SQL Server 2005. These objects physically exist in the Resource database but logically appear in the **sys** schema of every database on the instance. It complements the **master** database in a sense as the SQL Server service now also depends on this database. The **Resource** database makes it easy for service packs to be applied or rolled back whenever necessary. In SQL Server 2000, whenever a service pack is applied, all the system objects that reside on both system and user databases will be updated, making it more difficult to rollback the change whenever necessary. It is also the reason why Microsoft recommends that you backup all the system and user databases before applying a service pack.

In SQL Server 2005, changes will only be made to the this database and will be reflected on all the system and user databases on the instance. If you need to apply a service pack on multiple instances, all you need to do is copy the Resource database's MDF and LDF files to the target instances. Rolling back the changes is as simple as overwriting the database files with an older copy. The physical file names of the **Resource** database are **mssqlsystemresource.mdf** and **mssqlsystemresource.ldf** and are located, by default, in **<drive>:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data**

Why is it important?

The **Resource** database appears to be a critical system database as the SQL Server service is now dependent on this. You can verify by renaming the database files while the service is stopped. You will not be able to start the service after this. You can also try moving the **master** database on a different location without moving the **Resource** database together with it and

you will not be able to start the service. Its location is dependent on the **master** database. This is critical during a disaster recovery process as we have gotten used to dealing with only the **master** database in previous versions.

Backing up the Resource database

Since the **Resource** database is not available from the SQL Server tools, we cannot perform a backup similar to how we do it with the other databases. You can backup the database using the following options:

1. You can use a simple **xcopy** command to copy from the source location to a destination where you keep your daily database backups. Use the **-Y** option to suppress the prompt to confirm if you want to overwrite the file. You can create a scheduled task to do this on a daily basis. If you want to keep multiple copies of the database files, you can create an automated script to rename them after the copy process.

```
xcopy <drive>:\Program Files\Microsoft SQL  
Server\MSSQL.1\MSSQL\Data\mssql\systemresource.mdf <destination folder> /Y  
  
xcopy <drive>:\Program Files\Microsoft SQL  
Server\MSSQL.1\MSSQL\Data\mssql\systemresource.ldf <destination folder> /Y
```

2. You can use your file-based backup utilities such as NTBackup, IBM Tivoli Storage Manager, Symantec BackupExec, etc.

Restoring the Resource database

It is important to document the location of your **master** database as part of your disaster recovery process. In previous versions of SQL Server, all we need to do to restore the server instance is to worry about the **master** database.

After a SQL Server 2005 instance has been rebuilt a restore of the **master** database will be done, the **Resource** database files should go along with it should a **WITH MOVE** option be required. This means that if the old location of the **master** database will be different from the one after the restore, the **Resource** database files should already be there prior to restoring the **master** database. This is very critical if a hardware failure occurred and you need to move the system databases on a different drive during the server instance rebuild.

To restore the **Resource** database, just copy the database files to the location of the **master** database files. If you have an older version of the **Resource** database, it is important to re-apply any subsequent updates. This is why the recommended approach is to simply do a daily backup of these files.

Case study 4: How to recover the database without having ldf file.

While trying to attach a Database without a logfile with the procedure **sp_attach_single_file_db** we get the following error message:

```
sp_attach_single_file_db 'test' 'C:\Program Files (x86)\Microsoft SQL  
Server\MSSQL.1\MSSQL\DATA\test.mdf'
```

The log cannot be rebuilt because the database was not cleanly shut down.
The database is not detached, before the logfile has been lost.

When we create a new Database with the same name and path, stop the sql server, replace with the original .mdf and start the sql server, the Database is visible but there are no items in the tree.

Resolution:

- 1)create a database with the same name in another directory as the one you're trying to attach
- 2)re-create all filegroups and files as necessary
- 3)shutdown the server
- 4)swap in the old mdf file and any ndf files
- 5)bring up the server and let the database attempt to be recovered and then go into suspect mode
- 6)put the database in emergency and single_user modes
- 7)run DBCC CHECKDB (dbname, REPAIR_ALLOW_DATA_LOSS) which will rebuild the log and run full repair
- 8) return database to online, multi_user mode

Note: sp_attach will only work for cleanly detached databases, if it doesn't work, you have to restore the db from a backup.

Case study 5: How to Recover Crashed SQL Server with existing data files and logs

Resolution:

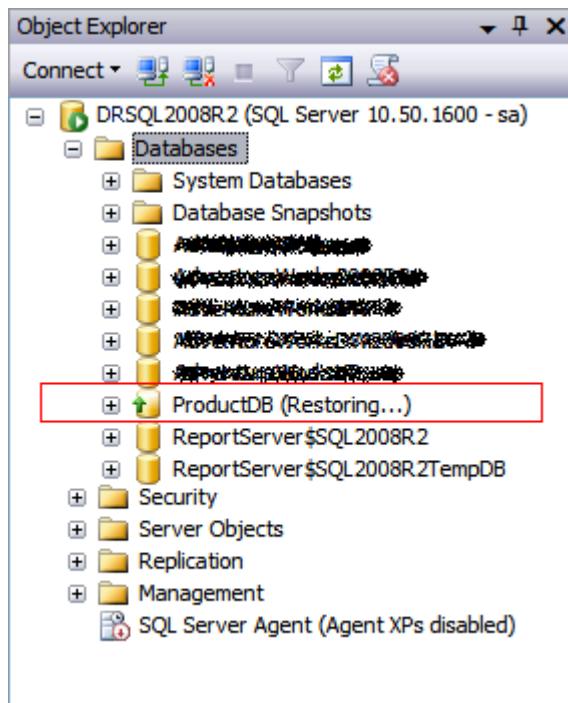
1. Before installation of SQL Server take the backup of existing Datafiles and Transaction logs.
2. Rename the datafiles and transaction logs for all system databases master,model, msdb and pubs to *file_name*_OLD.mdf and *file_name*_OLD.ldf.
3. Install the SQL Server.
4. Stop the services for SQL Server.
5. Rename the New datafiles and Transaction logs for all system databases to *file_name*_NEW.mdf and *file_name*_NEW.ldf.
6. Rename the Old datafiles and transaction logs for all system databases to Original name *file_name*.mdf and *file_name*.ldf.
7. Start the services for SQL Server.
8. Check all user and system Databases exists on the server and nothing is offline.
9. Verify the sql server and windows logins.

Case Study 6: How to Restart an Interrupted SQL Server Database Restore

The **RESTORE DATABASE...WITH RESTART** command is a very useful command which is available in SQL Server 2005 and higher versions. A Database Administrator can use this command to finish restoring an interrupted database restore operation.

In the below snippet you can see that ProductDB is in a (**Restoring...**) state once the SQL Server came online after the unexpected failure.

In the below snippet you can see that ProductDB is in a (**Restoring...**) state once the SQL Server came online after the unexpected failure.

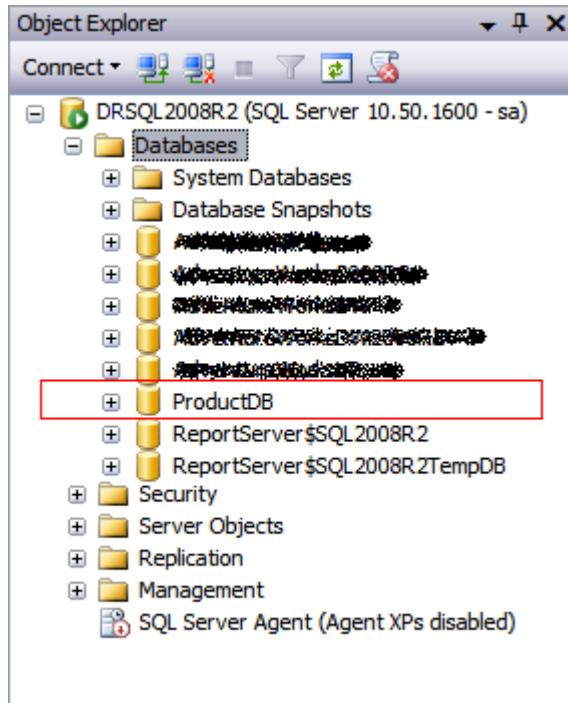


During such scenarios one can execute the **RESTORE DATABASE...WITH RESTART** command to successfully complete the database restore operation.

Below are two commands. The first gets a list of the backups on the file and the second does the actual restore with the restart option.

```
-- get backup information from backup file
RESTORE FILELISTONLY
FROM DISK ='C:\DBBackups\ProductDB.bak'
GO
-- restore the database
RESTORE DATABASE ProductDB
FROM DISK ='C:\DBBackups\ProductDB.bak'
WITH RESTART
GO
```

Below you can see that after running the **RESTORE DATABASE...WITH RESTART** command the database was successfully restored allowing user connectivity.



Case study 7: Point in Time Recovery Using New Timeline Feature

Point in time recovery allows you to recover your database to a particular point in time right before an erroneous transaction occurred. However, to be able to use Point in Time recovery for a database the database has to use the Full or Bulk Logged recovery model.

If your database is using the Simple recovery model you will need to change the recovery model to Full or Bulk-Logged and then start taking backups. To change the recovery model for a database you can execute the below TSQL code.

```
-- use this to change to Bulk Logged recovery model ALTER DATABASE databaseName SET RECOVERY BULK_LOGGED
```

```
-- or use this to change to Full recovery model ALTER DATABASE databaseName SET RECOVERY FULL
```

Example

Let's go through an example to demonstrate this enhancement for a database restore using SQL Server Management Studio in SQL Server 2012. Before I started, I checked that the AdventureWorks2008R2 recovery model was set to Full. In this example, first we will take a full

backup of the AdventureWorks2008R2 database, then we will drop the ErrorLog table and finally we will take a transaction log backup.

To achieve Point in Time Recovery for a database the DBA must have an idea when the problematic query was executed. In this demo the problematic query of dropping the ErrorLog table is shown in the table below for reference.

Activity	TSQL Code	Activity Time
Take full backup of AdventureWorks2008R2 database	BACKUP DATABASE ADVENTUREWORKS2008R2 TO DISK='C:\Backups\ADVENTUREWORKS2008R2.bak' GO	@ 6:10:00 AM
Drop [ErrorLog] table in AdventureWorks2008R2 database	USE AdventureWorks2008R2 GO DROP TABLE [dbo].[ErrorLog] GO	@ 6:12:11 AM
Perform a Transactional log backup	BACKUP LOG ADVENTUREWORKS2008R2 TO DISK='C:\Backups\ADVENTUREWORKS2008R2.BAK' WITH STATS = 10 GO	@ 6:14:00 AM

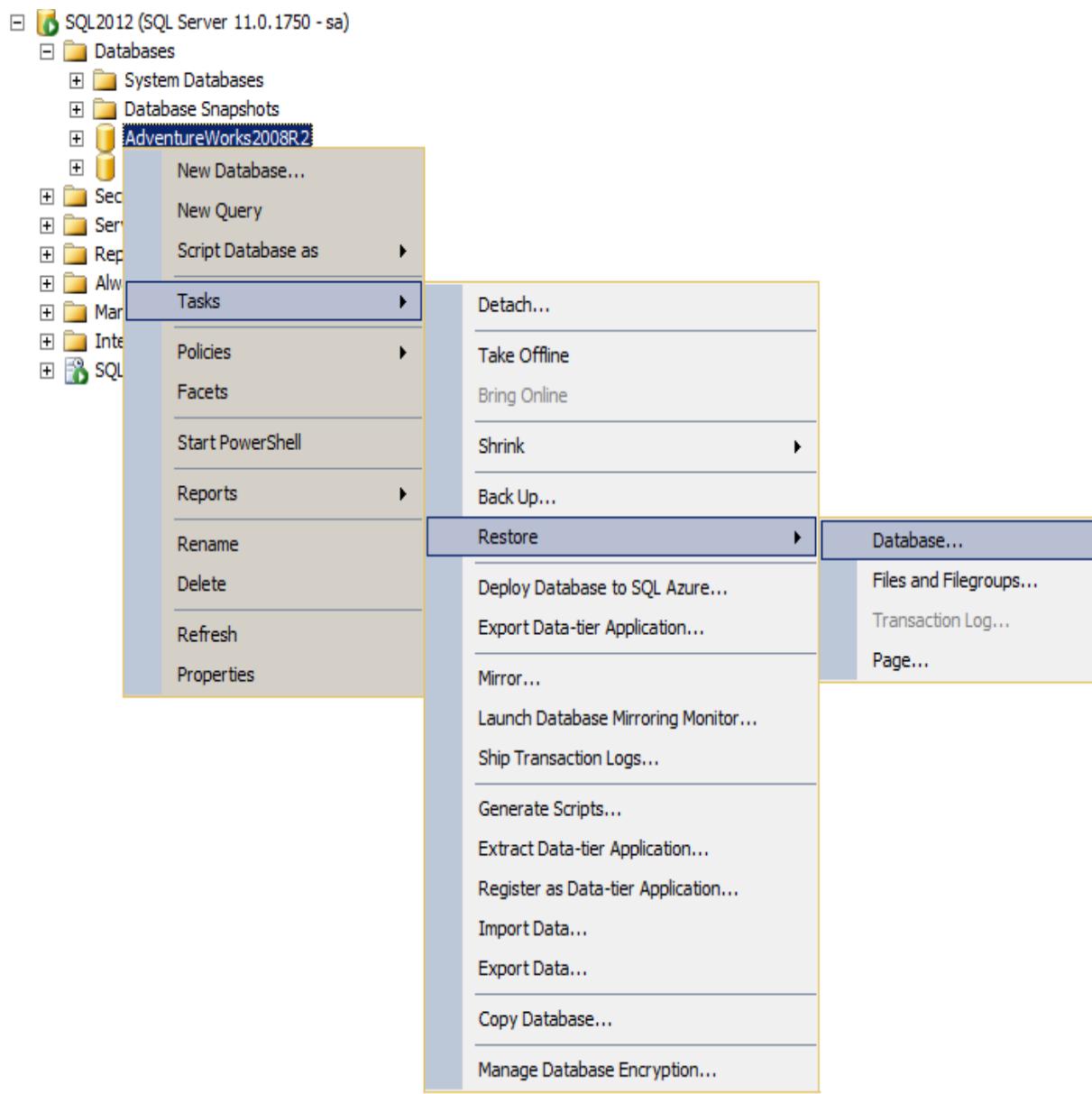
Here are the commands that were used.

```
/* Take full backup of AdventureWorks2008R2 database @ 6:10:00 AM */  
BACKUP DATABASE ADVENTUREWORKS2008R2  
TO DISK='C:\Backups\ADVENTUREWORKS2008R2.bak'  
  
/* Drop [ErrorLog] table in AdventureWorks2008R2 database @ 6:12:11 AM */  
USE AdventureWorks2008R2  
GO  
DROP TABLE [dbo].[ErrorLog]  
GO  
  
/* Perform a Transactional log backup @ 6:14:00 */  
BACKUP LOG ADVENTUREWORKS2008R2  
TO DISK='C:\Backups\ADVENTUREWORKS2008R2.BAK'  
WITH STATS = 10  
GO
```

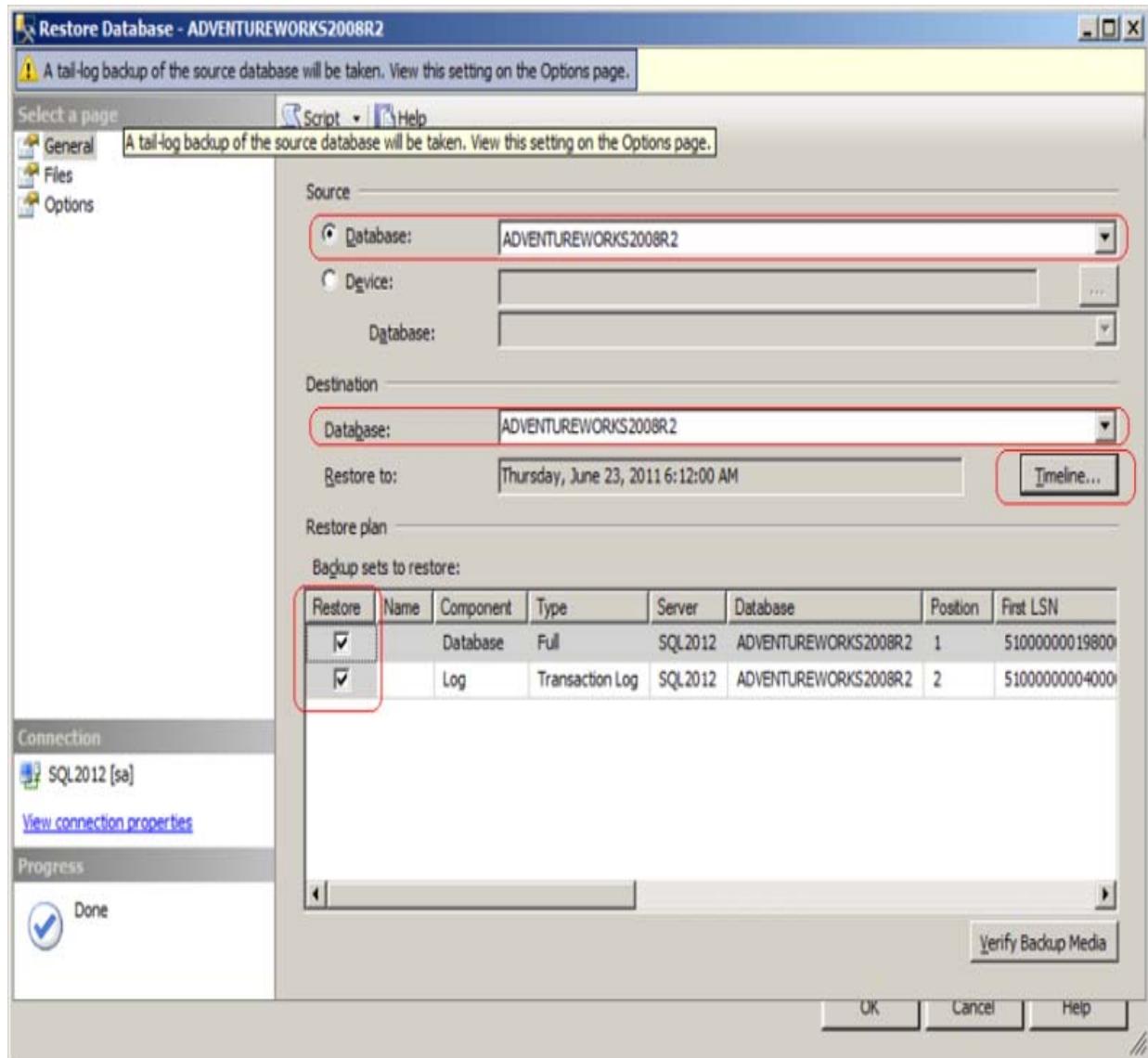
Steps to restore a database using Timeline in SQL Server 2012

Now that we have our backups in place we will walk through the restore process.

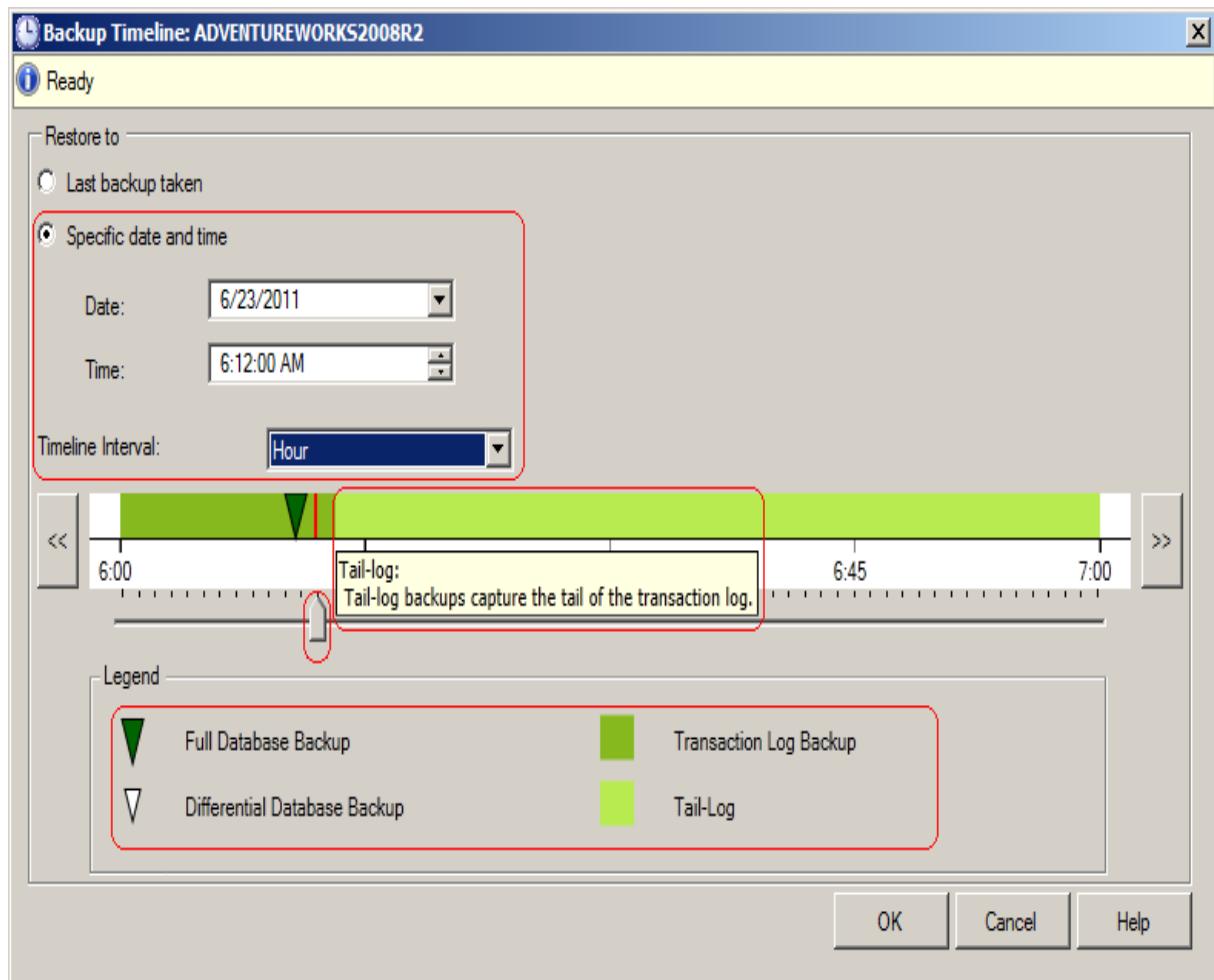
1. In Object Explorer expand Databases and then right click AdventureWorks2008R2 database and choose Tasks > Restore > Database... as shown in the snippet below to open up database restore dialog box.



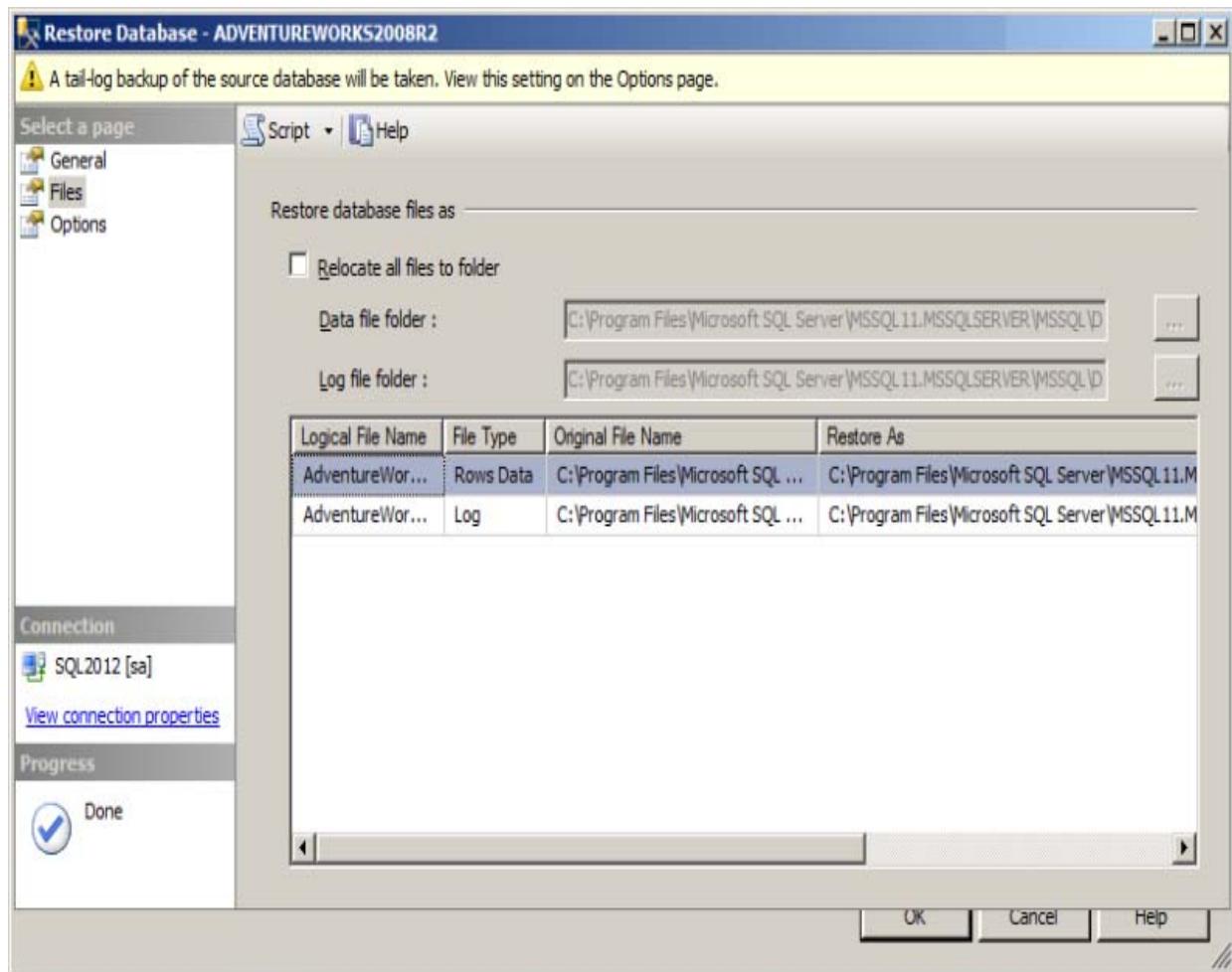
2. In the General Page of Restore Database Dialog Box you need to specify the Source Database and Destination Database names. Next, you need to select all the check boxes under Restore Plan Backup sets to restore as shown in the snippet below.



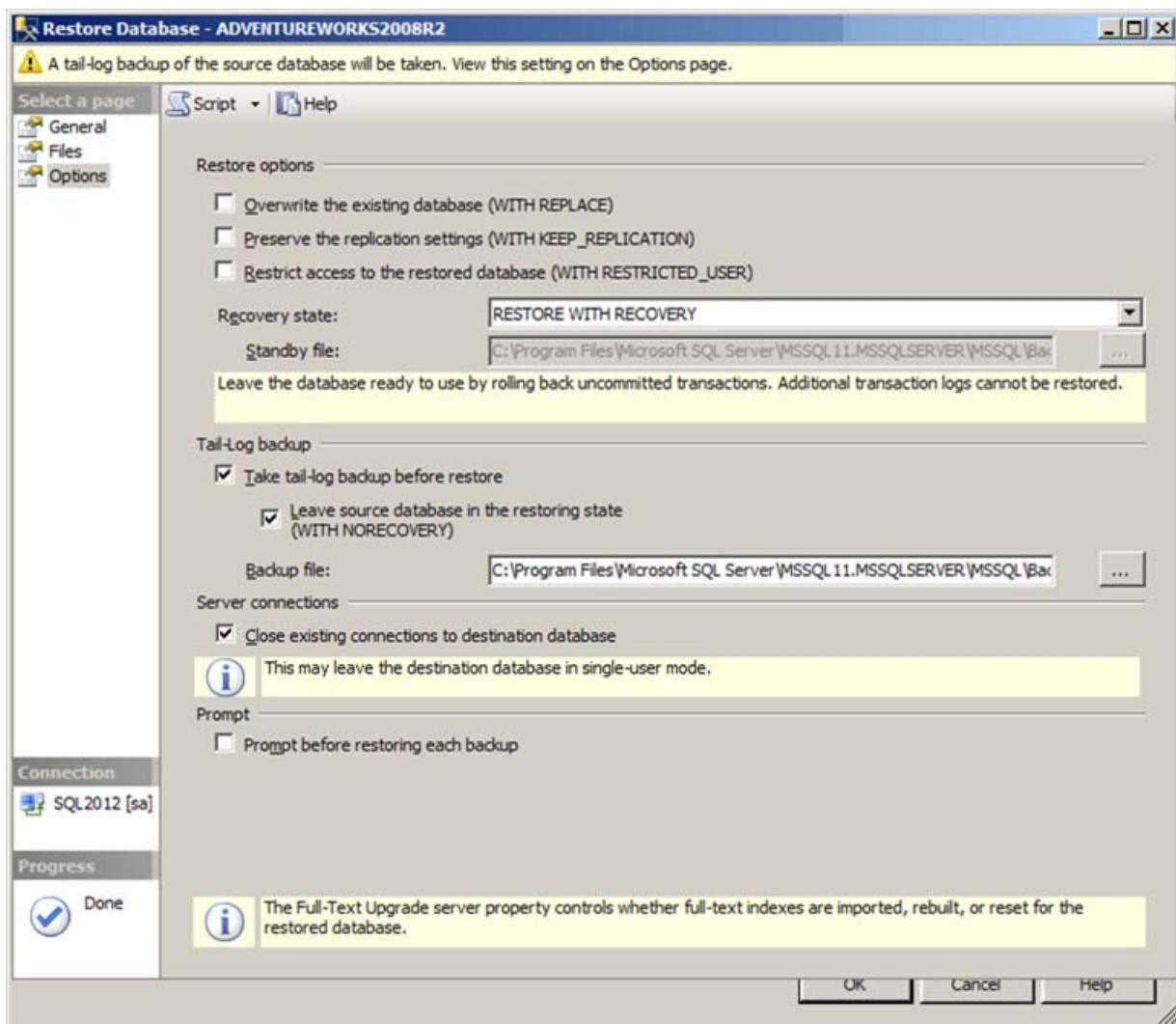
3. In order to restore a database to achieve point in time recovery click the Timeline... button next to the Restore to: option in the above snippet.
4. In the Backup Timeline: AdventureWorks2008R2 dialog box you need to choose the Point in Time to stop the recovery of the database as shown in the snippet below. In this demo, we know that the ErrorLog table was accidentally dropped after 6:12 AM. Hence, we know that the database should be restored to a point just before the disaster struck.



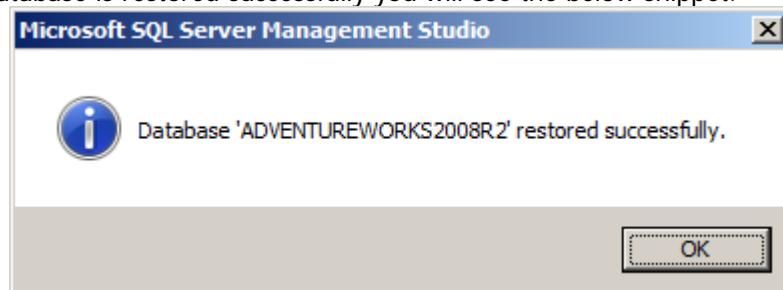
5. In the above snippet, you could see that to choose the desired point in time recovery for the database you can drag the arrow to the specific time as per your need. Once you have specified the point in time for the recovery click OK to save the settings and to get back to the Restore Database Dialog Box's General Page.
6. In the Files Page you can change the path where the database files should reside after the restore.



7. In the Options Page, you will see different Restore options. You need to choose the Recovery State as RESTORE WITH RECOVERY and select the check box Tail-Log backup and Leave source database in the restoring state with norecovery and specify the Backup file path for the tail log backup of the source database. Click OK to achieve point in time recovery of the database.. Click OK to achieve point in time recovery of the database.



8. Once the database is restored successfully you will see the below snippet.



The most important thing which a database administrator needs to find out is what is the recovery point time. In some scenarios you may know the time, but in other cases you may not know

Best Practices on High Availability, Backup & Restore

General High Availability:

1. Physically protect your SQL Servers from unauthorized users.
2. Physically document all of your SQL Server instances. Incorporate effective change management.
3. Always use a RAIDed array or SAN for storing your data.
4. Use SQL Server clustering, database mirroring, or log shipping to provide extra fault tolerance.
5. Replication is not an effective means to protect your data.
6. Always use server-class hardware, and standardize on the same hardware as much as possible.
7. Use hardware and software monitoring tools so you can quickly become aware of when problems first arise.
8. After testing, apply all new service packs and hot fixes to the OS and SQL Server.
9. Cross-train staff so that there are multiple people who are able to deal with virtually any problem or issue.

Disaster Recovery

1. You must create a disaster recovery plan and include every detail you will need to rebuild your servers.
2. As your SQL Servers change over time, don't forget to update your disaster recovery plan.
3. Write the disaster recovery plan so that any computer literate person will be able to read and follow it. Do not assume a DBA will be rebuilding the servers.
4. Fully test your disaster recovery plan at least once a year.

Backup

1. All production databases should be set to use the full recovery model. This way, you can create transaction log backups on a periodic basis.
2. Whenever possible, perform a daily full backup of all system and user databases.
3. For all production databases, perform regular transaction log backups, at least once an hour.
4. Perform full backups during periods of low user activity in order to minimize the impact of backups on users.
5. Periodically test backups to ensure that they are good and can be restored.
6. Backup first to disk, then move to tape or some other form of backup media.
7. Store backups offsite.

8. If using SQL Server 2005 encryption, be sure to backup the service master key, database master keys, and certificates.
9. If you find that backup times take longer than your backup window, or if backup file sizes are taking up too much space on your storage device, consider a third party backup program, such as SQL Backup Pro.
10. Document, step-by-step, the process to restore system and user databases onto the same, or a different server. You don't want to be looking this information up during an emergency.

Chapter – 16

Log- Shipping

What is Log Shipping

Log Shipping is one of the methods for creating a Standby server, by god forbiddeen if something happens to our production server we need a standby server and Microsoft has come up with this idea and introduced Log Shipping in SQL 2000 itself. It is mainly used in OLTP environment (Online Transaction Processing). It is used as a High Availability Solution and also in Disaster recovery situations.

Overview of Log Shipping

Let's discuss the overview of log shipping. The basic idea is nothing but backup and restore of the database and transaction logs sequentially from the primary server to secondary or standby server. We might be having some extremely critical databases in our production environment and need them to be online 24*7. If the production server is down due to some natural disasters or in some cases we need to reboot the server after applying some service packs or in case of some hardware upgradation i.e adding extra hard disks, all those require some downtime of the production server in that case we can make use of log shipping.

Pre-Requisites

The following are the prerequisites to configure Log Shipping SQL 2012

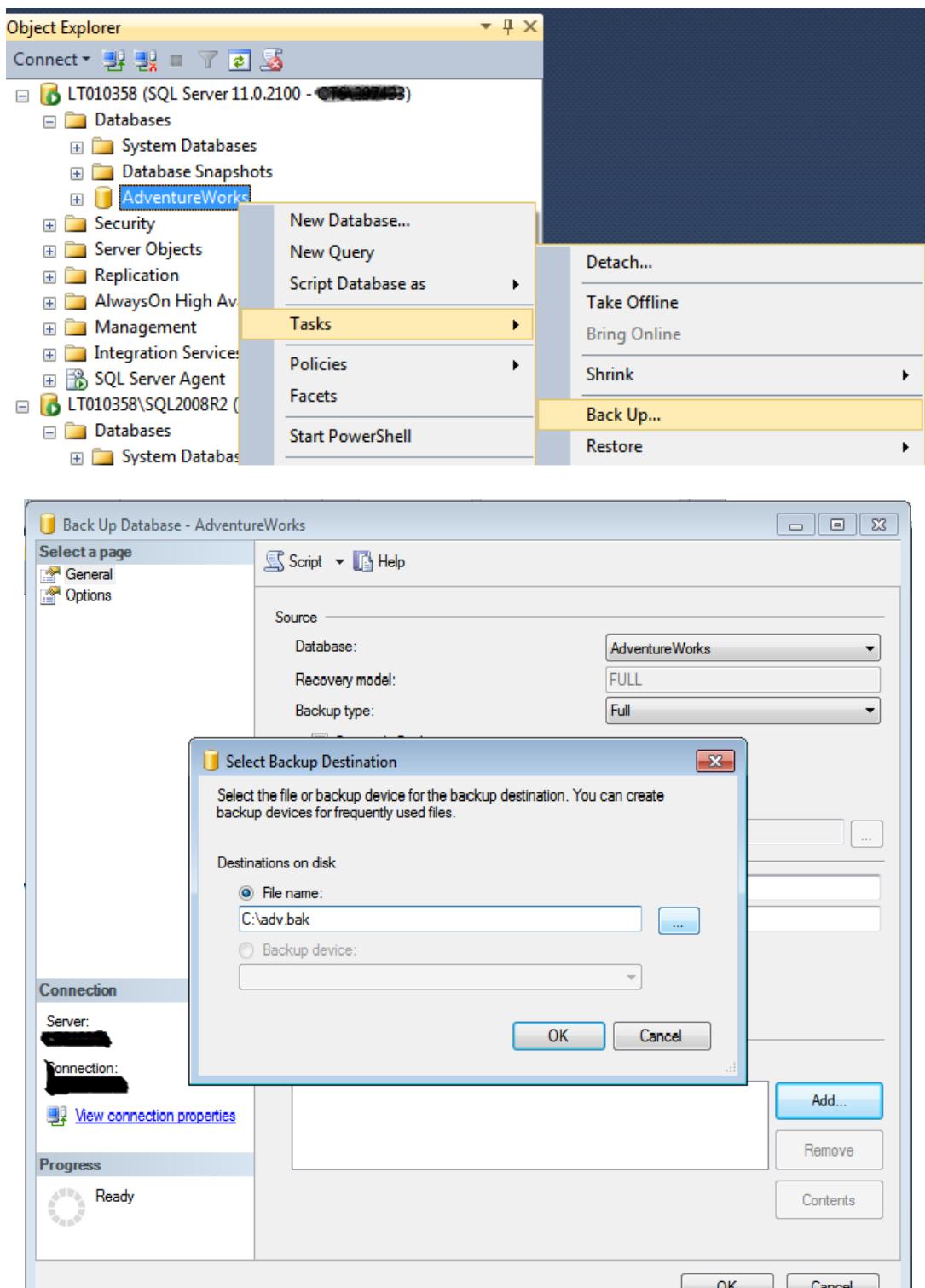
1. SQL Server 2012 Enterprise Edition, SQL Server 2012 web Edition, or SQL Server 2012 Standard Edition must be installed on all server instances involved in log shipping.
2. The servers involved in log shipping should have the same case sensitivity settings.
3. The SQL Services in both the primary and secondary server should be the same with same password. Preferably --a domain account.
4. The DB to be log Shipped should be in Full Recovery or Bulk logged recovery model, so that T-logs can be applied, else you cannot configure log shipping. Use the below command to change the recovery model or else you can change in the SSMS by right clicking the DB properties.
5. Shared folder should be created in Primary server and grant read permissions to secondary server service account

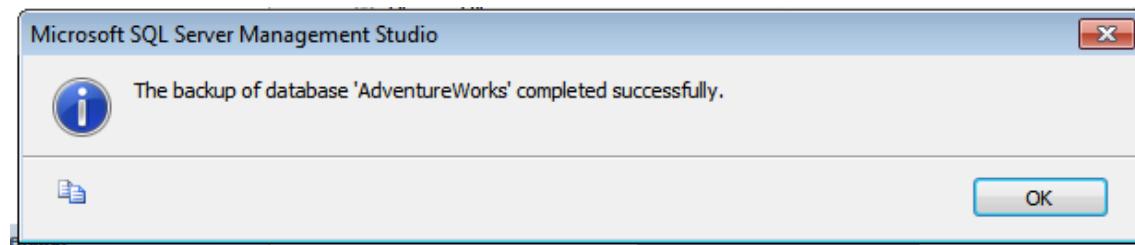
Initialization:

Take backup in 2012 default instance and restore in 2012 another named instance with standby mode--with step by step screen shots.

BACKUP:

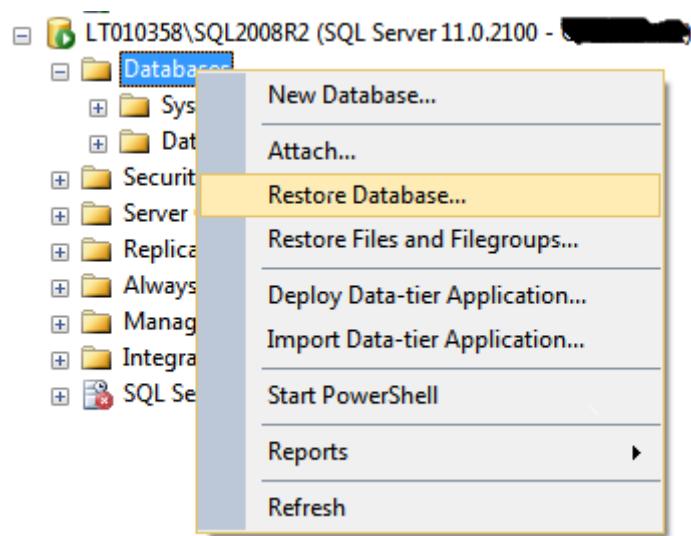
Go the primary server → goto the particular database → tasks → select the database → backup type as Full → add the backup location path → click ok



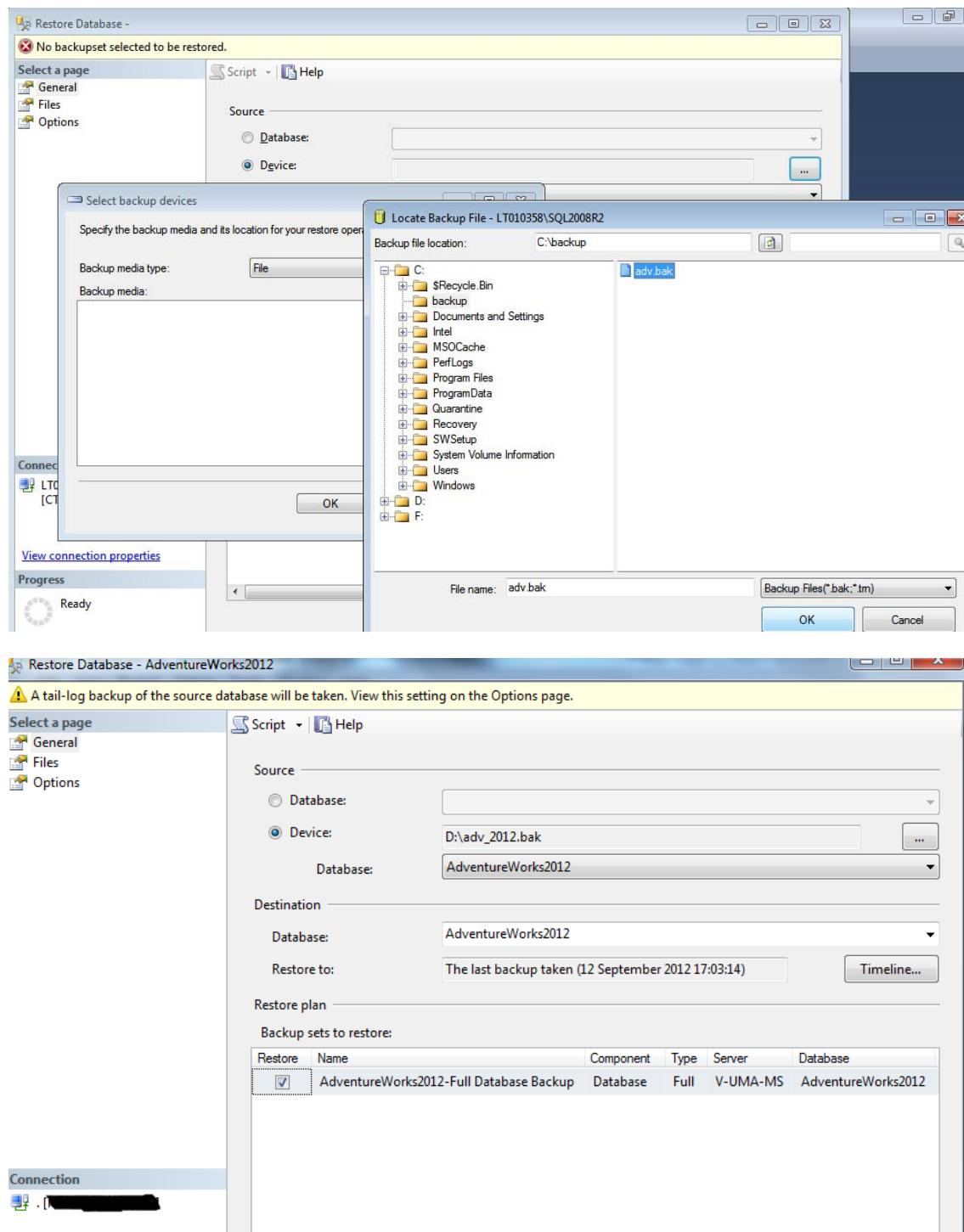


RESTORATION:

Go to the server → select databases → select restore database option → give the database name in General tab → select the from databases → give the backup file path

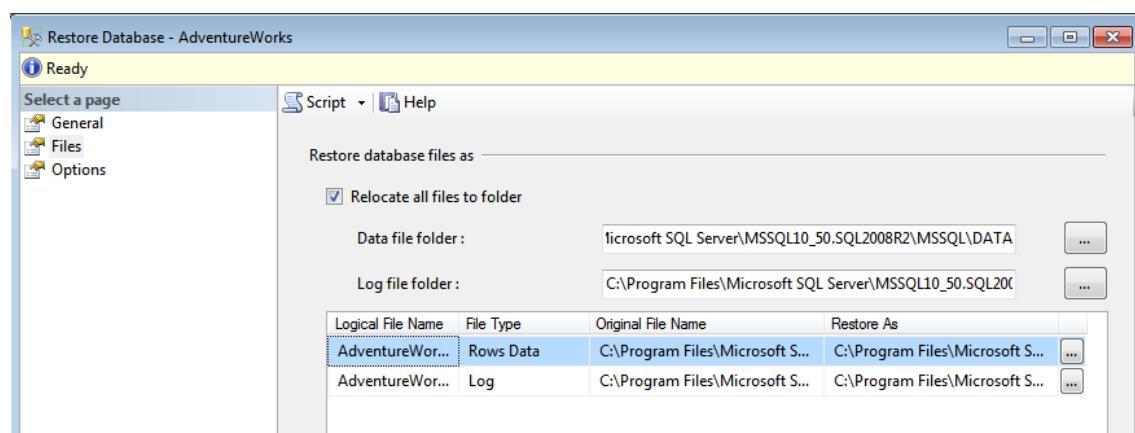
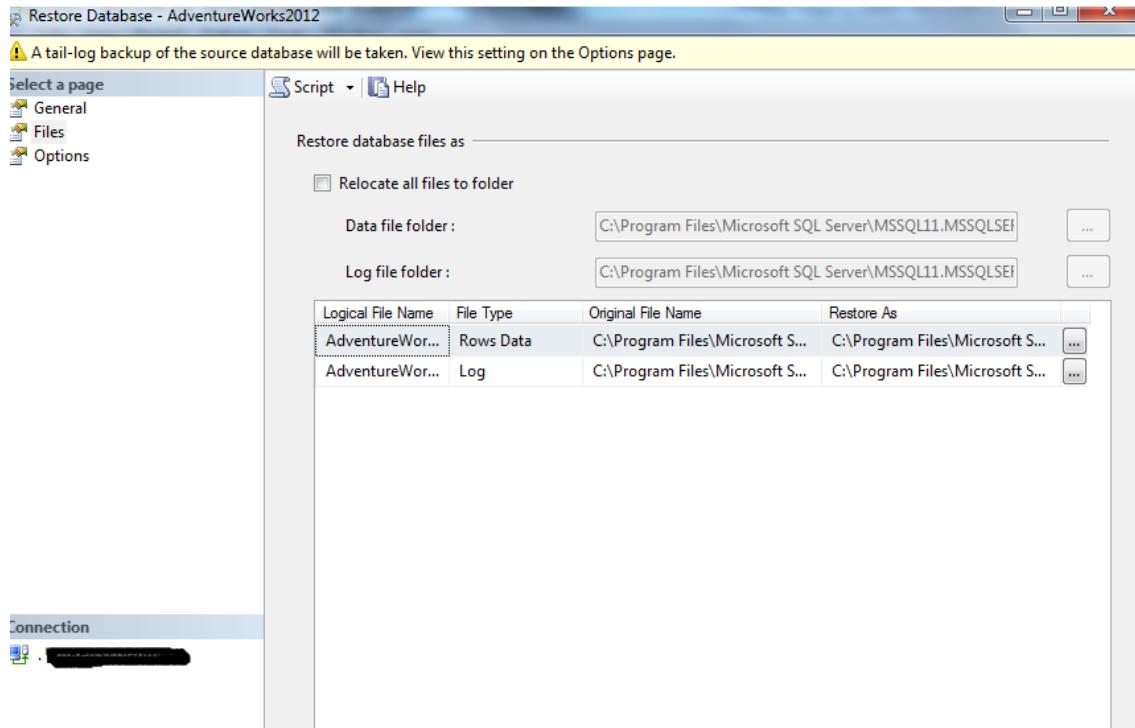


Select Device option → locate the backup location path →

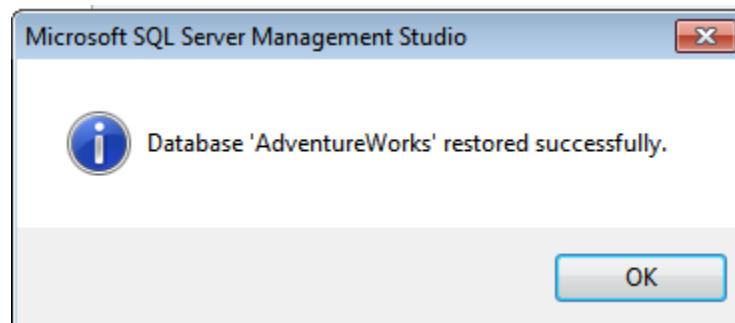
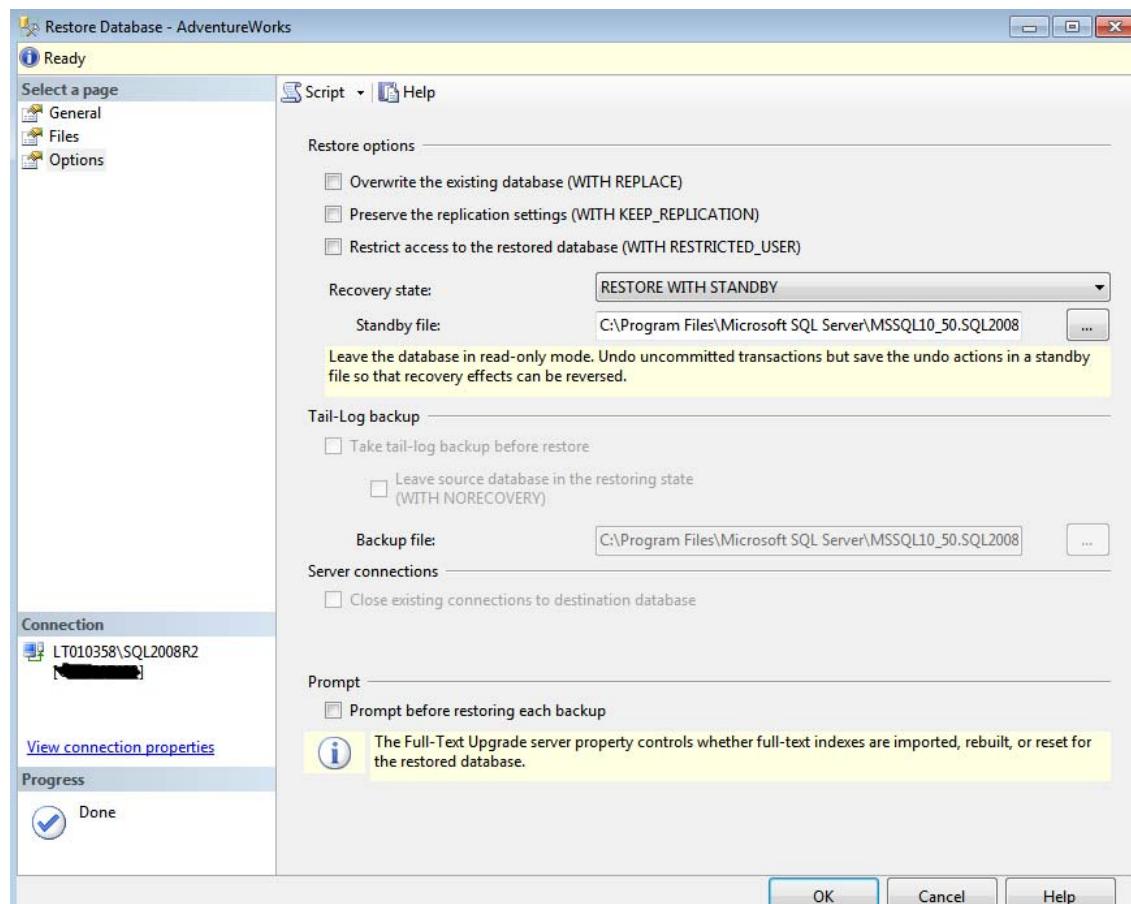


Goto files pane → give the correct data and log file path according to the that server settings
→ and click ok

Go to → option pane select the recovery option(stand by)—>click ok

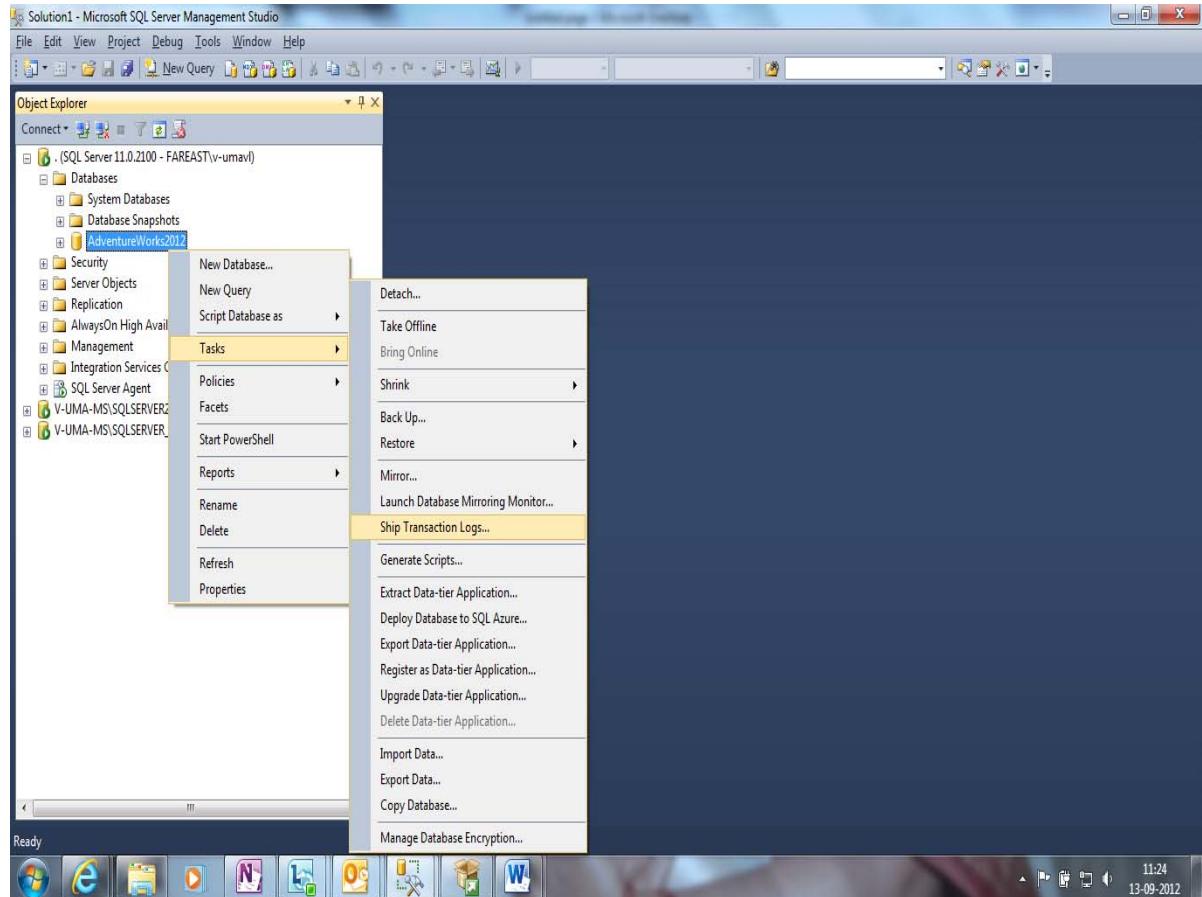


Make sure that you are un-checking the tail-log checkbox.

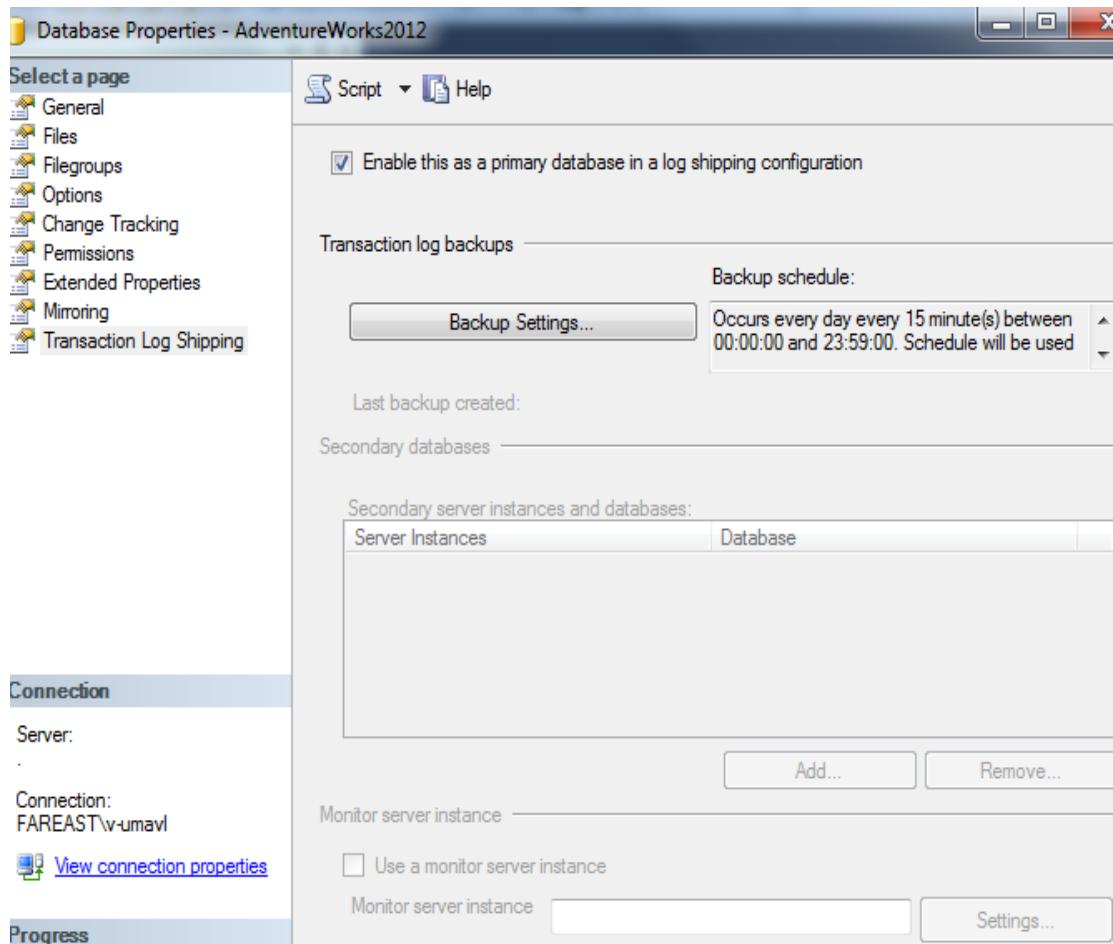


Deployment:

To the primary server → select the Adventure works database (which needs to be log shipped)
→ enable → tasks → select ship transactional logs option → click ok



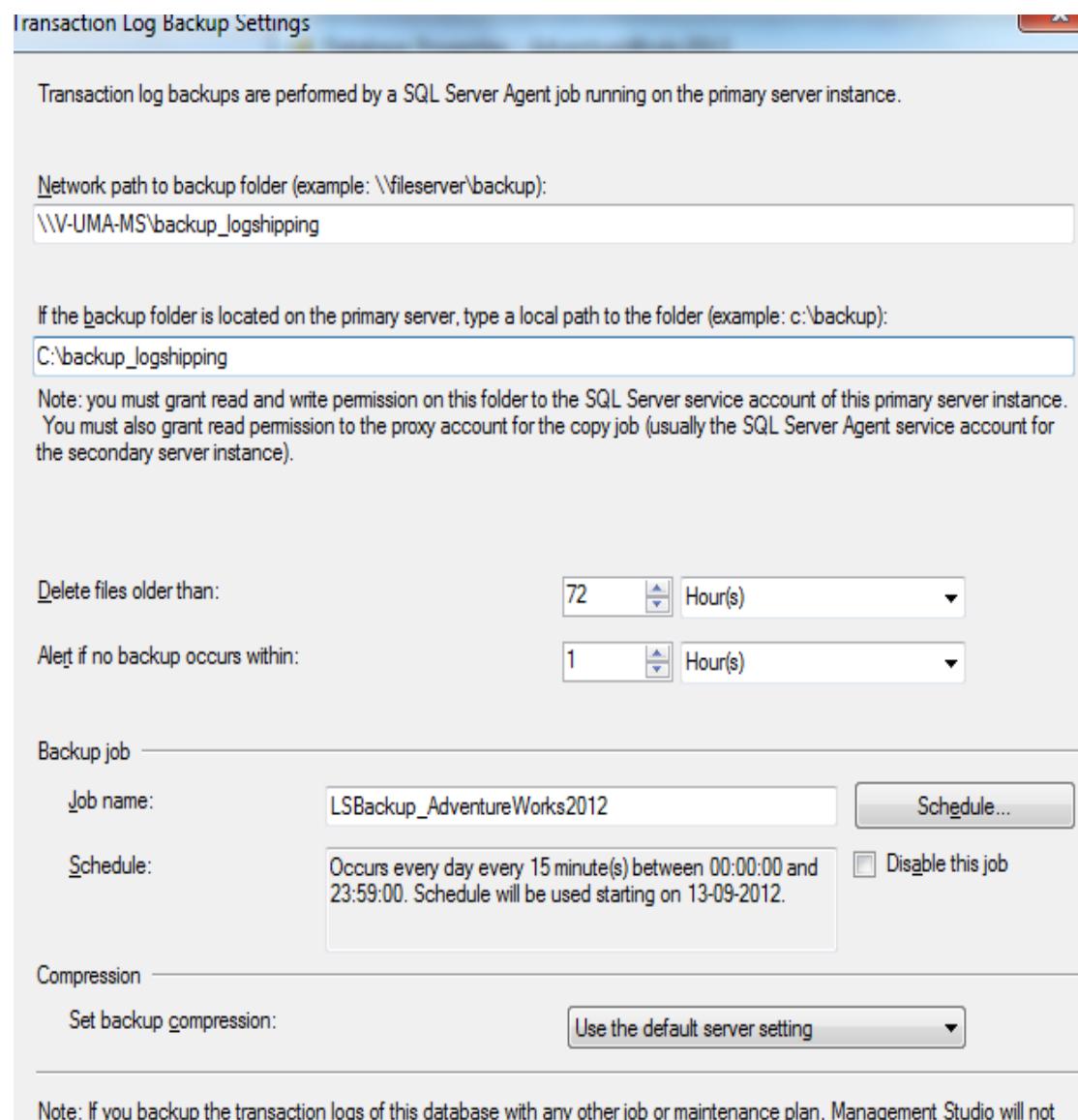
Enabled the option –Go to backup setting options



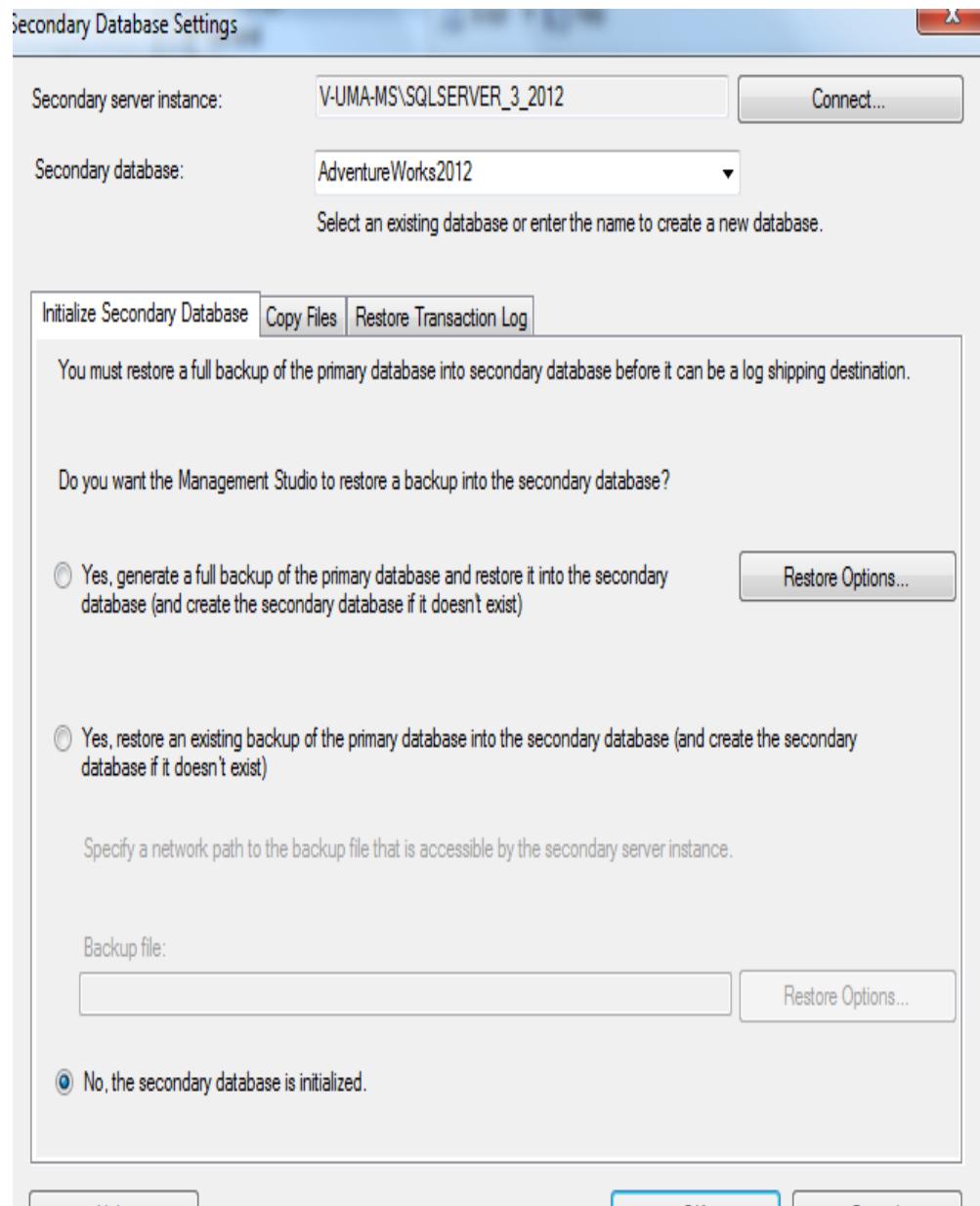
And give the network path for backup folder on primary→

And give the local path for same file→

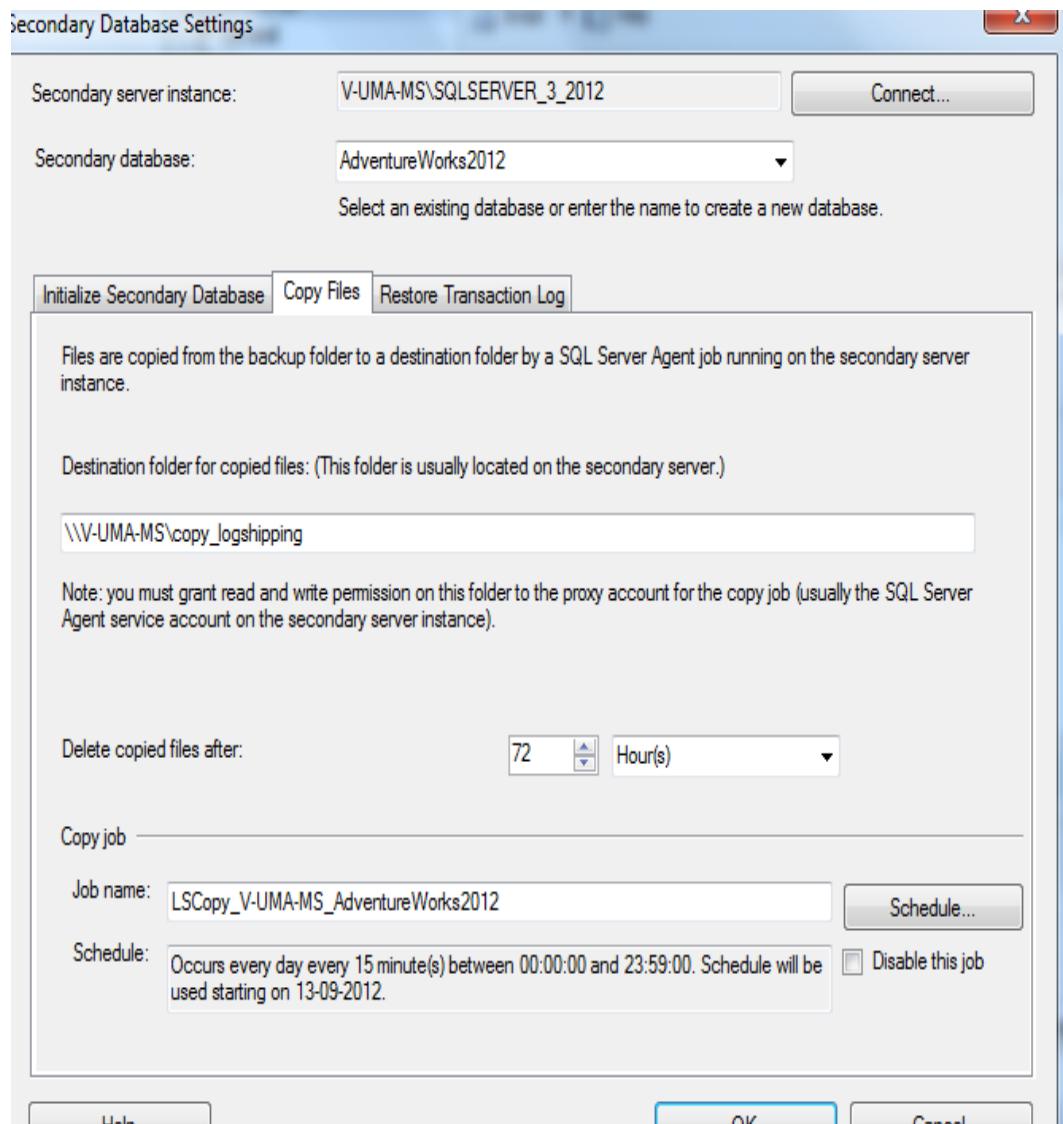
If you want to change the schedule for this backup job then go to schedule set the thing according to your requirement --> click ok



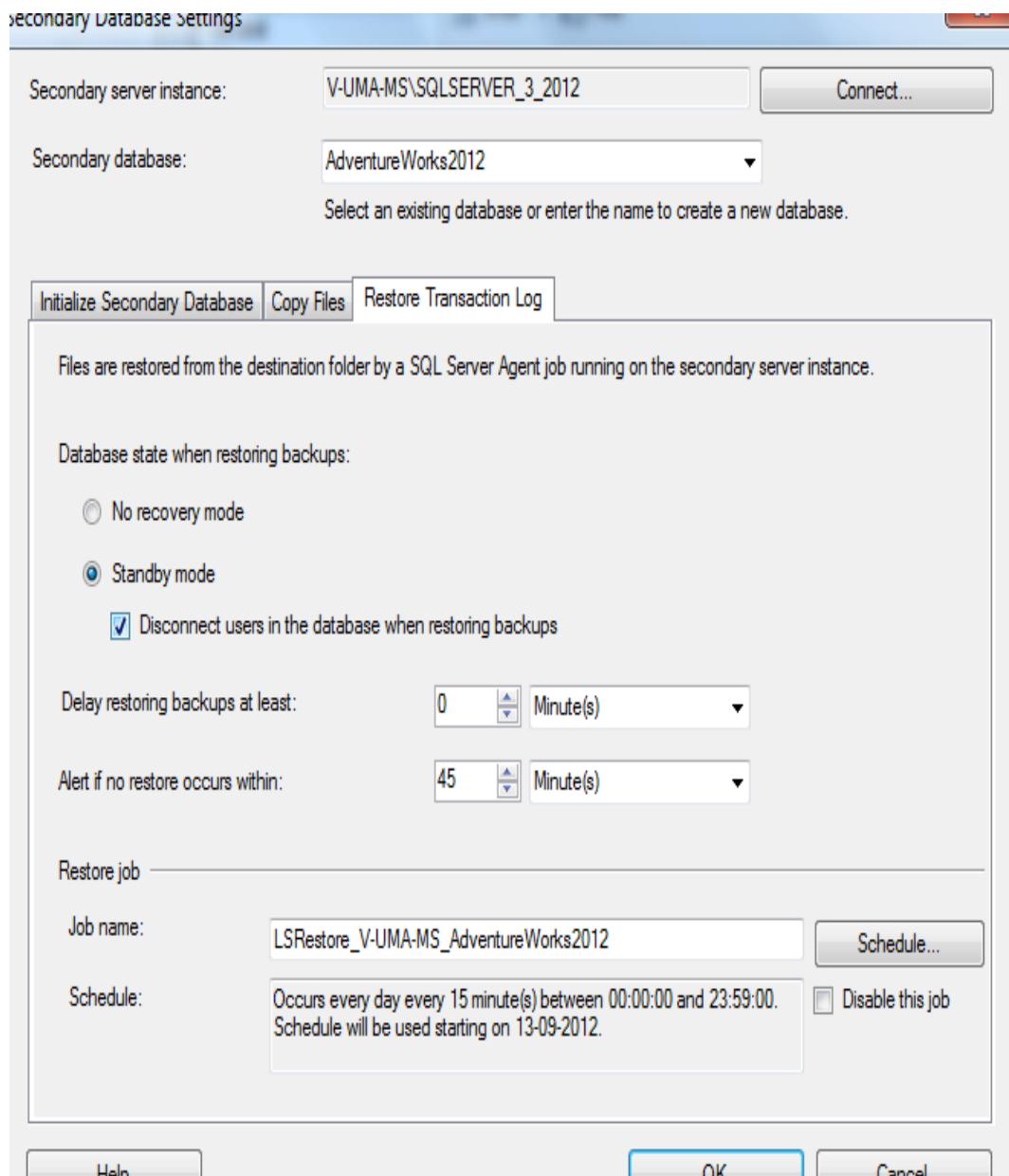
Click –add secondary database → connect the secondary server →



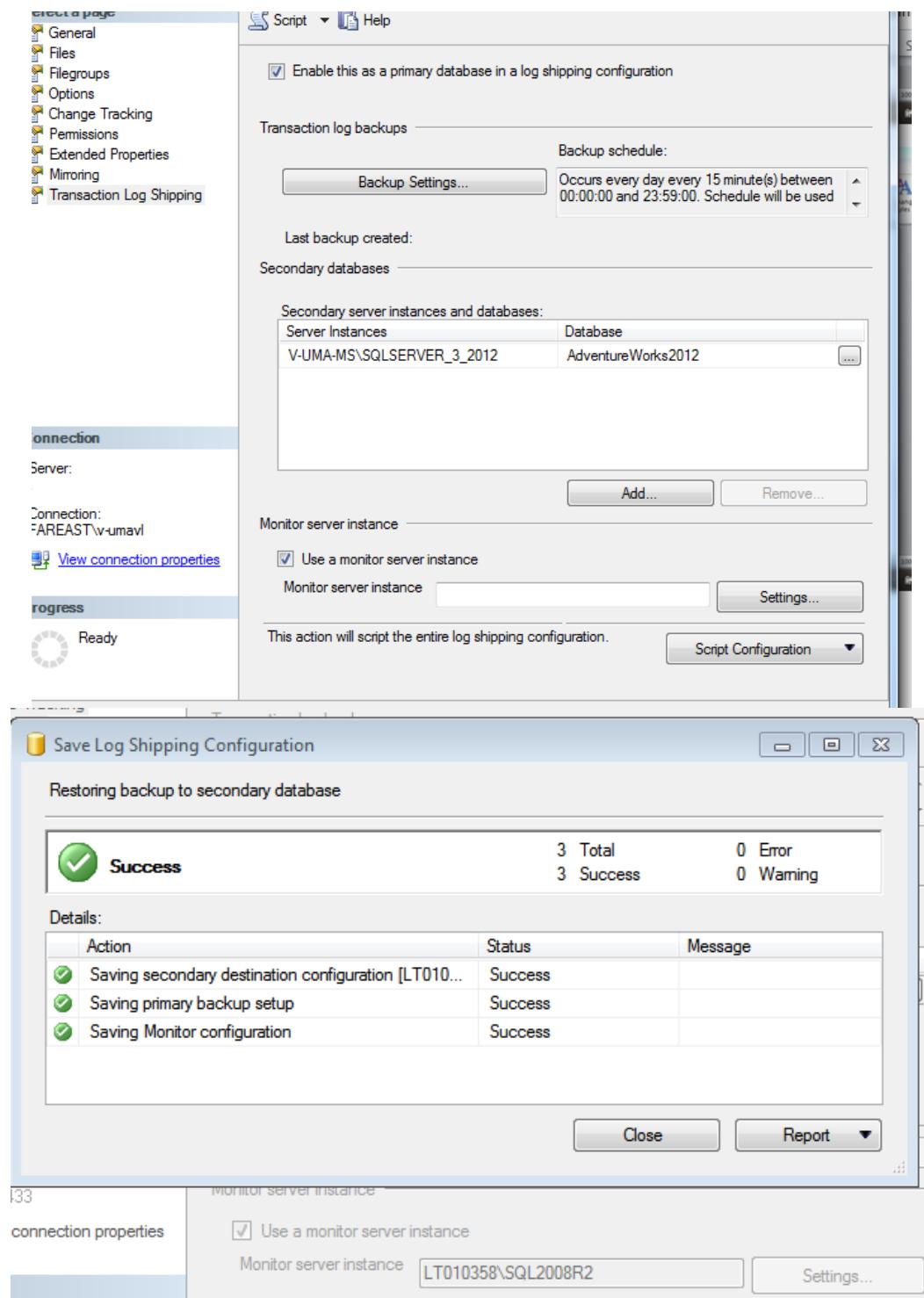
Go to the second option copy files → give the network path for the copy folder which is in secondary server → if you want to change the schedule for this copy job then select schedule tab and change settings according to your specification



Goto the last option Restore the transaction log select the secondary database is in Stand by then select the second stand by → select the disconnect thy users option else select the first option which is no recovery mode .If you want to change the schedule for the restore job then select the schedule and change accordingly your requirement → click ok



If monitor is there then select the use monitor option and →connect the monitor server—Click ok



Failover in SQL 2012 Log Shipping

The most important aspect in Log Shipping is Failover. Lets discuss it detail ! ! ! If the primary server in Log Shipping becomes unavailable or if it is going to be down due to some manual intervention, DBA should immediately perform the following steps for failover.

Step 1: Try to backup the Tail end of the transaction log in primary server with NORECOVERY option i.e perform a tran log backup if the primary is still accessible.

Backup log DBName to disk = "Local path or Network path" with NORECOVERY ---> A

Else execute the below T-SQL in secondary server to bring the secondary online,

Restore database DBName with Recovery ---> C

Step 2: If you were able to perform ---> A in step 1 then proceed with ---> B in step 2 to bring the secondary db online from read-only state. If you were able to perform only ---> C in step1 then go to step 3

Restore log DBName from disk = "Local path or Network path" with RECOVERY ---> B

Step 3: The syslogins and sysusers table in primary and secondary server should be in sync otherwise the DB users and other application users from primary will not be able to login into SQL server or into the DB in secondary server after failover occurs.

Two ways are there to accomplish the above task namely

* Create the highly critical application users logins in the secondary server similar to primary just before configuring log shipping. Use the below sps to resolve orphaned users

```
USE
GO
sp_change_users_login @Action='update_one', @UserNamePattern='', @LoginName=''
GO
```

Script to find orphahend users

```
exec sp_change_users_login 'Report'
```

Use Master

```
SET NOCOUNT ON
```

```
SELECT 'EXEC sp_addlogin @loginame = "' + loginname + """
      , @defdb = "' + dbname + """
      , @deflanguage = "' + language + """
      , @encryptopt = "skip_encryption"""
      , @passwd =' 
      , cast(password AS varbinary(256))
```

```

', @sid =
, sid
FROM sys.syslogins
where loginname Not like 'NA%'      and
      loginname not like 'Builtin%' and loginname Not like 'sa'

```

Run the above script on Source server copy the result and execute on Destination server

Eg:-

```

EXEC sp_addlogin @loginame = 'CorpCommUser'      , @defdb = 'CorpComm',
@deflanguage = 'us_english' , @encryptopt = 'skip_encryption' , @passwd =
0x01003F04413C64CEE4767BA2DD0053A02C6056640C4C88C24DFA , @sid =
0xCEE1766A76520E43A98DCB141B031F7E

```

Step 4: Also Disable the log shipping jobs in the primary and secondary servers,once failover occurs.

Step 5: Once the failover occurs the original secondary server is configured as primary and log shipping is again newly configured from this new primary server(original secondary) to original primary(now secondary).

Step 6: When you once again want to revert to the original state of log shipping i.e original primary was primary and original secondary was secondary, you need take a full backup in new primary server(original secondary) and restore it in original primary and reconfigure the log shipping from the original primary to original secondary.

Frequently Raised Errors In Log-Shipping

Question : IS it possible to log ship database between SQL 2000 & SQL 2008?

Answer: No, thats impossible, In SQL 2008 transaction log architecture is changed compared to SQL 2000 and hence you won't be able to restore tlog backups from SQL 2000 to SQL 2008 or vice versa.

Question:I'm getting the below error message in restoration job on secondary server, WHY?

[Microsoft SQL-DMO (ODBC SQLState: 42000)]

Error 4305: [Microsoft][ODBC SQL Server Driver][SQL Server]The log in this backup set begins at LSN 7000000026200001, which is too late to apply to the database. An earlier log backup that includes LSN 6000000015100001 can be restored.

[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE LOG is terminating abnormally.

Answer: Was your sql server or agent restarted Y'day in either source or destination ? because the error states there is a mismatch in LSN. A particular tran log was not applied in the destination server hence the subsequent tran logs cannot be applied as a result !

You can check log shipping monitor \ log shipping tables to check the which transaction log is last applied to secondary db, if the next consecutive transaction logs are available in the secondary server share folder you manually RESTORE the logs with NORECOVERY option, Once you restored all the logs automatically from the next cycle the job will work fine.

Incase if you are not able to find the next transaction log in secondary server shared folder, you need to reconfigure log shipping. Try the below tasks to re-establish log shipping again.

- Disable all the log shipping jobs in source and destination servers
- Take a full backup in source and restore it in secondary server using the With Standby option
- Enable all the jobs you disabled previously in step1

Question: Is it possible load balance in log shipping?

Answer: Yes of course it's possible in log shipping, while configuring log shipping you have the option to choose standby or no recovery mode, and there you select STANDBY option to make the secondary database read-only.

Question: Can I take full backup of the log shipped database in primary server??

Answer: In SQL Server 2000 you won't be able to take full backup of log shipped database, because this will break the LSN chain and it directly affects the log shipping. In SQL Server 2008, yes its possible. You can take full backup of log shipped database and this won't affect the log shipping.

Question : Can I shrink log shipped database log file??

Answer: Yes of course you can shrink the log file, but you shouldn't use WITH TRUNCATE option. If you use this option obviously log shipping will be disturbed.

Question : Can I take full backup of the log shipped database in secondary server??

Answer: No chance, you won't be able to execute BACKUP command against a log shipped database in secondary server.

Question: I've configured Log shipping successfully on standby mode, but in the restoration job I'm getting the below error. What I do to avoid this in future??

Message

2006-07-31 09:40:54.33 *** Error: Could not apply log backup file 'C:\Program Files\Microsoft

SQL Server\MSSQL.1\MSSQL\Backup\LogShip\TEST_20060731131501.trn' to secondary

database 'TEST'.(Microsoft.SqlServer.Management.LogShipping) ***

2006-07-31 09:40:54.33 *** Error: Exclusive access could not be obtained because the
database is in use.

RESTORE LOG is terminating abnormally.(.Net SqlClient Data Provider) ***

Answer: To restore transaction logs to the secondary db, SQL Server needs exclusive access on the database. When you configure it in standby mode, users will be able to access the database and runs query against the secondary db. Hence If the scheduled restore jobs runs at that time, the db will have a lock and it won't allow SQL Server to restore the tlogs. To avoid this you need to check "Disconnect users in the database when restoring backups" options in log shipping configuration wizard.

Question : Suddenly I'm getting the error below, How can I rectify this???

[Microsoft SQL-DMO (ODBC SQLState: 42000)] Error 4323: [Microsoft][ODBC SQL Server Driver][SQL Server]The database is marked suspect. Transaction logs cannot be restored. Use RESTORE DATABASE to recover the database.

[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE LOG is terminating abnormally

Answer : We had the same issue some time ago, this was related to a new file being created in a filegroup on the source. Don't know if this applies to your case, but restoring a backup of this new file on the secondary server solved the problem.

Question : Is it possible to log ship database from SQL server 2005 to SQL server 2008 and vice versa?

Answer : Yes you can log ship database from SQL server 2005 to SQL Server 2008 this will work. However log shipping from SQL Server 2008 to SQL Server 2005 is not possible because you won't be able to restore SQL server 2008 backup to SQL Server 2005 (downgrading version)

Error message 14420 and error message 14421 that occur when you use log shipping:

Error message 14420 Error: 14420, Severity: 16, State: 1

The log shipping destination %s.%s is out of sync by %s minutes.

Error message 14421 Error: 14421, Severity: 16, State: 1

The log shipping destination %s.%s is out of sync by %s minutes.

If you are using SQL Server 2008, the description for these error messages are different:

Error message 14420 Error: 14420, Severity: 16, State: 1

The log shipping primary database %s.%s has backup threshold of %d minutes and has not performed a backup log operation for %d minutes. Check agent log and logshipping monitor information.

Error message 14421 Error: 14421, Severity: 16, State: 1

The log shipping secondary database %s.%s has restore threshold of %d minutes and is out of sync. No restore was performed for %d minutes. Restored latency is %d minutes. Check agent log and logshipping monitor information.

Log shipping uses Sqlmaint.exe to back up and to restore databases. When SQL Server creates a transaction log backup as part of a log shipping setup, Sqlmaint.exe connects to the monitor server and updates the **log_shipping_primaries** table with the **last_backup_filename** information. Similarly, when you run a Copy or a Restore job on a secondary server, Sqlmaint.exe connects to the monitor server and updates the **log_shipping_secondaries** table.

As part of log shipping, alert messages 14220 and 14221 are generated to track backup and restoration activity. The alert messages are generated depending on the value of **Backup Alert** threshold and **Out of Sync Alert** threshold respectively.

The alert message 14220 indicates that the difference between current time and the time indicated by the **last_backup_filename** value in the **log_shipping_primaries** table on the monitor server is greater than value that is set for the **Backup Alert** threshold.

The alert message 14221 indicates that the difference between the time indicated by the `last_backup_filename` in the `log_shipping_primaries` table and the `last_loaded_filename` in the `log_shipping_secondaries` table is greater than the value set for the **Out of Sync Alert** threshold.

Resolution:

Troubleshooting Error Message 14420

By definition, message 14420 does not necessarily indicate a problem with log shipping. The message indicates that the difference between the last backed up file and current time on the monitor server is greater than the time that is set for the **Backup Alert** threshold.

There are several reasons why the alert message is generated. The following list includes some of these reasons:

1. The date or time (or both) on the monitor server is different from the date or time on the primary server. It is also possible that the system date or time was modified on the monitor or the primary server. This may also generate alert messages.
2. When the monitor server is offline and then back online, the fields in the `log_shipping_primaries` table are not updated with the current values before the alert message job runs.
3. The log shipping Copy job that is run on the primary server might not connect to the monitor server `msdb` database to update the fields in the `log_shipping_primaries` table. This may be the result of an authentication problem between the monitor server and the primary server.
4. You may have set an incorrect value for the **Backup Alert** threshold. Ideally, you must set this value to at least three times the frequency of the backup job. If you change the frequency of the backup job after log shipping is configured and functional, you must update the value of the **Backup Alert** threshold accordingly.
5. The backup job on the primary server is failing. In this case, check the job history for the backup job to see a reason for the failure.

Troubleshooting Error Message 14421

By definition, message 14421 does not necessarily indicate a problem with Log Shipping. This message indicates that the difference between the last backed up file and last restored file is greater than the time selected for the **Out of Sync Alert** threshold.

There are several reasons why the alert message is raised. The following list includes some of these reasons:

1. The date or time (or both) on the primary server is modified such that the date or time on the primary server is significantly ahead between consecutive transaction log backups.
2. The log shipping Restore job that is running on the secondary server cannot connect to the monitor server `msdb` database to update the `log_shipping_secondaries` table with the correct value. This may be the result of an authentication problem between the secondary server and the monitor server.

3. You may have set an incorrect value for the **Out of Sync Alert** threshold. Ideally, you must set this value to at least three times the frequency of the slower of the Copy and Restore jobs. If the frequency of the Copy or Restore jobs is modified after log shipping is set up and functional, you must modify the value of the **Out of Sync Alert** threshold accordingly.
4. Problems either with the Backup job or Copy job are most likely to result in "out of sync" alert messages. If "out of sync" alert messages are raised and if there are no problems with the Backup or the Restore job, check the Copy job for potential problems. Additionally, network connectivity may cause the Copy job to fail.
5. It is also possible that the Restore job on the secondary server is failing. In this case, check the job history for the Restore job because it may indicate a reason for the failure.

Best Practices of Log-Shipping

- If you don't currently employ clustering or database mirroring for your SQL Servers because of cost, consider employing log shipping to help boost your high availability. It provides reasonably high availability at low cost.
- If you take advantage of SQL Server 2000 or 2005 log shipping capability, you will want to keep the log shipping monitoring service on a SQL Server of its own, not on the source or destination
- Servers participating in log shipping. Not only is this important for fault tolerance, but because the log shipping monitoring service incurs overhead that can affect the performance of the source and destination servers.
- Monitor log shipping daily to ensure that it is working successfully.
- Learn what you need to know to fix shipping if synchronization is lost between the production and backup databases.
- Document, and test your server recovery plan, so you will be ready in case of a server failure.

Case Study: How to add files to a log-shipped database

How to add a file to a log shipped database without reconfiguring log shipping.

I did it in SQL 2008 Enterprise edition server to find out the information.

Steps:

1. I configured log shipping between the databases infisystem
2. Stopped and disabled all the backup, copy and restore jobs.
3. Added a secondary file named Infi_data1.ndf to the log shipped infisystem database in primary server

The following is the script to add a new file to the infisystem database in primary server:

```
ALTER DATABASE infisystem
ADD FILE
(
    NAME = Infi_data1,
    FILENAME = 'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\Infi_data1.ndf',
    SIZE = 5MB,
    MAXSIZE = 15,
    FILEGROWTH = 10%
)
```

4. Manually took a transaction log backup for the database after adding the file
5. Manually copied the transaction log backup file to secondary server
6. Manually restored that particular transaction log backup using the WITH MOVE option and WITH NORECOVERY clause in secondary server

Basically while using the WITH MOVE option I would mention the newly created secondary file i had recently added to the log shipped database.

```
RESTORE filelistonly from disk='D:\Database\SQLDBA\Infisys.trn'
RESTORE log Infisystem FROM Disk='D:\Database\SQLDBA\Infisys.trn'
WITH MOVE 'Infi_data1' TO 'C:\Program Files\Microsoft SQL
Server\MSSQL10.KATMAI\MSSQL\DATA\Infi_data1.ndf',
Norecovery
```

Enable all Jobs and see the log-shipping status in monitor server.

Case Study: how to monitor Log Shipping for SQL Server databases

Log Shipping is a basic SQL Server high-availability technology that is part of SQL Server. It is an automated backup/restore process that allows you to create another copy of your database for failover or reporting purposes.

You can use the below items to investigate if there are any issues with your databases that are setup for Log Shipping. We will cover each of these items and how they can be used.

- SQL Server Error Log
- SSMS Built In Report
- System Stored Procedures
- Query the MSDB database
- Application/System EventViewer Log

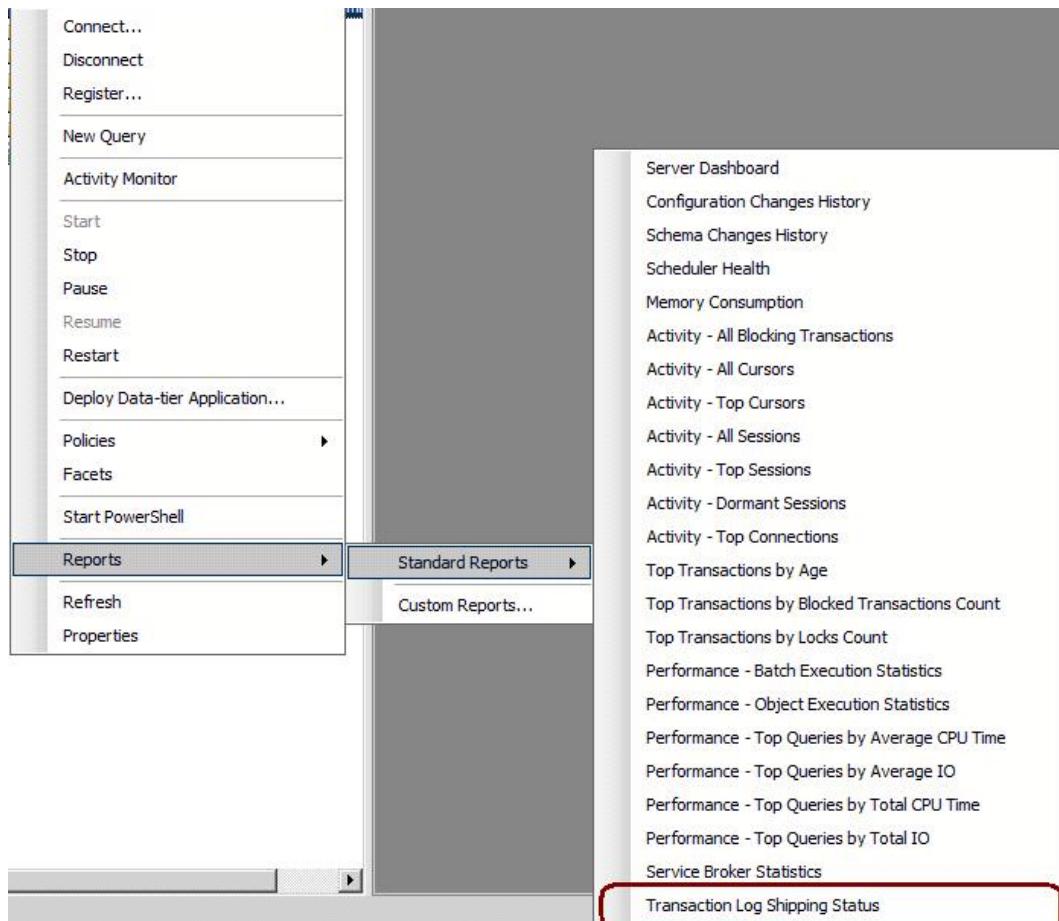
SQL Server Error Log

Check the SQL Server error log for error messages related to log shipping such as database backup and restore failures using these commands.

```
-Query to check the Log Shipping related error messages  
select * from sys.messages where description like '%shipping%' and msplangid = 1033  
--Execute it on Primary/Secondary server  
EXEC xp_readerrorlog 0,1,"Error",Null  
--Execute it on Primary/Secondary server  
EXEC xp_readerrorlog 0,1,"Shipping",Null  
--Execute it on Primary server  
EXEC xp_readerrorlog 0,1,"Backup",Null  
--Execute it on secondary server  
EXEC xp_readerrorlog 0,1,"Restore",Null
```

SSMS Built In Report

You can use the **Log Shipping Status report** by right clicking on the Server Name in Management Studio > Reports > Standard Reports > Transaction Log Shipping Status.



Once you click on the **Log Shipping Status report**, you will get a report as shown below. You can open the status report for the monitoring server, primary server or secondary server. The report will show you the log shipping status (whether it is healthy or not) as well as metadata such as the primary and secondary database names, time since the last backup, last restore file, etc...



This report shows the status of log shipping configurations for which this server instance is a primary, secondary, or monitor.

Status	Primary Database -- Secondary Database	Backup			Copy	Restore				
		Time Since Last	Threshold	Alert Enabled		Time Since Last	Time Since Last	Latency of Last File	Threshold	Alert En
Good	[SQLPRD].SQLDBPool	10 min	120 min	True						

System Stored Procedures

You can execute the below Log Shipping System Stored Procedure to monitor log shipping and get detailed information about log shipping.

- **sp_help_log_shipping_monitor**
 - This is the how SQL Server generates the Log Shipping Status report by executing sys.sp_help_log_shipping_monitor procedure. This procedure returns the log shipping status (whether it is healthy or not) as well as metadata such as primary and secondary database names, time since last backup, last backup file, last restore file, etc...
- **sp_help_log_shipping_monitor_primary**
 - returns all columns from the log_shipping_monitor_primary table for the specified primary log shipping database. It returns server name, database name, time of last backup, backup threshold, threshold alert and history retention period.
- **sp_help_log_shipping_monitor_secondary**
 - returns all columns from log_shipping_monitor_secondary table for the specified secondary log shipping database. It will return database name, server name, restore threshold, last copied file, time of last copy / restore and history retention period.

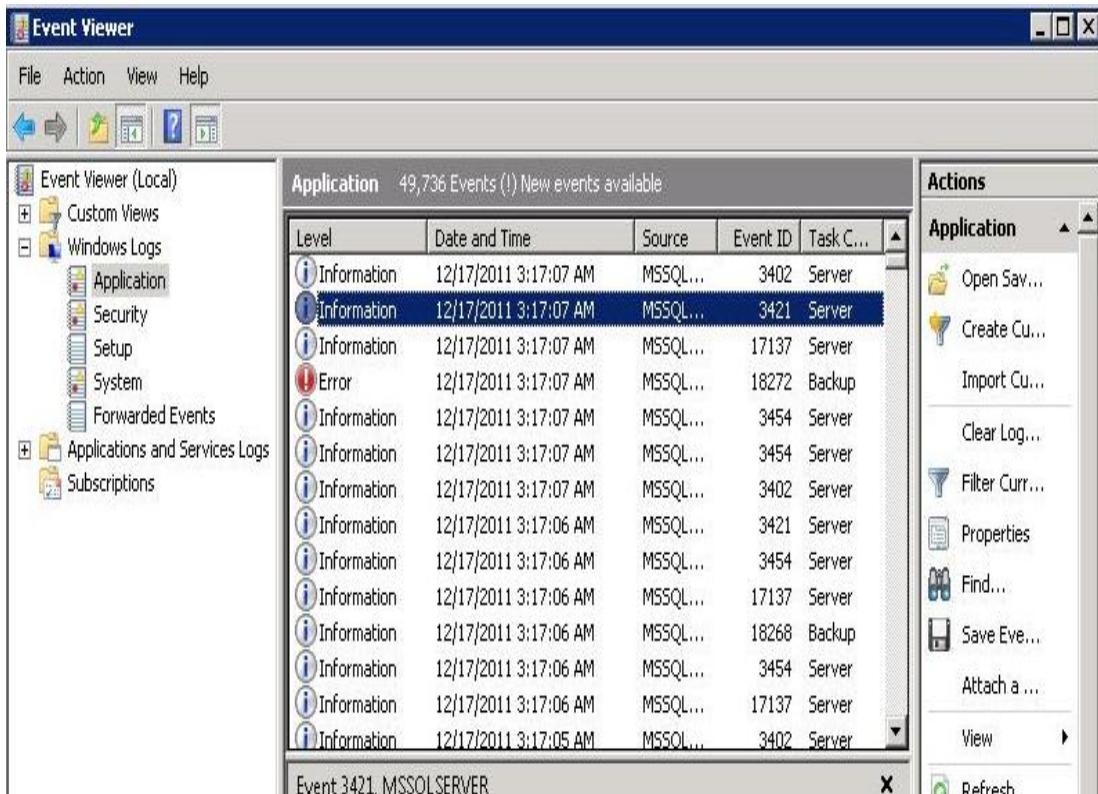
Query the MSDB database

You can monitor the log-shipping jobs and errors from the MSDB tables as well.

```
--Query to list out the Log Shipping Jobs
SELECT *
FROM msdb.dbo.sysjobs
WHERE category_id = 6
--Query to check the job history error messages if any
SELECT *
FROM [msdb].[dbo].[sysjobhistory]
WHERE [message] like '%Operating system error%'
--Query to check the Log Shipping errors
SELECT *
FROM [msdb].[dbo].[log_shipping_monitor_error_detail]
WHERE [message] like '%Operating system error%'
```

Application/System Event Viewer Log

Another option is to check the Application/System Event Viewer Log for any log shipping, backup, restore or system related issues.



Chapter – 17

Database Mirroring

Overview of Mirroring:

Mirroring is mainly implemented for increasing the database availability. Similar to log shipping mirroring is also implemented on per database basis. Database mirroring maintains two copies of a single database that must reside on different instances of SQL Server Database Engine (server instances). Typically, these server instances reside on computers in different locations. One server instance serves the database to clients (the principal server), while the other server instance acts as a hot or warm standby server (the mirror server).

Mirroring provides a hybrid solution i.e

1. Provides a copy of the database like Log Shipping and
2. Rapid failover capabilities like Clustering

Advantages of Mirroring :

- * Increases data protection ---> Depending on the mode of operation Mirroring provides minimal data loss.
- * Increases availability of a database ---> In the event of a disaster, in high-safety mode with automatic failover, failover quickly brings the standby copy of the database online (with no data loss). In the other operating modes, the database administrator has the alternative of forcing service (with possible data loss) to the standby copy of the database.
- * Improves the availability of the production database during upgrades ---> During service packs installation or any patch applied on Principal server which requires downtime, the standby comes into effect.

Components in Mirroring:

Database mirroring consist of the following components

1. Principal ---> The Principal is the originating server i.e it is the source server which contains the database which is configured for mirroring. There can be only one principal database and it has to be in a separate SQL Server instance than the mirror database.
2. Mirror ---> The Mirror is the receiving database in a mirror pair i.e it is the destination server which contains the mirrored database. There can be only one mirror for each principal database. The mirror needs to be on its own separate SQL Server instance preferably on separate physical server.
3. Mirrored Pair ---> A Principal and Mirror operating together are called a Mirrored Pair. The changes on the principal are reflected in the mirrored database
4. Witness ---> A Witness is optional and it monitors the Mirrored Pair. It ensures that both principal and mirror are functioning properly. The Witness is also a separate SQL

Server instance preferably on a separate physical server than principal and mirror. One Witness server can monitor multiple Mirrored Pairs.

5. Quorum ---> A Quorum is the relationship between the Witness, Principal and the Mirror.
6. Endpoint ---> Endpoint is the method by which SQL Server Database engine communicates with applications. In the context of Database mirroring endpoint is the method by which the Principal communicates with the Mirror. The mirror listens on a port defined in the endpoint. The default is 5022. Each database mirror pair listens on its own unique port.

To list all the database mirror endpoints run,

---> Select * from sys.database_mirroring_endpoints

To list all the endpoints

---> Select * from sys.tcp_endpoints

Database Mirroring can be configured for three different operating modes:

High Availability Operating Mode - This provides durable, synchronous transfer of data between principal and mirror, including automatic failure detection and failover. There is performance overhead on this mode because a transaction is not considered committed until SQL Server has successfully committed it to the transaction log on both the principal and the mirror database. And as the distance between the principal and the mirror increases, the performance impact also increases. There is a continuous ping process between all three to detect failover. If the witness server is not visible from the mirror, you must either reconfigure the operating mode for the database mirroring session or turn off the witness.

Alternatively, you can manually fail over a database mirroring session at the mirror in High Availability Mode by issuing the following command at the principal. You can also issue the same command if you have to take principal down for maintenance.

ALTER DATABASE SET PARTNER FAILOVER

High Performance Operating Mode - In this configuration you dont need a WITNESS Server and the Mirror Server acts as an WARM standby and does not support automatic failure detection or failover. There is any asynchronous data transfer between principal and mirror. This mode provide better performance and you can have geographic dispersion between the principal and the mirror.

High Protection Operating Mode (Recommended Mode) - This mode is the same as High Availability Mode except failover is manual and you have to manually promote the mirror to be the principal. Data transfer is synchronous.

Prerequisites for Database Mirroring

1. Make sure that the two partners that is the principal server and mirror server, are running the same edition of Microsoft SQL Server 2012. The partners require either SQL Server 2012 Standard Edition or SQL Server 2012 Enterprise Edition or SQL Server 2012 Developer Edition.

2. If you are using a witness, make sure that SQL Server 2012 is installed on its system. The witness can run on any reliable computer system that supports SQL Server 2012 Standard Edition, Enterprise Edition, or Express Edition.
3. SQL 2012 or later version is required for Mirroring
4. The principal database must be in the FULL recovery model. Log records that result from bulk-logged operations cannot be sent to the mirror database.
5. Verify that the mirror server has enough disk space for the mirror database.
6. All of the server instances in a mirroring session should use the same master code page and collation. Differences can cause a problem during mirroring setup.
7. The mirror database must have the same name as the principal database.
8. The mirror database must be initialized from a restore of the principal database with NORECOVERY, followed by restores in sequence of principal transaction log backups. Prior to configuring mirroring ensure that at least 1 tran log is restored in addition to full backup with NORECOVERY mode.

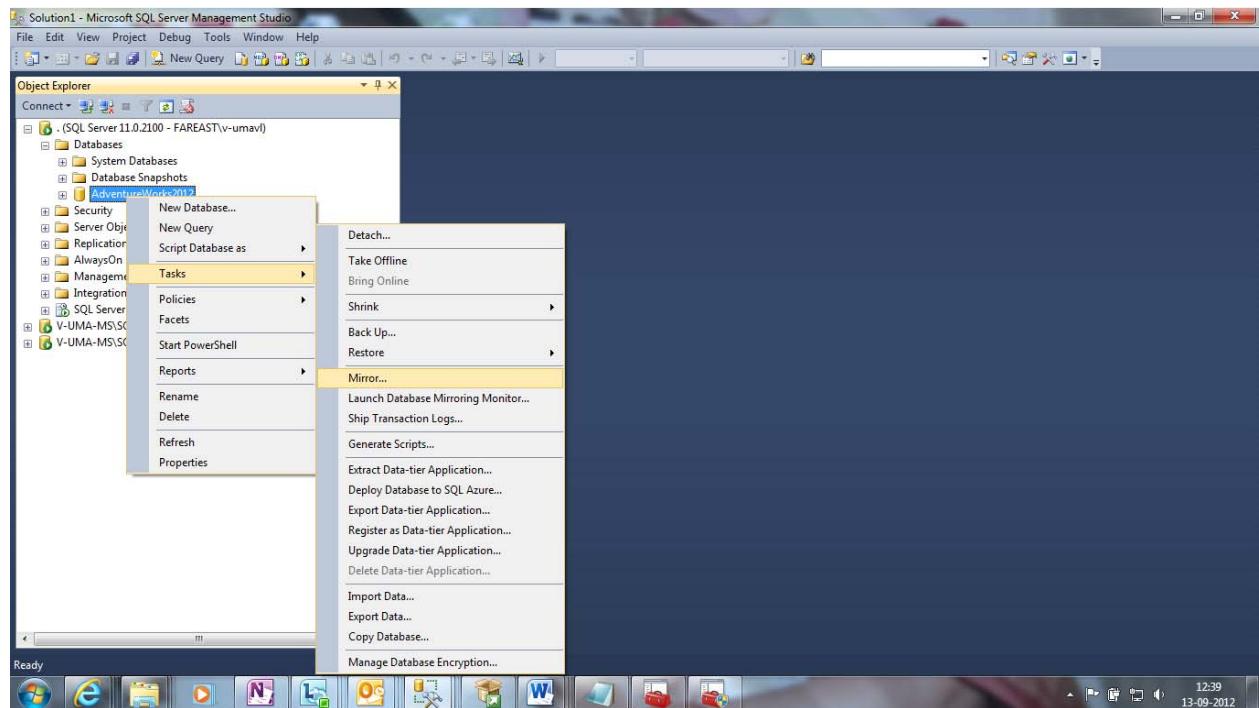
Initialization:

First take full backup and one Transaction log backup from Principal server and restore it on Mirror server with **NO Recovery** Option.

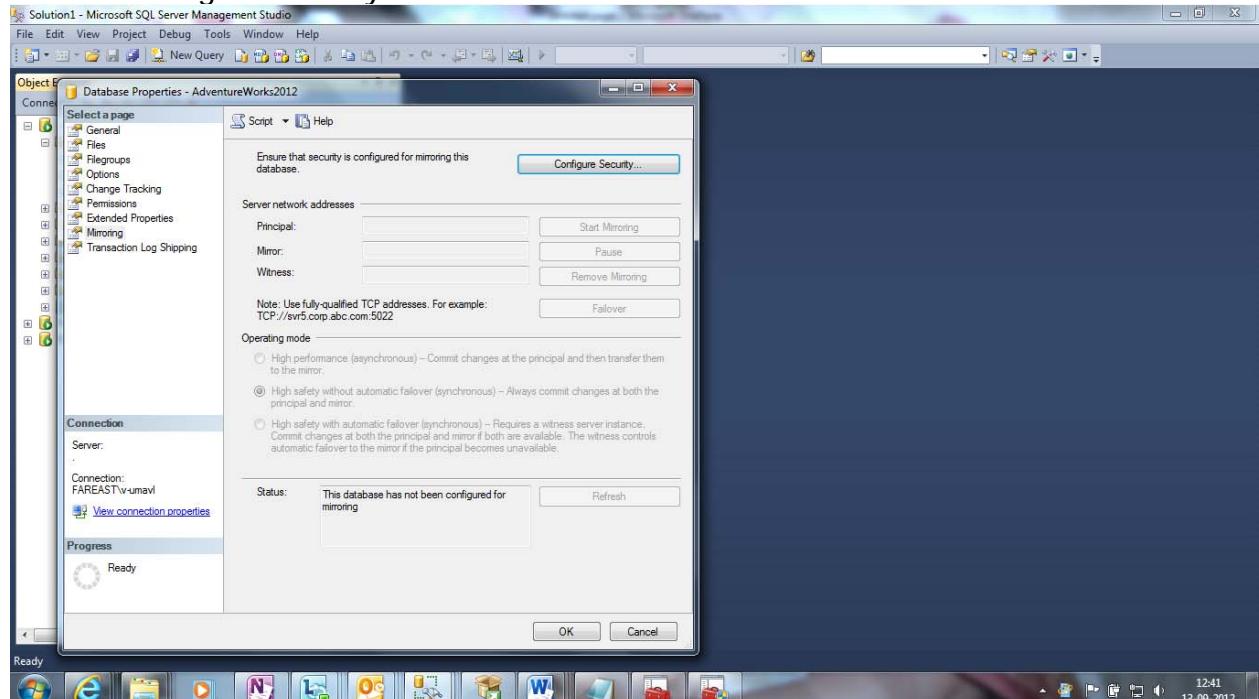
Deployment:

Please follow the screen shots to setup Database mirroring.

Goto principal server→go to database(which needs to be mirrored)→tasks→select Mirror



Click on configure security



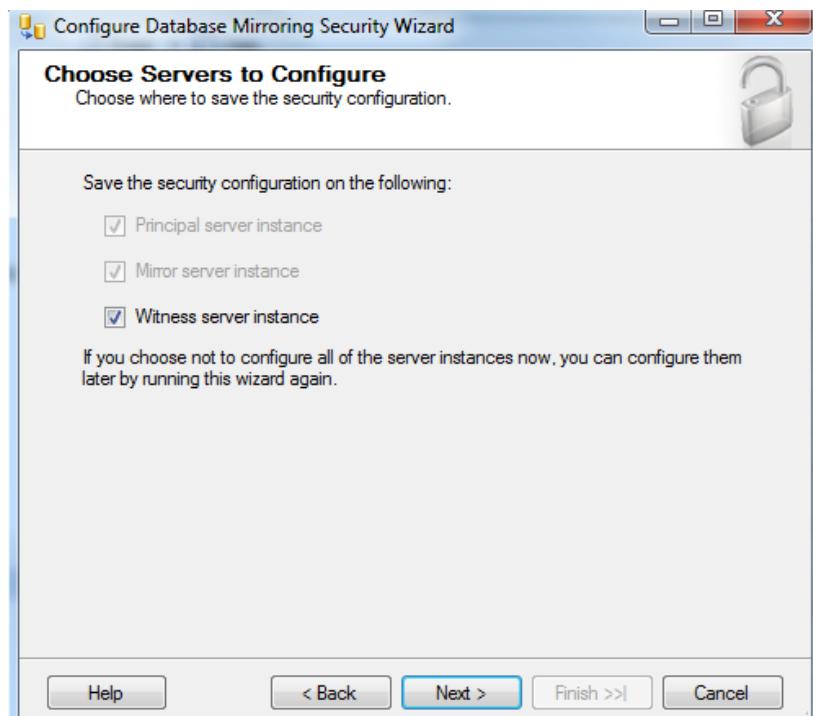
It will launch mirroring wizard



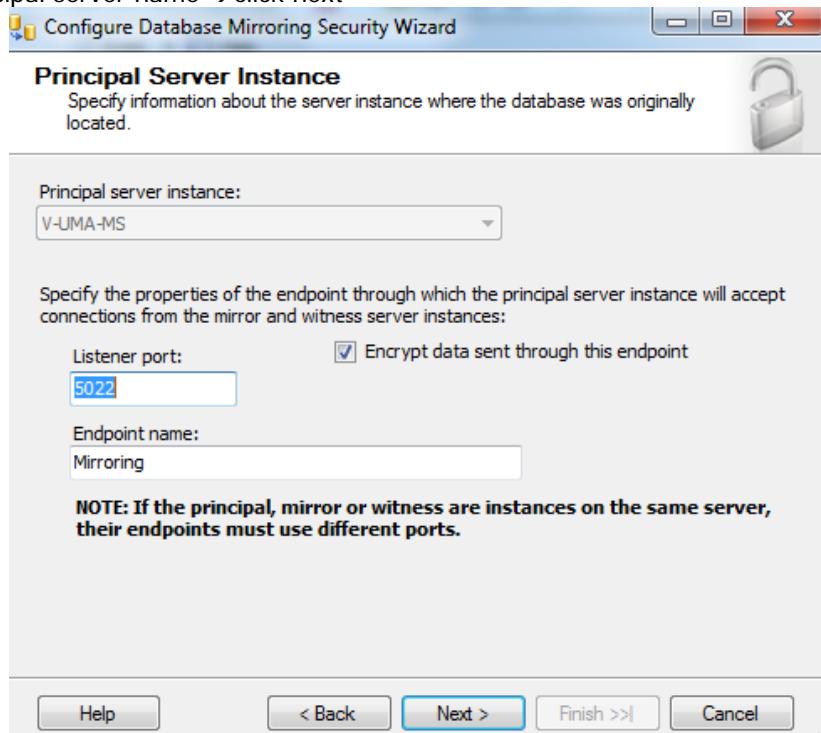
Click—next→select option “yes” for including witness server for high safety with automatic failover mode.



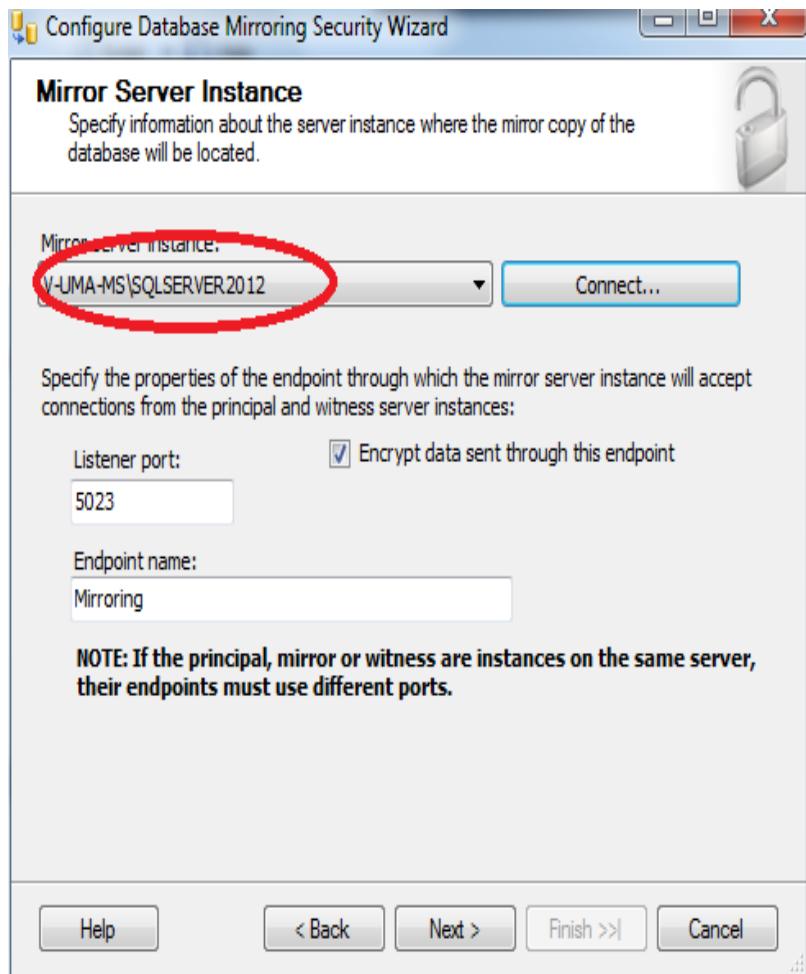
Click next



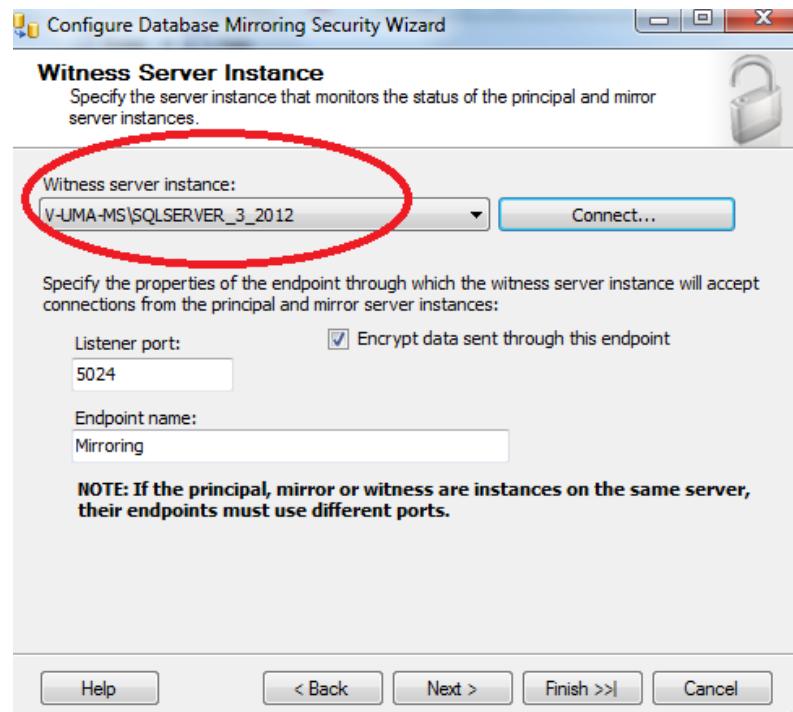
Select principal server name → click next



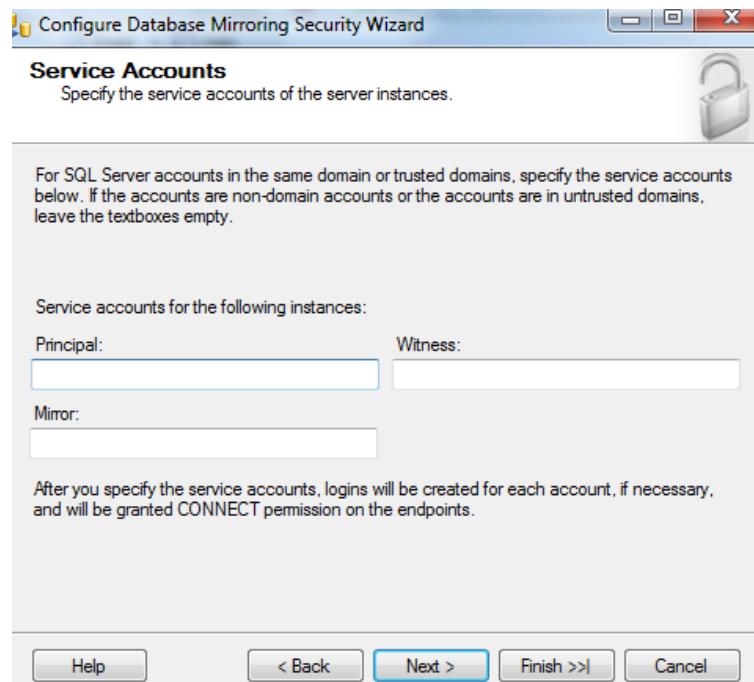
Select the mirrored server → click next



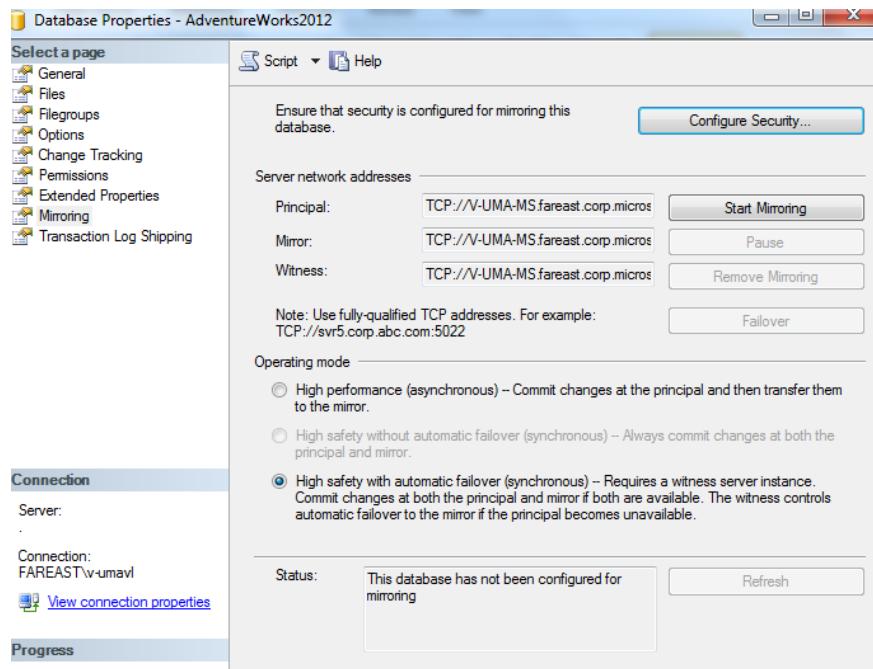
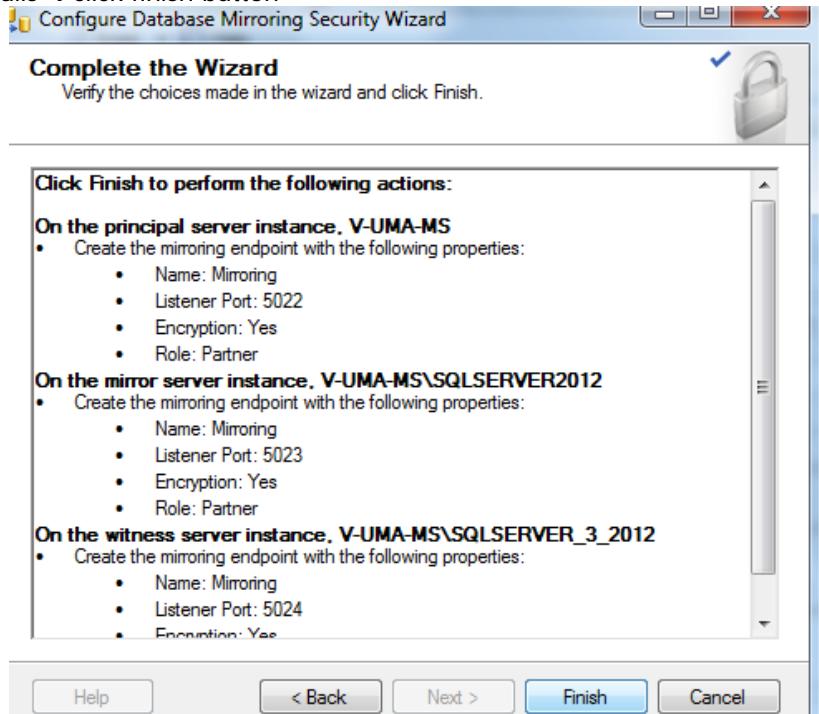
Select the witness server → click next



Leave the boxes if 3 servers are in same domain



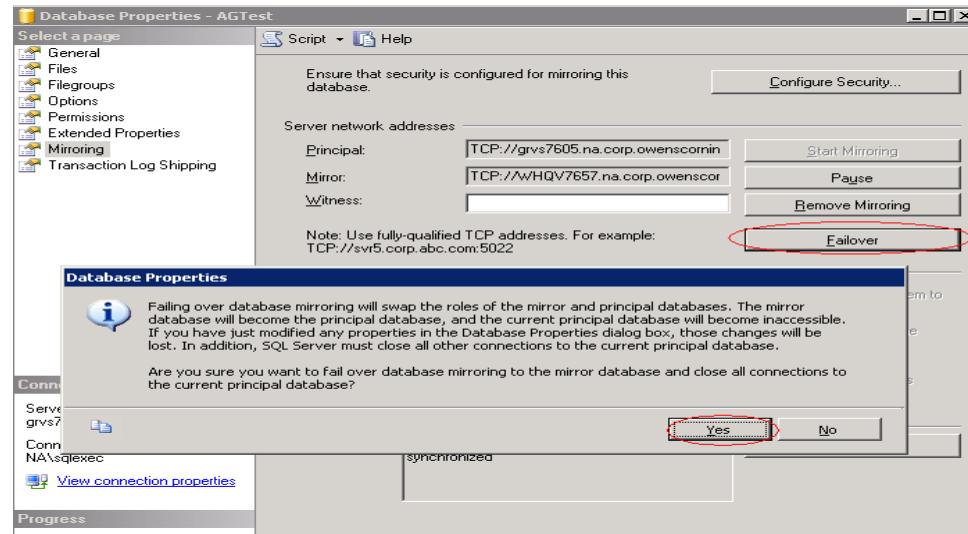
Chck the details → click finish button



Then click start mirroring button...after few seconds you can see the status as synchronized.

Failover in Database Mirroring using GUI mode where both Principal and mirror server's are available.

1. On Principal Server -Database -Right click –Go to –Properties and Click on Mirroring.
2. Click on **Failover** Button then hit **Yes** Button.



Please refresh both Principal and Mirror server's to view the changes

How can I bring mirror database online after principal server is down ?

Safety FULL with Witness:

Well the answer for this 'depends on the mode in which mirroring is configured'. If mirroring is configured in High Availability mode (Full safety) then we don't need to worry about failover as the mirror server will form a quorum with witness and will initiate an automatic failover. The safety level can be set using the below command,

```
ALTER DATABASE dbname SET SAFETY FULL
```

```
ALTER DATABASE dbname SET SAFETY OFF
```

Safety FULL without Witness:

This scenario provides high safety, but automatic failover is not allowed. This mode is called as High Protection mode. In the event of failure of the principal, the database service becomes unavailable. You need manual intervention to make the database service available. You must break the mirroring session and then recover the mirror database.

For example, prior to the failure, Server_A and Server_B acted as principal and mirror respectively. Server_A fails. You need to execute the following on Server_B to make the database service available:

```
ALTER DATABASE dbname SET PARTNER OFF  
RESTORE DATABASE dbname WITH RECOVERY
```

Safety OFF :

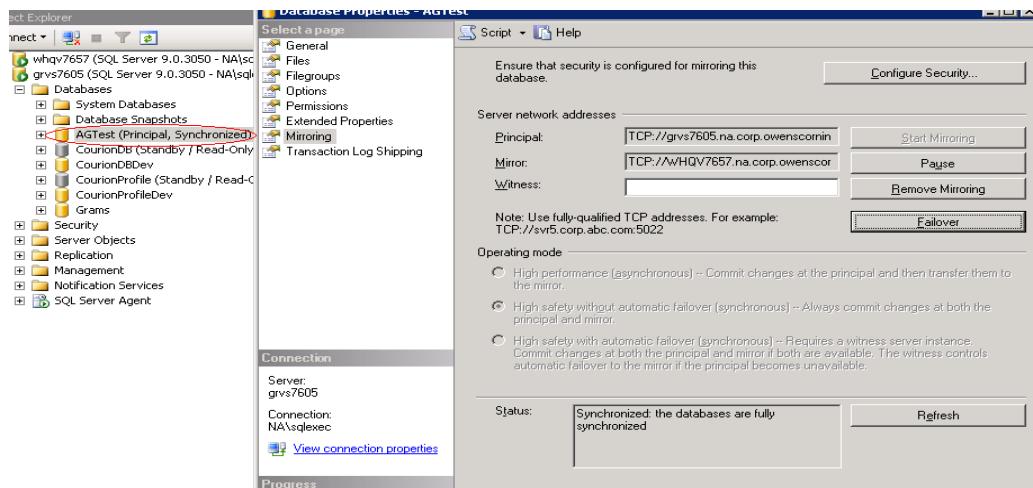
In the event of failure of the principal, the database service becomes unavailable. You can perform a force service to make the database service available on the mirror. However, since the safety level is OFF, it is possible that there were transactions that didn't make it to the mirror at the time of the failure of the principal. These transactions will be lost. Therefore, manual failover with safety OFF involves acknowledging the possibility of data loss. For example, prior to the failure, Server_A and Server_B acted as principal and mirror respectively. Server_A fails. You need to execute the following on Server_B to make the database service available:

```
ALTER DATABASE dbname SET PARTNER FORCE_SERVICE_ALLOW_DATA_LOSS
```

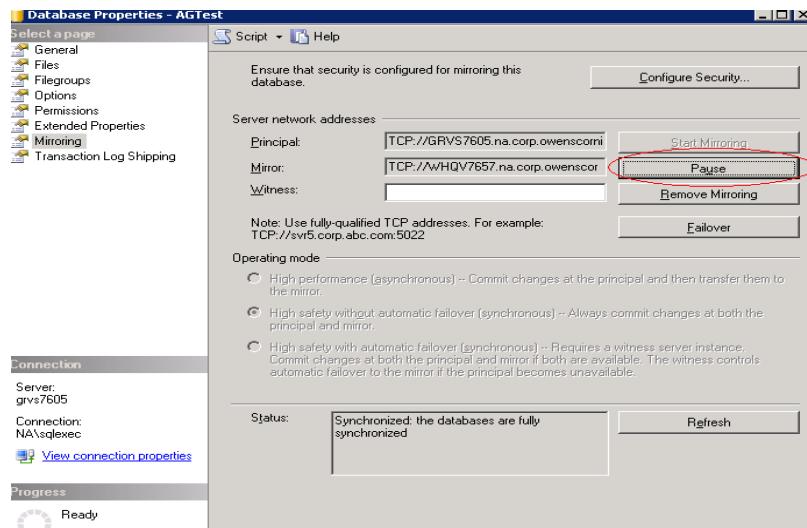
Once the database on Server_A becomes operational, it automatically assumes the role of the mirror. However, the mirroring session remains SUSPENDED, and you will need to manually RESUME the mirroring session.

This can be achieved using GUI Mode also.

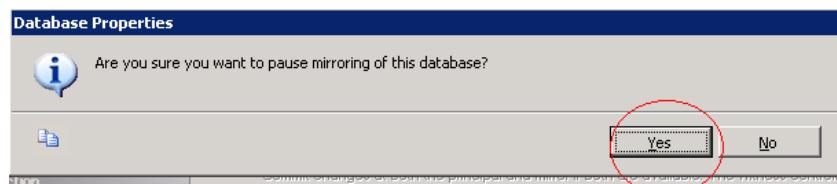
1. On Principal Server - <Database> -Right click -Go to -Properties and Click on Mirroring.



2. Click on **Pause** Button to suspend the database mirroring.

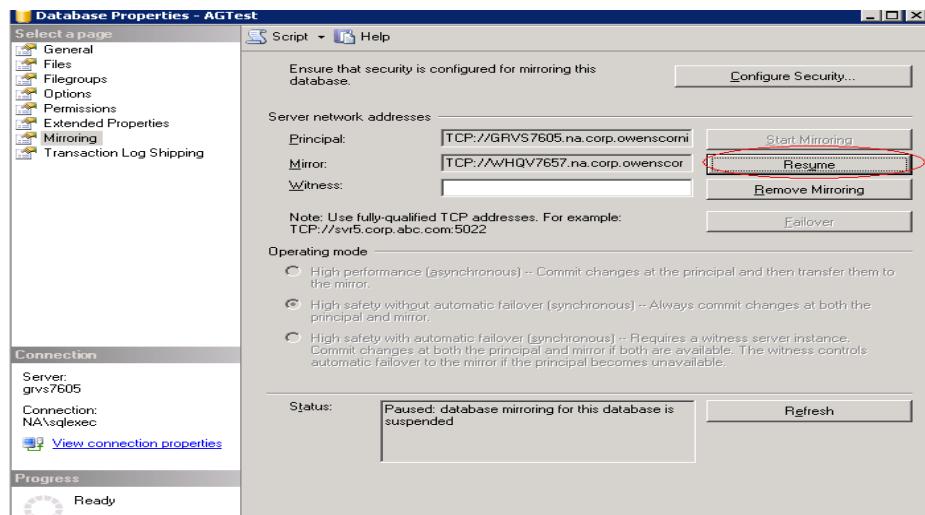


3. Click on **Yes** Button



Now database mirroring is in Suspended mode.

4 .To Resume the database mirroring on same screen Click on **Resume** Button



Advantages Database Mirroring:

- Database mirroring architecture is more robust and efficient than Database Log Shipping. It can be configured to replicate the changes synchronously to minimized data loss.
- It has automatic server failover and client failover mechanism.
- Configuration is simpler than log shipping and replication, and has built-in network encryption support (AES algorithm).
- Because propagation can be done asynchronously, it requires less bandwidth than synchronous method (e.g. host-based replication, clustering) and is not limited by geographical distance with current technology.
- Database mirroring supports full-text catalogs.
- Does not require special hardware (such as shared storage, heart-beat connection) and clusterware, thus potentially has lower infrastructure cost

Disadvantages of Log Shipping:

- Potential data lost is possible in asynchronous operation mode. RTO will vary and depend on several factors, such as propagation interval time and bandwidth speed.
- Mirror server/database is not available for user operation.
- It only works at database level and not at server level. It only propagates changes at database level, no server level objects, such as logins and fixed server role membership, can be propagated.
- Automatic server failover may not be suitable for application using multiple databases.

Mirroring Vs Clustering

Clustering	Database Mirroring
<p>Clustering failover using SQL Virtual Server provides the highest availability because it immediately fails over to the second node. This failover is transparent to the end-user/client application. Clustering failover provides protection against SQL Server failures, Windows operating system crashes and hardware failures (other than disk subsystem).</p> <p>Clustering failover requires special hardware. Also, failover clustering uses a shared disk subsystem, and therefore, the computers must be physically located in the same data center, unless you plan to</p>	<p>This same can be achieved with High-Availability operation mode without additional witness server. Db mirroring provides protection against SQL server, database failures.</p> <p>Here incase of db mirroring you don't require any additional hardware. To configure db mirroring, we need an SQL instance or server linked to the primary server which will provide you high availability.</p>

implement Distance Clustering. Failover clustering does not protect you against a failure in the disk subsystem and the data loss that results because of the hardware failure.

Troubleshooting Information on Database Mirroring

Mirroring Catalog Views:

A database mirroring session consists of the relationship formed between the partner servers and potentially the witness server. Each of the participating servers keeps some metadata about the session and the current state of the databases.

- * *sys.database_mirroring* ---> provides information about principal and mirror
- * *sys.database_mirroring_witnesses* ---> provides information about witness server

All the metadata required for database mirroring (in particular the mirroring failover Lsn and partner server names) are kept by the mirroring partners. The witness only keeps data necessary for its role as a witness in a High Availability mode, in particular the role sequence number, which tracks the number of role changes in the session.

Database mirroring states and transition:

Database states for each server are kept during the database mirroring session, recorded on each partner server, and reported by the *sys.database_mirroring* catalog view. The *mirroring_state* column returns a number for the state, and the *mirroring_state_desc* column returns the descriptive name for the state. State information about the witness is also reported from the same catalog view.

In addition to the states reported for each database, there are three phrases that are useful in describing the servers and databases involved in database mirroring.

1. **Exposed** - The data on the principal is exposed when it is processing transactions but no log data is being sent to the mirror. When a principal database is exposed, it is active with user connections and processing transactions. However, no log records are being sent to the mirror database, and if the principal should fail, the mirror will not have any of the transactions from the principal from the point the principal entered the exposed state. Also, the principal's transaction log cannot be truncated, so the log file will be growing indefinitely.
2. **Cannot serve the database** - When a principal server does not allow any user connections to the database and any transactions to be processed. When a witness has been set, if the principal server cannot form a quorum with another server, it will stop serving the database. It will not allow user connections and transactions on the principal database, and will disconnect all current users. As soon as it can form a quorum again, it will return to serving the database.
3. **Isolated** - A server is isolated when it cannot contact any of the other servers in the database mirroring session, and they cannot contact it. A server may be operational but communication lines are down between it and both other servers in the database mirroring session. In that case, we'll call the server isolated. If a witness has been set, then, if the

principal server becomes isolated, it will no longer be able to serve the database, because there is no server in the session with which it can form a quorum.

When safety is FULL, the principal first enters the SYNCHRONIZING state and as soon as it synchronizes with the mirror, both partners enter the SYNCHRONIZED state. When safety is OFF, the partner databases start with the SYNCHRONIZING state. Once the mirror has caught up, the state goes to SYNCHRONIZED and stays there regardless of how far behind it is. For both safety settings, if the session is paused or there are redo errors on the mirror, the principal enters the SUSPENDED state. If the mirror becomes unavailable, the principal will enter the DISCONNECTED state.

In the DISCONNECTED and SUSPENDED states:

- * When a witness has been set, if the principal can form a quorum with the witness or mirror server, the principal database is considered exposed. That means the principal database is active with user connections and processing transactions. However, no log records are being sent to the mirror database, and if the principal should fail, the mirror will not have any of the transactions from the principal from the point the principal entered that state. Also, the principal's transaction log cannot be truncated, so the log file will be growing indefinitely.
- * When a witness has been set, if the principal cannot form a quorum with another server, it cannot serve the database. All users will be disconnected and no new transactions will be processed.
- * When safety is OFF, the principal database is considered exposed, because no transaction log records are being sent to the mirror

Database Snapshots

What is a Database Snapshot?

- * The Database Snapshot is a feature that is available in Enterprise edition of SQL Server 2012.
- * It provides a read-only, static view of a database.
- * Multiple snapshots can exist for a single database on the same server instance as the database.

A Database Snapshot is a read-only, static view of a database (the *source database*). Multiple snapshots can exist on a source database and can always reside on the same server instance as the database. Each database snapshot is consistent, in terms of transactions, with the source database as of the moment of the snapshot's creation. A snapshot persists until it is explicitly dropped by the database owner.

If you do not know how Snapshot database work, here is a quick note on the subject. However, please refer to the official description on Book-on-Line for accuracy. Snapshot database is a read-only database created from an original database called the "source database". This database operates at page level. When Snapshot database is created, it is produced on sparse files; in fact, it does not occupy any space (or occupies very little

space) in the Operating System. When any data page is modified in the source database, that data page is copied to Snapshot database, making the sparse file size increases. When an unmodified data page is read in the Snapshot database, it actually reads the pages of the original database. In other words, the changes that happen in the source database are reflected in the Snapshot database.

Database Snapshots are completely unrelated to Snapshot Replication and Snapshot Isolation.

Uses of Database Snapshot

The following are the important usage of snapshots namely,

1. **Reporting Purposes -->** When your db is used for reporting purposes you can create snapshots so that they provide a static view of the data and also prevents blocks from occurring as insert or update statements are prevented as snapshots are read-only copies of your source db.
2. **Querying data from standby servers -->** In Database mirroring the mirror db will be in NORECOVERY mode and you cannot read the contents of the db. If you want to read through the contents of mirrored db or if you want to use the mirrored db for reporting purposes you can make use of Database Snapshots.
3. **System upgrades -->** Before applying service packs the DBA will take backup for all the databases which will be time consuming and also takes lot of space. In such scenarios database snapshots comes handy. Creating the snapshot consumes less amount of time and also the space required is very minimal.

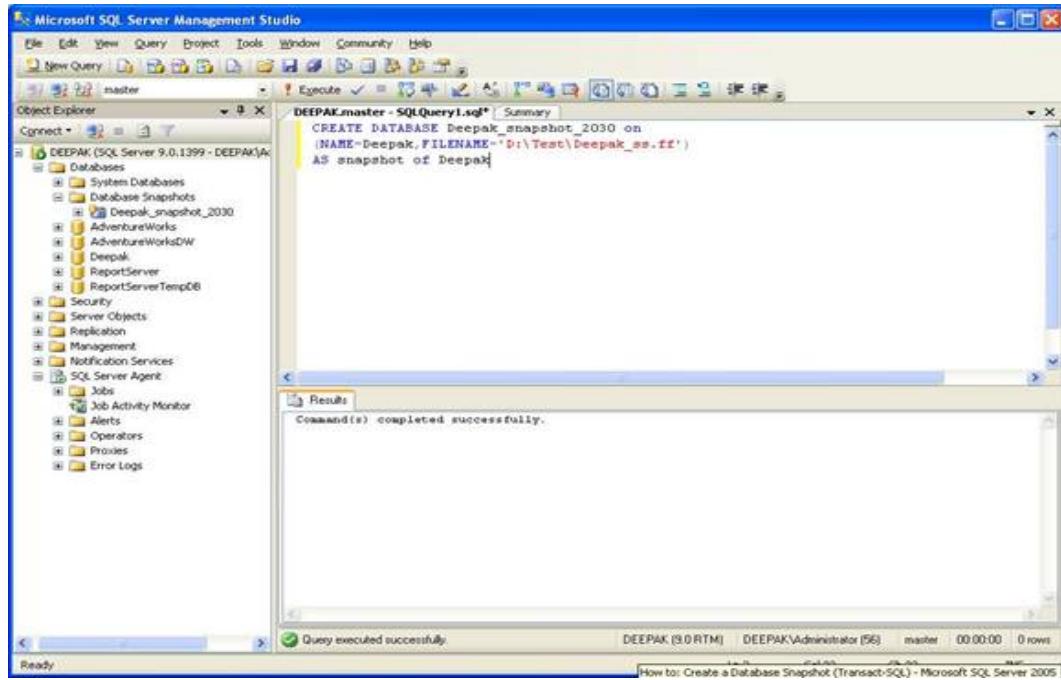
Creating Database Snapshot

One of the exasperating things while creating a snapshot is that you cannot use SQL Server Management Studio (SSMS) instead you need to make use of T-SQL. I am creating a snapshot of the database "Deepak" as follows,

```
CREATE DATABASE Deepak_snapshot_2030 on  
(NAME=Deepak,FILENAME='D:\Test\Deepak_ss.ss')  
AS SNAPSHOT OF Deepak
```

The Snapshot name is "Deepak_snapshot_2030". It can be any name but its advisable to include the time @ which the snapshot was created, in my case it is 2030(i.e 8:30 pm) so that it will be easy to identify. While creating a snapshot a Sparse file is created in my case it is named as "Deepak_ss.ss".

You need to ensure that the drive containing this sparse file must have NTFS file system else you will not be able to create the snapshot. Initially I tried to create a database snapshot in F:\drive which has FAT file system and received the below error. Refer the below figure,



Best Practices of Mirroring

- The principal database and the mirror database should be on separate physical hardware, and ideally, in different physical locations.
- The witness server should be on separate physical hardware, and be on a separate network (best if at a third location).
- Initial database mirroring setup should be done during less busy times, as the setup process can negatively affect performance of the production database being mirrored.
- Use high availability mode whenever possible, and high performance mode only when required.
- The hardware, along with the OS and SQL Server configuration, should be identical (at least very similar) between the two servers.
- While a fast connection is not required between mirrored servers, the faster the connection, and the better quality the connection, the better.
- You will want to optimize the performance of the mirrored database as much as possible to reduce the overhead caused by the mirroring process itself.
- Thoroughly test database mirroring before putting it into production.
- Monitor database mirroring daily to ensure that it is working properly, and is meeting performance goals.
- Develop a formal operational and recovery procedure (and document) to support mirroring. Periodically test the failover process to ensure that it works.

Case study/practical troubleshooting:

1: Login Failures while connecting to new principal database after failover?

After configuring database mirroring in SQL Server 2008 and performing failover, the original mirror database now becomes the new principal database. We might have even created the same login (as in principal) in original mirror server prior to failover. But after failover if we try to connect or if the application tries to connect, the following error will be returned,

Cannot open database requested by the login. The login failed.

In that case we need to map the login to the user in the database using the procedure `sp_change_users_login` after which the application or the user will be able to successfully connect to the new principal database.

This problem occurs because the SIDs the SQL Server logins on each server do not match. Although the names for the logins are the same, the login is resolved via the SID. This is not a problem with Windows/Domain user/group logins because the SIDs for these logins are created based on the domain SID for the user/group, and hence will be the same for the same given user/group no matter what SQL Server the user/group is added to.

In order to make the `sp_change_users_login` synchronization step unnecessary, we need to create the SQL Server logins on the mirror server not only with the same name, but also with the same SID as on the principal server. This can be accomplished by using the SID specification in the 'CREATE LOGIN' statement when creating the logins on the mirror server. Here is an example where we create a the same login in mirror server as the one in principal server.

`CREATE LOGIN WITH PASSWORD ='password',SID ='sid for same login on principal server'`

To retrieve the SID for each login from the principal server query the `sys.sql_logins` catalog view.

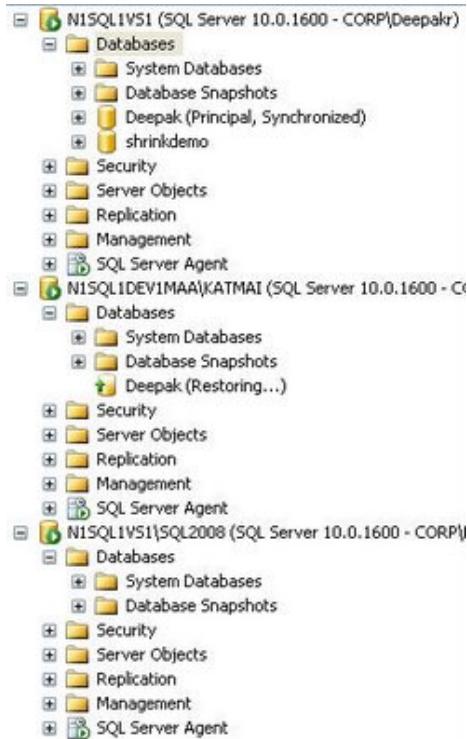
You can also create all the logins with same SID in mirror server from principal server using `sp_helprevlogin` procedure. Consider this step as pre-requisite for configuring db mirroring.

2: How to move the database files of a mirrored database to a new location without any downtime.

Principal Server – N1SQL1VS1 – Dbname is “Deepak”

Mirror Server – N1SQL1DEV1MAA\KATMAI

Witness Server – N1SQL1VS1\SQL2008 - Dbname is “Deepak”



I have used witness server to facilitate automatic failover. All the 3 machines are having SQL 2008 RTM Enterprise edition as shown in the below screenshot.

I am moving the log file from C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA folder to C:\Temp folder.

sp_helpdb Deepak					
Results		Messages			
name	db_size	owner	dbid	created	status
Deepak	741.38 MB	CORP\Deepakr	5	Jan 2 2009	Status=ONLINE, Updateability=READ_WRITE, UsedAcc...
name fileid filename					
Deepak	1	C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\Deepak.mdf			
Deepak_log	2	C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\Deepak_log.ldf			

The steps I am mentioning below are applicable for SQL 2012 and SQL 2008 servers.

Step1: Execute the below query in Principal server,

USE Master

ALTER DATABASE Deepak

```
MODIFY FILE (NAME='Deepak_log',
FILENAME='C:\TEMP\Deepak_log.ldf')
```

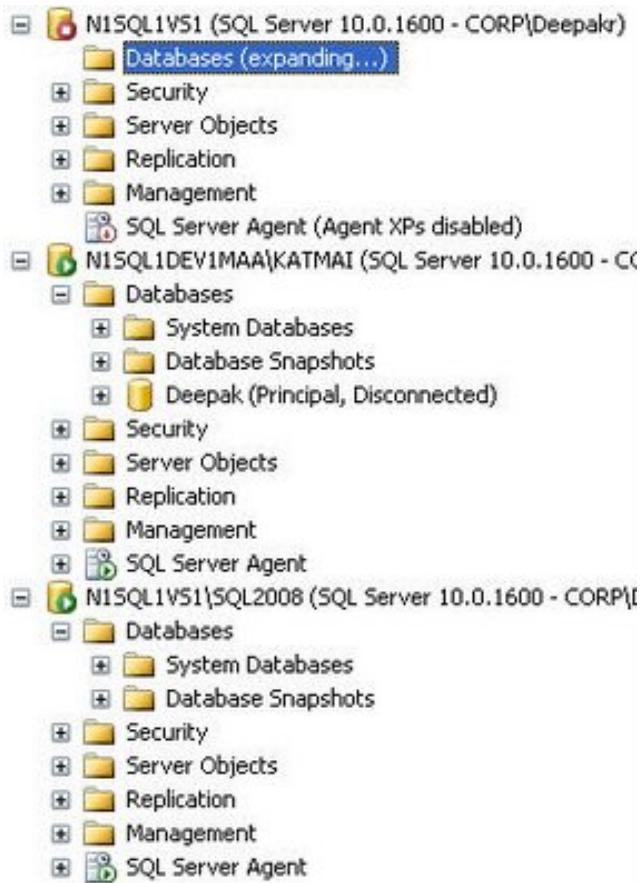
The following will be the output of the above command.

The file "Deepak_log" has been modified in the system catalog.

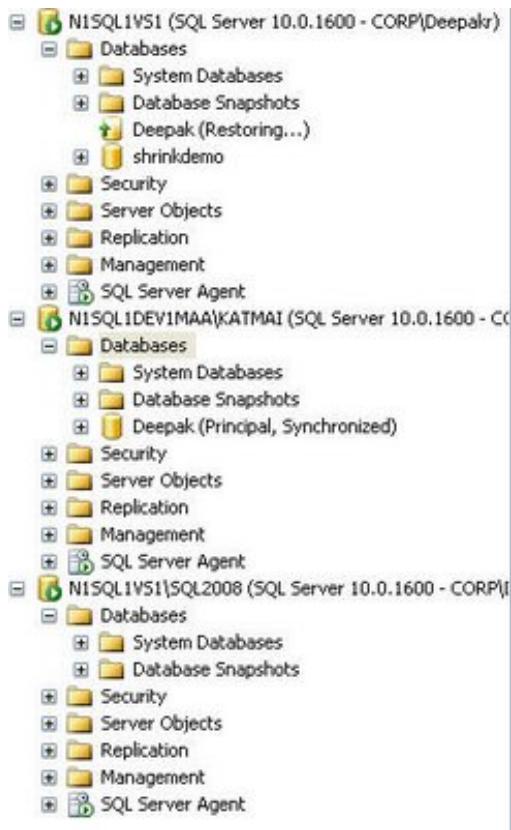
The new path will be used the next time the database is started.

Step2: Now stop SQL Server in principal server N1SQL1VS1, move the deepak_log.ldf file to C:\temp

In principal server, As shown in the below screenshot the mirror server N1SQL1DEV1MAA\KATMAI now becomes the new principal server and will be in disconnected state as the SQL Server in original principal server is down.



Step3: Start SQL Service in N1SQL1VS1 and it will now become the new mirror server and the database will be in "Restoring state". The new principal server is "N1SQL1DEV1MAA\KATMAI" and will be in "Synchronized state" as shown in the below screenshot.



Step4: Now to move the log file in the new principal server N1SQL1DEV1MAA\KATMAI we have to repeat the steps mentioned in Step1. The database files reside in D:\database\data\ folder and will be moved to C:\temp folder.

Step5: Execute the below query in new Principal server, "N1SQL1DEV1MAA\KATMAI"

```
USE Master
```

```
ALTER DATABASE Deepak
```

```
MODIFY FILE
```

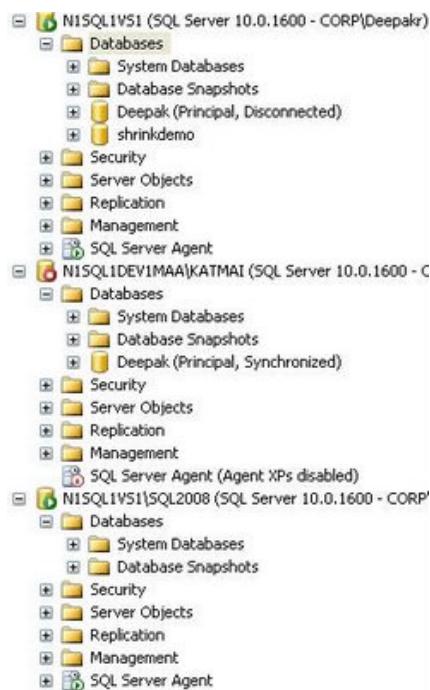
```
(NAME='Deepak_log', FILENAME='C:\TEMP\Deepak_log.ldf')
```

The following will be the output of the above command.

The file "Deepak_log" has been modified in the system catalog

. The new path will be used the next time the database is started.

Step6: Now stop SQL Service in the new principal server N1SQL1DEV1MAA\KATMAI (failover will happen and the now the principal server is N1SQL1VS1 and mirror server will be N1SQL1DEV1MAA\Katmai) and move the database files to the new location C:\Temp and start SQL Server as shown in the below screenshot.



Step7: Connect to the new principal server N1SQL1VS1 and run sp_helpdb deepak and check the location of the database files. As shown in the below screenshot it will now reside in the new location C:\temp.

Similarly do a failover and run sp_helpdb deepak, you can see that the new location for the new database as C:\Temp.

name	fileid	filename	filegroup	size	maxsize	growth	usage
Deepak	1	C:\Program Files\Microsoft SQL Server\MSSQL10.MSS...	PRIMARY	133376 KB	Unlimited	1024 KB	data only
Deepak_log	2	C:\TEMP\Deepak_log.ldf	NULL	625732 KB	2147483648 KB	10%	log only

Chapter – 18

Replication

As it relates to SQL Server, replication is a way of keeping data synchronized in multiple databases. Implementing and maintaining replication might not be a simple proposition: If you have numerous database servers that need to be involved in various types of replication, a simple task can quickly become complex. Implementing replication can also be complicated by the application architecture.

Microsoft SQL Server has supported replication since version 6.0, and setting up replication has become significantly easier over the years (in fact, 99 percent of replication setup can be accomplished by clicking through replication wizards). However, replication involves much more than setup, and unfortunately there aren't many sources of information for implementing and troubleshooting it. The only way to learn replication is to dig through the knowledge base articles and figure things out on your own.

Replication Terminology

SQL Server replication is commonly described by using the publisher/subscriber metaphor. A database server that makes data available for replication (source server) is referred to as the *publisher*; a collection of one or more database objects that are enabled for replication is called a *publication*. SQL Server supports replicating tables, views, stored procedures, and user-defined functions.

One or more servers that get data and/or transactions from the publisher are called *subscribers*. Replication is managed by the system database, which by default is called *distribution*. A distribution database—which can reside on the publisher, subscriber, or on a separate server—is created when you configure replication.

The server that hosts the distribution database is referred to as the distribution server or *distributor*.

It is recommended that you always use a server that is dedicated to distributing transactions. Thus, the distribution server should be used for nothing but replication.

Each database server can act as a publisher and subscriber at the same time. Each publisher can have multiple subscribers, and each subscriber can receive transactions from multiple publishers.

You should also become familiar with replication *agents*, which are implemented as SQL Server jobs that perform a particular task according to their schedule.

Snapshot Replication

Snapshot replication simply takes a "snapshot" of the data on one server and moves that data to another server (or another database on the same server). After the initial synchronization snapshot, replication can refresh data in published tables periodically—based on the schedule you specify. Although snapshot replication is the easiest type to set up and maintain, it requires copying all data each time a table is refreshed.

Between scheduled refreshes, data on the publisher might be very different from the data on subscriber. In short, snapshot replication isn't very different from emptying out the destination table(s) and using a DTS package to import data from the source.

Transactional Replication

Transactional replication involves copying data from the publisher to the subscriber(s) once and then delivering transactions to the subscriber(s) as they occur on the publisher. The initial copy of the data is transported by using the same mechanism as with snapshot replication: SQL Server takes a snapshot of data on the publisher and moves it to the subscriber(s). As database users insert, update, or delete records on the publisher, transactions are forwarded to the subscriber(s).

To make sure that SQL Server synchronizes your transactions as quickly as possible, you can make a simple configuration change: Tell it to deliver transactions continuously. Alternatively, you can run synchronization tasks periodically. Transactional replication is most useful in environments that have a dependable dedicated network line between database servers participating in replication. Typically, database servers subscribing to transactional publications do not modify data; they use data strictly for read-only purposes. However, SQL Server does support transactional replication that allows data changes on subscribers as well.

Merge Replication

Merge replication combines data from multiple sources into a single central database. Much like transactional replication, merge replication uses initial synchronization by taking the snapshot of data on the publisher and moving it to subscribers. Unlike transactional replication, merge replication allows changes of the same data on publishers and subscribers, even when subscribers are not connected to the network. When subscribers connect to the network, replication will detect and combine changes from all subscribers and change data on the publisher accordingly. Merge replication is useful when you have a need to modify data on remote computers and when subscribers are not guaranteed to have a continuous connection to the network.

Replication can be used effectively for many different purposes, as discussed in the following sections.

Providing High Availability

Occasionally, you might consider using replication for high availability; that is, to replicate transactions from the main server to a standby server. If the main server fails, you can then point your data sources to the standby server. Be aware that using replication for high availability takes careful planning and testing. Replication does not provide any sort of automatic fail-over. SQL Server supports other methods of providing high availability, such as clustering and log-shipping, which might be more appropriate for your environment.

Transporting Data

Another common use for replication is to simply move data changes from publishers to subscribers. This method is particularly useful for moving transactional data to a data warehousing server, in which it is transformed and aggregated for OLAP reporting. SQL Server provides other ways of transporting data: DTS, BCP, BULK INSERT statements, and others. Be

sure to carefully consider the alternatives before implementing replication because other solutions might be cheaper or even faster than replication.

Replication needs to be planned carefully. Setting things up is easy, but there is no magic UNDO button that will reverse all your actions. Therefore, be sure to test your plan thoroughly before implementing a replication solution. The following sections discuss some of the planning steps necessary for transactional replication.

Replication Agents

Transactional replication involves three agents: snapshot, log reader, and distribution. The snapshot agent takes a snapshot of records on the publisher and copies the data out to the snapshot folder. The snapshot agent also generates scripts for database schema and includes CREATE TABLE and CREATE INDEX scripts.

The snapshot agent doesn't have to run continuously for transactional replication. If you get your replication working properly the first time, you might never have to run the snapshot agent again after the initial synchronization. However, if you do have problems with the subscriber servers missing data, the snapshot agent is there to help.

The log reader agent reads the transaction log on the published databases. This agent moves transactions that are marked for replication to the distribution database. The distribution agent delivers transactions from the distribution database to the subscribers. Log reader and distribution agents have to run continuously (or at scheduled intervals) to keep replication working.

In addition to snapshot, log reader, and distribution agents, replication also uses a few other jobs (agents) to keep things organized. The history cleanup agent, for example, is used to delete transactions that have already been delivered from the distribution database. Indeed, if this agent did not work, the distribution database would grow very large.

Agent Profile Settings

Replication agents are implemented as SQL Server jobs that call executable files with certain parameters. You should be aware that clicking through the replication wizards configures agents to run with default parameters. If you need to tweak agent parameters for troubleshooting or for performance reasons, you'll have to modify the replication agent's profile. (I'll discuss replication agents' parameters in the next article.)

Security Considerations

Replication agents must have appropriate security permissions to read data on the publisher, move transactions to the distributor and apply the data and transactions to the subscribers. You can allow replication agents to run using security credentials of SQL Server Agent service; alternatively, you can define a login that has a more limited set of permissions. Security is not a joking matter: Allow your replication agents too much freedom, and a hacker can destroy your data on publishers as well as subscribers. On the other hand, not granting sufficient permissions to the agents prevents replication from working properly.

Considerations for Transactional Replication

There are a number of considerations for transactional replication:

- Transaction log space.

For each database that will be published using transactional replication, ensure that the transaction log has enough space allocated. The transaction log of a published database might require more space than the log of an identical unpublished database, because the log records are not truncated until they have been moved to the distribution database.

If the distribution database is unavailable, or if the Log Reader Agent is not running, the transaction log of a publication database continues to grow. The log cannot be truncated past the oldest published transaction that has not been delivered to the distribution database. We recommend that you set the transaction log file to auto grow so that the log can accommodate these circumstances.

We recommend that you set the **sync with backup** option on the distribution database, which delays the truncation of the log on the publication database until the corresponding transactions in the distribution database have been backed up. This can result in a larger transaction log in the publication database.

- Disk space for the distribution database.

Ensure that you have enough disk space to store replicated transactions in the distribution database:

If you do not make snapshot files available to Subscribers immediately (which is the default): transactions are stored until they have been replicated to all Subscribers or until the retention period has been reached, whichever is shorter.

If you create a transactional publication and make the snapshot files available to Subscribers immediately: transactions are stored until they have been replicated to all Subscribers or until the Snapshot Agent runs and creates a new snapshot, whichever is longer. If the elapsed time between Snapshot Agent runs is greater than the maximum distribution retention period for the publication, which has a default of 72 hours, transactions older than the retention period are removed from the distribution database.

Although making the snapshot available to Subscribers immediately improves the speed with which new Subscribers have access to the publication, the option can result in increased disk storage for the distribution database. It also means that a new snapshot is generated each time the Snapshot Agent runs. If the option is not used, a new snapshot is generated only if there is a new subscription.

- Primary keys for each published table.

All published tables in transactional replication must contain a declared primary key. Existing tables can be prepared for publishing by adding a primary key using the Transact-SQL statement

Configuring the Transactional Replication with SQL server 2005.

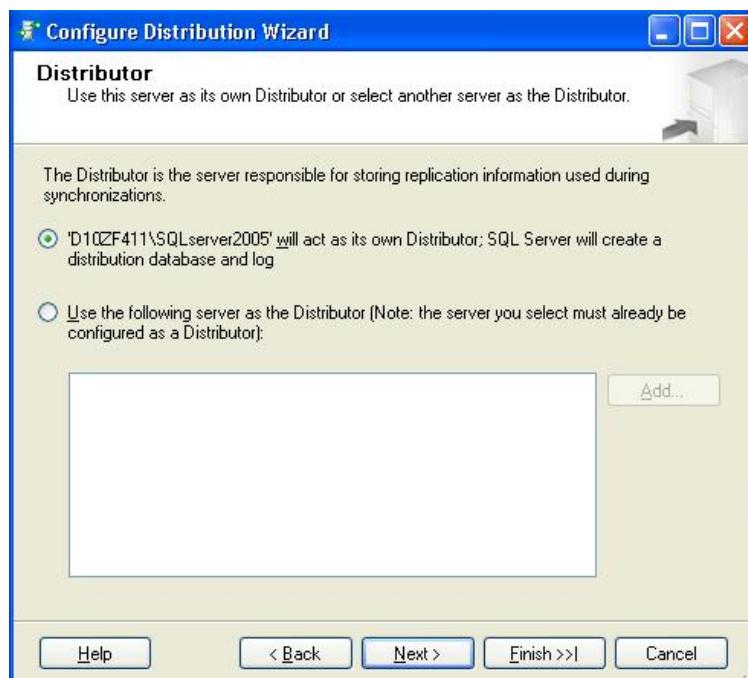
Configuring a Distributor

SQL Server 2005 introduced numerous welcome improvements to replication, not the least of which is shorter wizards. Following a wizard isn't difficult, but fewer wizard screens certainly

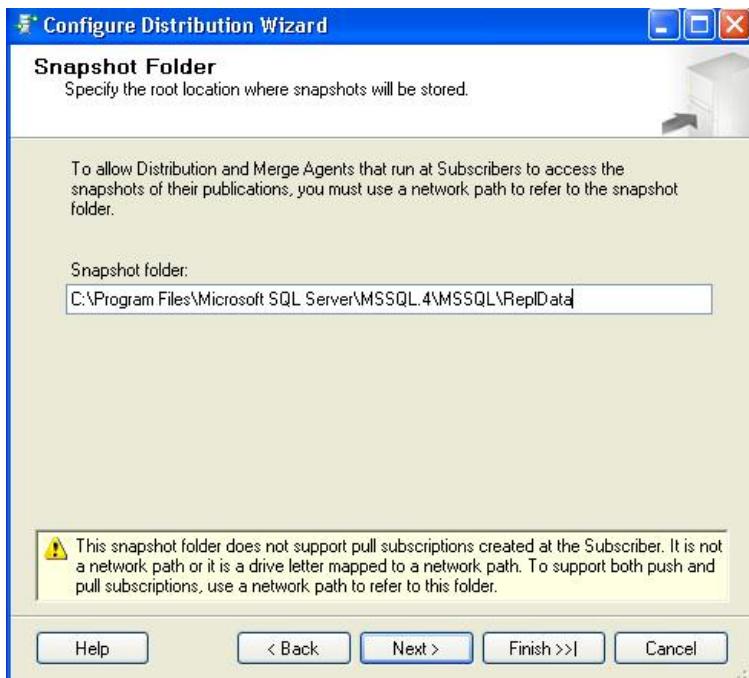
make replication setup quicker. As a rule, replication wizards in SQL Server 2005 are nearly 50% shorter than those in SQL Server 2000.

The first step in configuring replication is designating a server that will take care of storing and delivering replicated transactions—the distributor. A single server can act as a publisher, distributor, and a subscriber, all at the same time. However, in a realistic scenario you're likely to use two different servers as publisher and subscriber. Using a separate server as the distributor can help to reduce the load on the publisher.

To invoke the Configure Distribution Wizard, connect to an instance of SQL Server by using the SQL Server Management Studio (SSMS), navigate to the "replication" folder, right-click this folder, and choose Configure Distribution from the pop-up menu. Replication wizards are no longer modal; that is, you can continue working with SSMS while the wizard is active. The first screen of the wizard simply informs you of the tasks that this wizard can help you to accomplish. If you don't ever want to see this screen again, simply check the option to skip the introductory screen in the future. The next screen asks whether you want to use the local server or a different server as the distributor. Fig-1

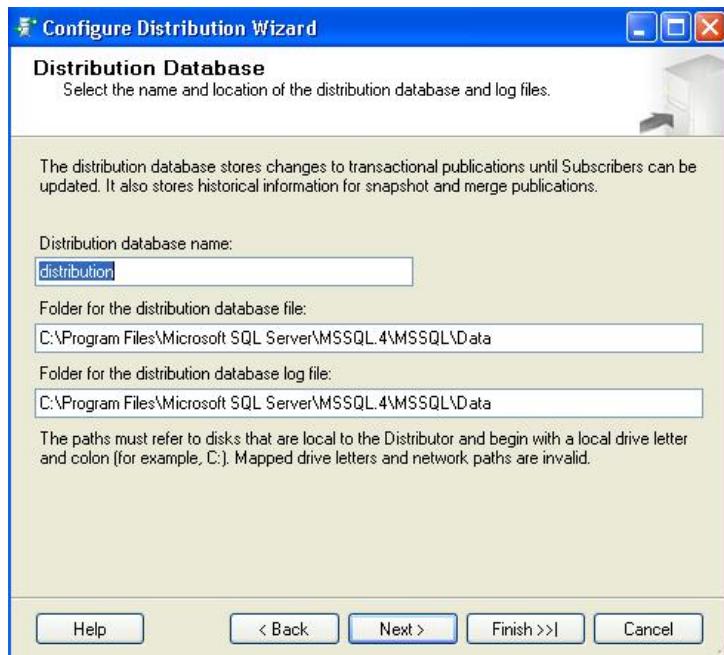


If you want to use a remote distributor, you must first run the Configure Distribution Wizard on that server. For this example, I'll use the same instance as both publisher and distributor. The next screen allows you to specify the snapshot folder where data and schema of the published database will be stored. By default, the snapshot folder is called ReplData and is created within the directory where the current SQL Server instance is installed.

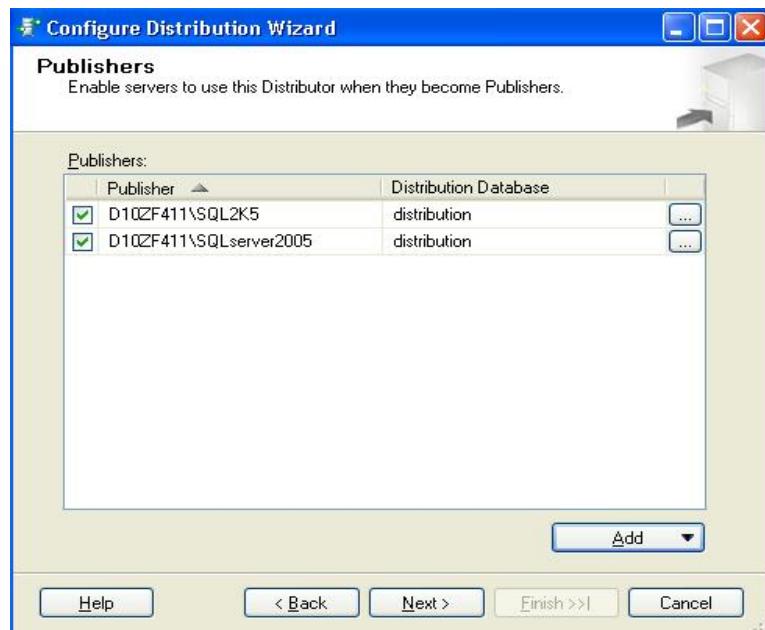


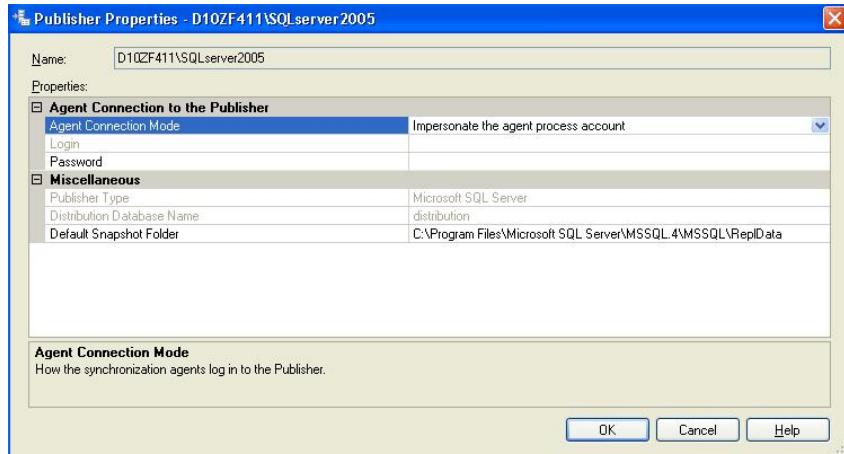
Notice the warning in the dialog box, indicating that the current directory doesn't support pull subscriptions. To use pull subscriptions, you need a network folder for storing snapshots. Because both publisher and subscriber instances of SQL Server in this example will reside on the same computer, I can safely disregard this message, and simply click Next.

The following screen allows for configuring the distribution database's name and the location for its data and transaction log files. By default, the distribution database is called distribution; you can modify the name if you have a compelling reason to do so. For example, if you have dozens or hundreds of publications, you might want to have multiple distribution databases, with descriptive names for each one. The wizard will use the default location for database and log files. You can configure the default location on the Database Settings tab in the Server Properties dialog box in SSMS (right-click the server and choose Properties to access the dialog box). Alternatively, you can change file locations in the wizard, Fig-3



The next screen enables servers to use the current distributor when they're configured as publishers (see Figure 4). This screen has a couple of interesting options. First, if you click the ellipsis (...) button next to a publisher, you'll get a dialog box that allows you to configure the log reader agent's security credentials as well as the snapshot folder for this publisher, as shown in Figure 5





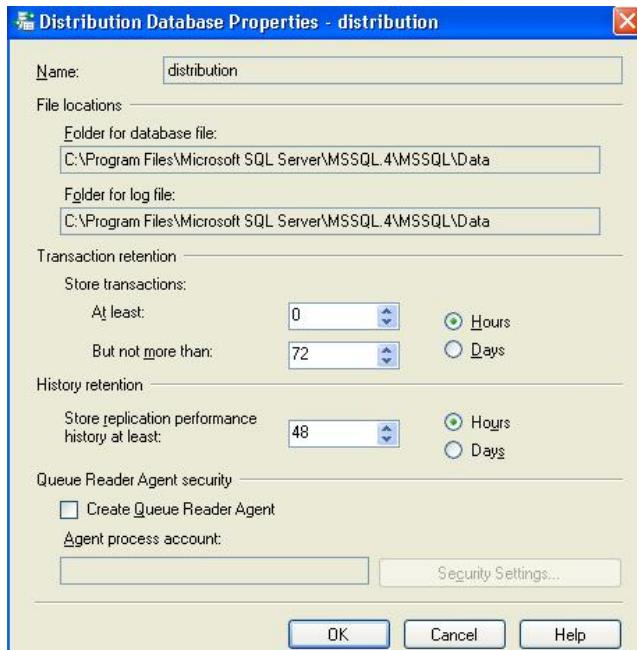
Second, the Add button allows you to add a SQL Server or Oracle publisher. This feature is worth your attention because using the distribution database for an Oracle publisher wasn't available in previous versions.

After you've enabled the publishers, you can set a password for remote publishers (see [Figure 6](#)). You must enter the same password twice. SQL Server 2005 allows the administrator to enforce password policies and password expiration. Hence, the wizard warns you that the password you enter for a remote publisher must meet the password policy requirements.



After you click Next on this screen, you can configure distribution right away, save the script for later execution, or perform both actions. If you choose to save the script, you'll be asked for the location where you want to save the file. At this point, the wizard presents a synopsis of the steps it's about to undertake; once you click Finish, the wizard will create the script for adding a distributor and/or save the script, depending on what you specified.

Once you've configured the distribution database, you can read or modify the distributor properties by right-clicking the replication folder and choosing Distributor Properties. The resulting dialog box has two pages—a "general" page and a "publishers" page. The "general" page allows you to view distribution database properties and modify settings for transaction retention and/or history retention (see Figure 7).



Notice that you're also allowed to create and configure a queue reader agent from this screen. The queue reader agent is beyond the scope of this article.

The "publishers" page of the Distribution Database Properties dialog box lets you add a publisher or change existing publishers' properties.

WARNING

The Configure Distribution Wizard adds a login called "distributor_admin" to your server. This login is a member of the sysadmin server role—meaning that it has no limitation as far as what it can do on the server. This is why it's absolutely imperative to create a strong password for connecting to the distributor.

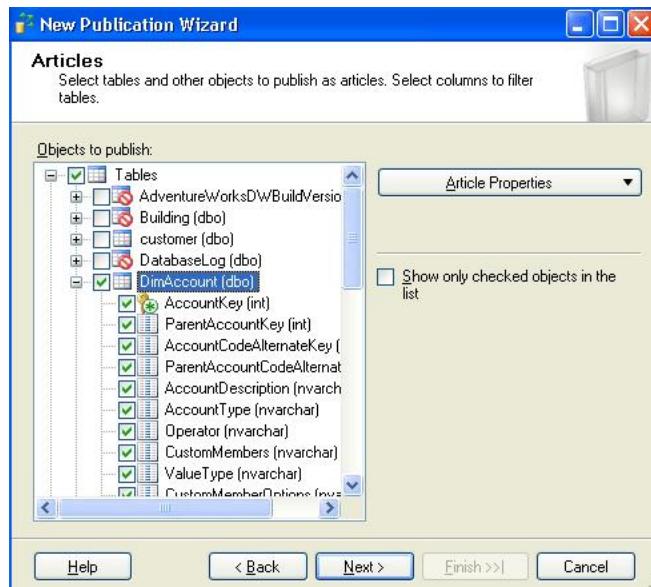
Creating a Publication

Once you've configured a distributor, you're ready to create publications. To invoke the publication wizard, right-click the local publications folder and choose New Publication from the pop-up menu. As with the Distribution Configuration Wizard, the first screen of this wizard is introductory in nature and can be skipped. The second screen allows you to choose the database in which you want to create a publication; for purposes of this article, I'll create a publication within the AdventureWorksDW database that can be created as part of SQL Server 2005 installation. After selecting the database, you must choose the publication type. The wizard offers the following options:

- Snapshot Publication
- Transactional Publication
- Transactional Publication with Updatable Subscriptions
- Merge Publication

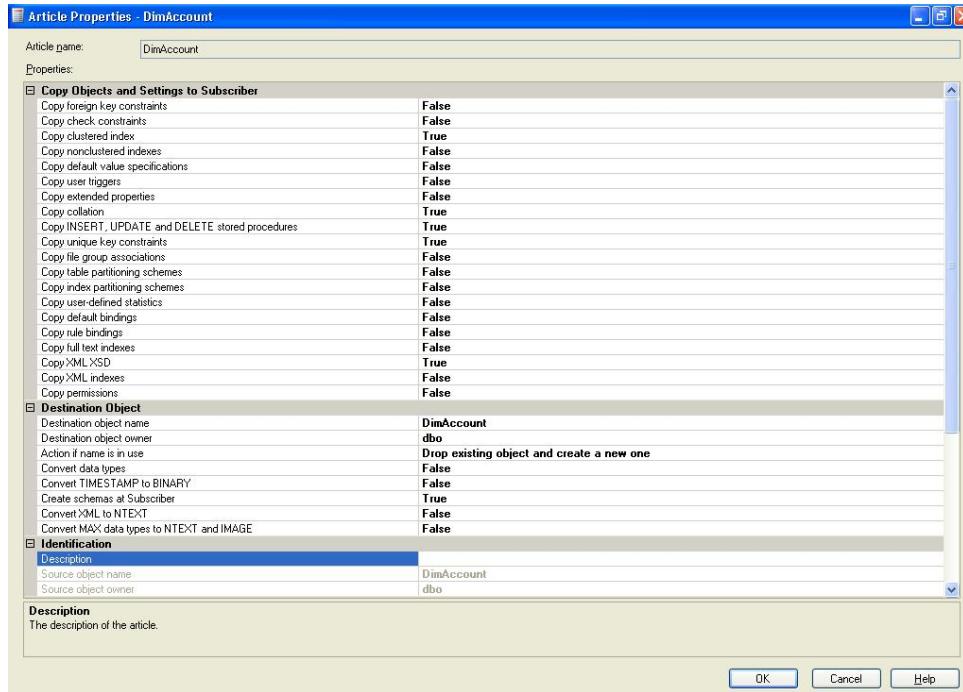
The wizard includes a brief description of each type of publication. I'll use the transactional publication for this example; refer to my earlier articles for more info about other publication types.

A transactional publication can contain one or more articles. An *article* can be a table, a view (including indexed views), a user-defined function, or a stored procedure. For this example, I'll replicate the dimAccount table from the AdventureWorksDW database. As shown in Figure 8, I can replicate all columns or a subset of all columns within a given table.



Replication has certain rules as far as which columns can be filtered. Transactional replication prohibits filtering primary-key columns. In addition, if your publication allows updateable subscriptions, you must replicate the msrepl_tran_version column (added by SQL Server when you create such publications). Further, publications that allow updateable subscriptions must replicate any column that doesn't allow nulls, doesn't have a predefined default, and isn't an identity column.

If you check the box *Show Only Checked Objects in the List*, the wizard limits the list of articles to only those that have been checked. The *Article Properties* button allows you to set properties for the highlighted article or for all table articles. As [Figure](#) shows, you can set a multitude of replication-related properties for each article.



Most properties you can set for table articles are self-explanatory; for example, the Copy Foreign Key Constraints option instructs the replication to include foreign key constraints when creating the table in the subscriber database.

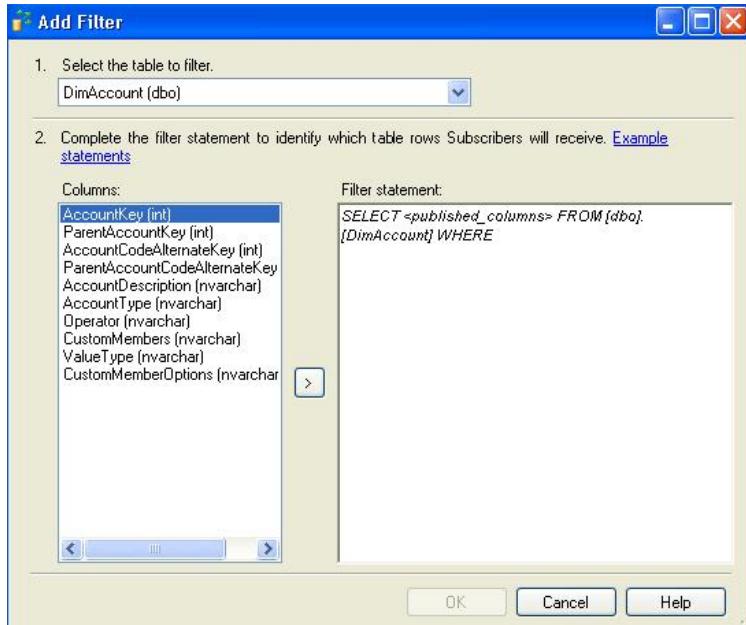
A few properties deserve additional consideration:

- **Destination Object Name, Destination Object Owner.** The destination table isn't required to have the same name or the same owner as the source object.
- **Convert Data Types.** This option automatically changes a user-defined data type to the base data type, because the user-defined data type might not exist on the subscriber(s).
- **Convert TIMESTAMP to BINARY.** When replicating a column with a TIMESTAMP data type, you can convert it to BINARY. The TIMESTAMP data type tracks the sequence of modifications; every time you change a data row, SQL Server will automatically change the value of the column with the TIMESTAMP data type. This is important because, if you're not careful, you might end up with different values in the column with the TIMESTAMP data type on the publisher and the subscriber.
- **Convert MAX Data Types to NTEXT and IMAGE.** This option translates VARCHAR(MAX) and VARBINARY(MAX) data types, which are new in SQL Server 2008, to respective data types supported in previous versions.
- **Convert XML to NTEXT.** Translates the new XML data type to NTEXT.
- Another option that wasn't available through wizards in previous versions of SQL Server is automatic identity range management. This option allows the database administrator

to set the ranges of valid values for the identity column in the publisher and subscriber databases. For example, we could assign values 1,000,000 and greater to the publisher and 1 to 1,000,000 to the subscriber. When the publisher database reaches the upper limit for the identity range, it will automatically assign a new range so that publisher and subscriber identity values don't overlap.

- The final group of options (not shown in Figure 9) determines how to replicate INSERT, UPDATE, and DELETE statements to the subscriber.

Once you've set the necessary properties for the article you want to replicate, you can add publication filters (see Figure 10). In previous versions of SQL Server, these filters were referred to as *horizontal filters*—you create them by supplying a WHERE clause to limit the number of published rows. As shown earlier, now you can filter the publication vertically by specifying which columns to publish.



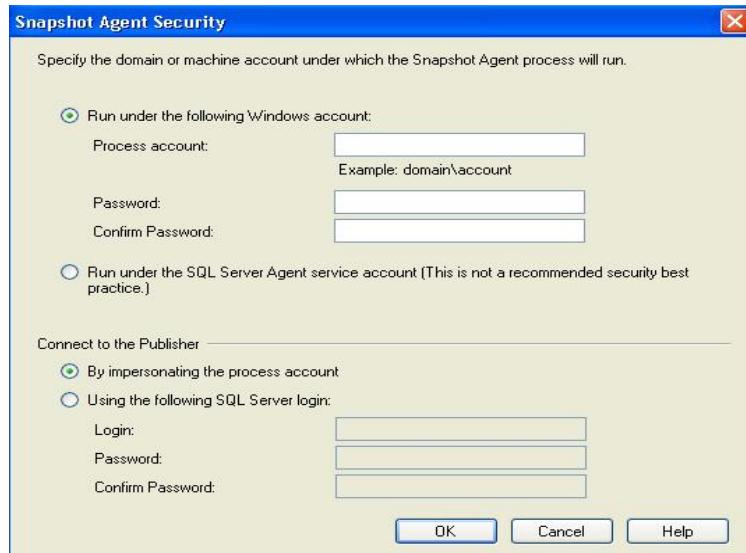
The next step is to create a snapshot and/or specify the snapshot agent's schedule, as shown in



The snapshot agent copies the schema and data of the replicated article(s) into the snapshot folder. If you click the Change button on this screen, you'll get the typical dialog box for creating job schedules; you can run the snapshot agents monthly, weekly, daily, or even multiple times per day.

Next you specify the security settings for the snapshot and log reader agents ([see Figure 12](#)). I'll discuss replication security in greater detail in a later article about transactional replication agents. For now, you just need to know that you can customize security for each agent or use different credentials for each.

The wizard next offers you the option to script the commands for creating the publication. Review the synopsis of the steps the wizard is about to undertake; then specify the publication name and click Finish to create the publication.

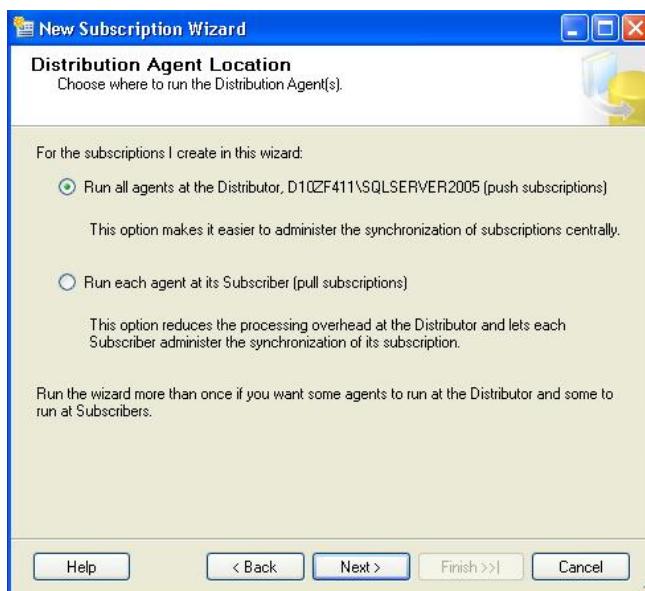


You can view the newly created publication's properties by expanding the local publications folder, right-clicking the publication, and choosing Properties from the pop-up menu. The properties dialog box has several pages, each of which has a specific purpose:

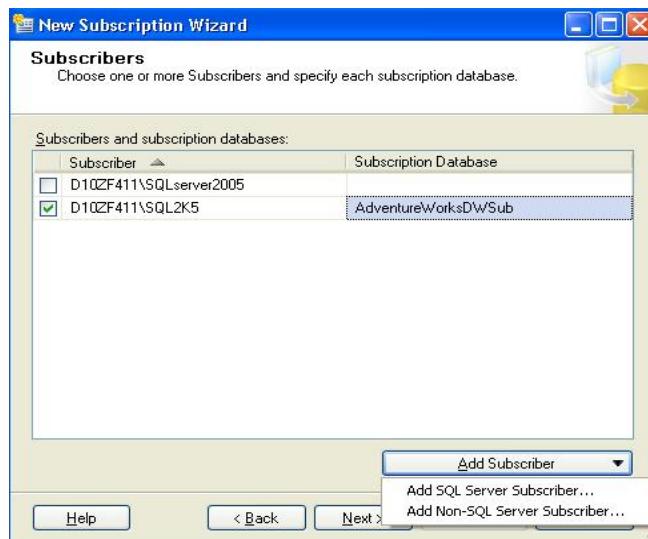
- **General.** Shows the publication's name, description, type, and the database on which the publication is based. You can modify subscription expiration options from this page.
- **Articles.** Lets you review the published articles, modify their properties, or add new articles to the publication.
- **Filter Rows.** Allows you to create horizontal filters for articles.
- **Snapshot.** Enables you to specify the snapshot folder location, snapshot format, or additional scripts to be executed before and after applying the snapshot.
- **Agent Security.** Controls security settings for the log reader and snapshot agents.
- **Subscription Options.** Provides a multitude of options for subscribers to the current publication. The following table describes the subscription options you can set through the Publication Properties dialog box. Note that several of these options are new in SQL Server 2005.

Creating Subscriptions

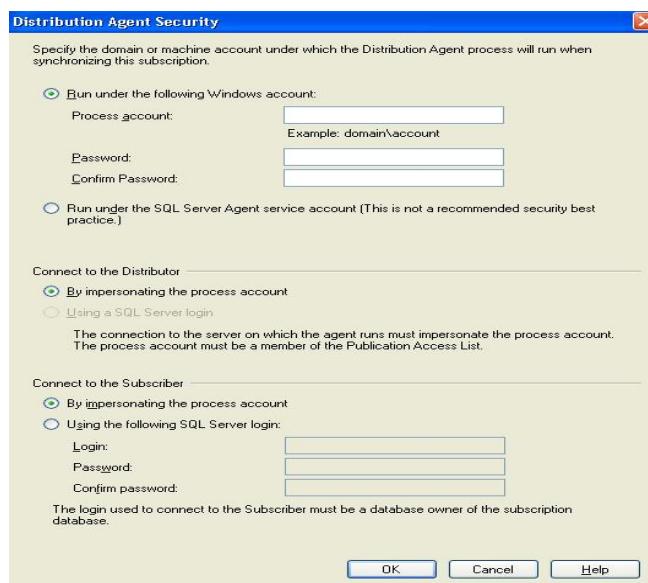
Unlike previous versions, of SQL Server 2005 allows you to use the same wizard to create either pull or push subscriptions. To invoke the new subscription wizard right-click the publication (or the local subscriptions folder) and choose New Subscriptions from the pop-up menu. After you get past the introductory screen, select the publication for which you want to create subscription(s). Next, indicate whether you want to use pull or push subscriptions (see Figure 13). Pull subscriptions reduce the load on the publisher, whereas push subscriptions make it easy to administer all subscriptions at the central location. For this example, I'll use push subscriptions, but the wizard screens are nearly identical for pull subscriptions.



Next you choose a subscribing server and database, as shown in [Figure 14](#). You can use an existing database or create a new database; if you choose to create a new database on the subscribing server, you'll get the typical dialog box for creating databases. More interestingly, note that the wizard allows you to use a non-SQL Server subscriber. You can choose either an Oracle or IBM DB2 subscriber for push subscriptions; only SQL Server subscribers are supported if using pull subscriptions.



After specifying the subscriber server and database, you need to configure distribution agent, keep in mind that you can either impersonate the SQL Server Agent or use a separate Windows login or SQL Server login for the distribution agent. For this example, I'll use the SQL Server Agent service account for running the distribution agent and for connecting to the subscriber.



Now it's time to define a synchronization schedule—how often you want the replicated transactions to be delivered to the subscriber(s). Replicating transactions *continuously* is the best option if you want to achieve minimal latency; however, this option requires more work on the publisher for push subscriptions and on the subscriber for pull subscriptions. *Scheduled delivery* is a good option if you want to minimize the load during business hours and deliver commands only at certain times each day. *On-demand delivery* can be a viable option if you want to synchronize your databases only occasionally.

After indicating the desired synchronization schedule, you can initialize the subscription database. During initialization, replication creates the published objects' schemas and copies data from the snapshot folder to the subscription database; in addition, the stored procedures used for replication are created in the subscriber database. In the dialog box, you can specify that you don't want to initialize the subscriptions—this option is useful if the schema and data already exist on the subscriber. Other options are to initialize subscriptions immediately or at first synchronization—that is, the first time the snapshot agent runs.



You're done specifying all the information that the wizard needs to create subscriptions. At this point, you have the option to script the subscription and/or to create subscriptions. The wizard allows you to review the summary of the steps it's about to undertake before you click the Finish button.

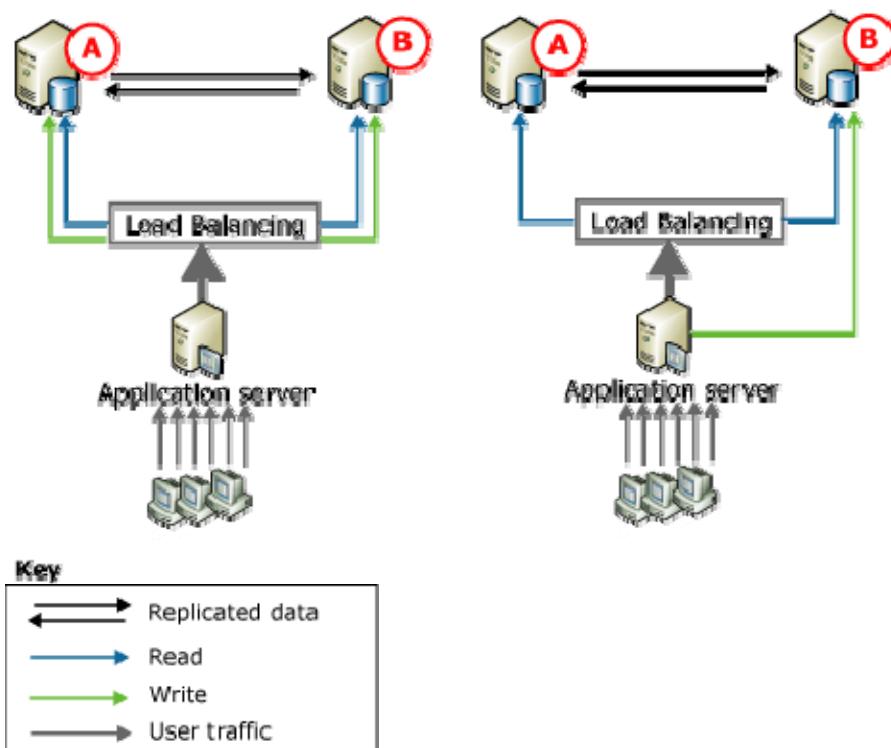
Peer To Peer Replication

Peer-to-peer replication provides a scale-out and high-availability solution by maintaining copies of data across multiple server instances, also referred to as *nodes*. Built on the foundation of transactional replication, peer-to-peer replication propagates transactionally consistent changes in near real-time. This enables applications that require scale-out of read operations to distribute the reads from clients across multiple nodes. Because data is maintained across the

nodes in near real-time, peer-to-peer replication provides data redundancy, which increases the availability of data.

With Peer to Peer replication the idea is that any subscriber is also a publisher, so data can move both ways such as bi-directional replication. With Peer to Peer replication there is a mechanism that knows when a transaction has been replicated, so the transaction does not continue to be replicated over and over again. The one area that still needs to be handled is if the same data is modified on multiple servers and therefore some type of conflict resolution may need to take place.

Below is a diagram that shows how Peer to Peer replication works. This diagram shows two different configurations: on the **left** side both nodes A and B are used for reading and writing and on the **right** side only node B is used for reading and writing where node A is used only for reading data. The application server, which could be a web server, determines which node is used for which component.



The following scenarios illustrate typical uses for peer-to-peer replication.

This model could be further extended to have a node C, D, etc... and the application server could determine which server is utilized for reading and/or writing. In most cases database servers are more read intensive than write intensive, so having more read only nodes would allow you to get additional I/O throughput on your database servers. If you also needed to

have more nodes handling writes, you could build this into your application servers to determine which writes are done to what servers.

With Peer to Peer replication when data updates take place the data is then replicated to all other Peers, so the offloading of read activities could be directed to any one of the nodes. From a failover solution, since all nodes act as both publishers and subscribers you can easily point your application server to any of the nodes to act as your primary database.

Case Study: How to Configure Peer-to-Peer replication in 2012

P2P replication is new to SQL server 2005 and though looks like complex in theory its one of the easiest to configure and maintain on long run. The following steps take you through the configuration process.

Configuring the distributor

Distributor has to be configured for all the nodes participating in the replication. Each node should have its own distributor. Once you click the configure distribution you will be taken to the welcome screen. Click next.



Next you will be asked for the details of the distribution database server and name. Confirm the detail and click next.

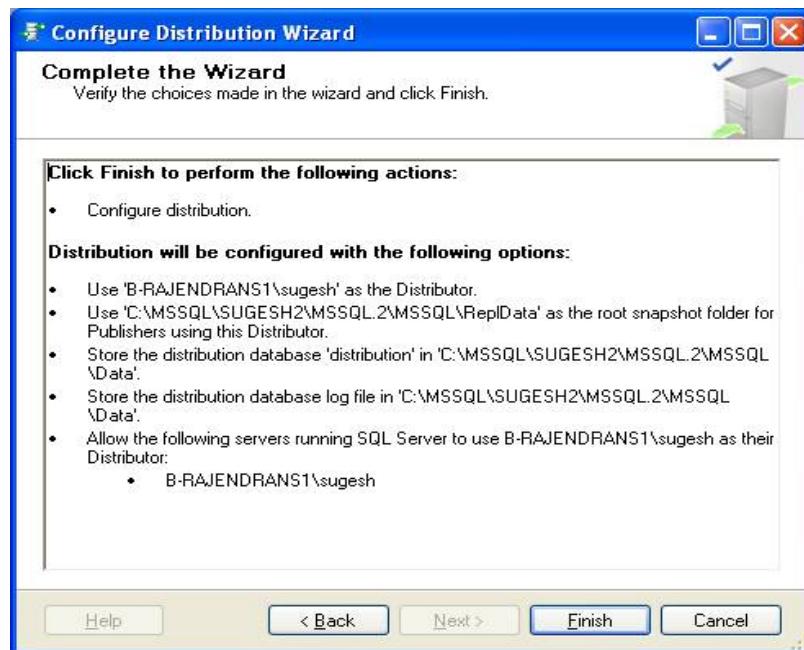
Next you will be asked for the shared folder where the snapshot will be placed. Confirm and Click next.

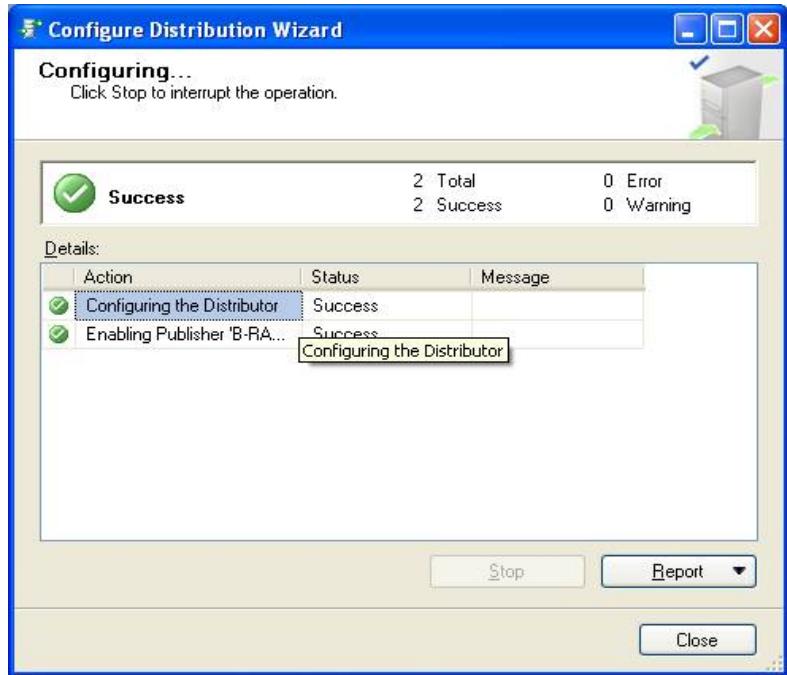


Next you will be asked to confirm the details of Distribution database click next after confirmation.



Once you are done with this, click finish so that the distributor will be configured for the server. Remember to follow these steps for all nodes participating in the replication. If you already have it configured the you are all set to start with the data replication.





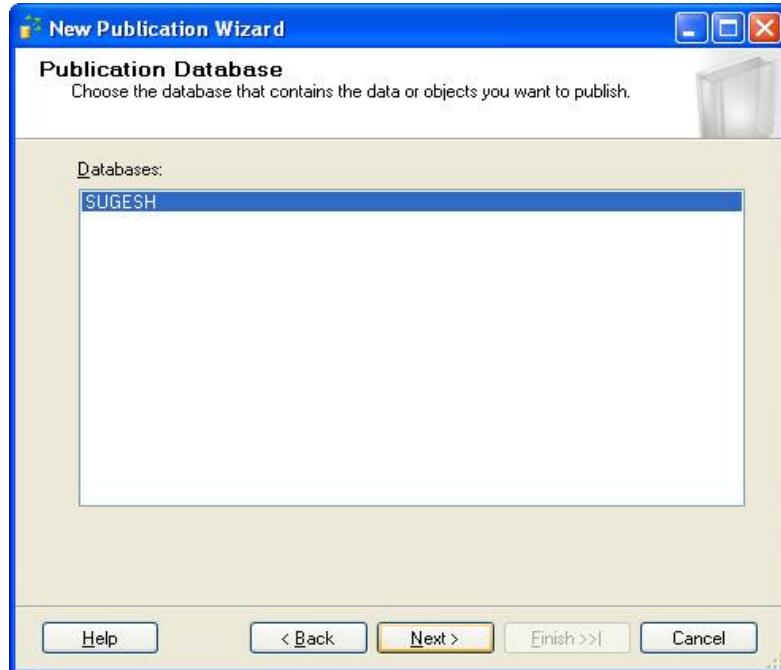
Configuring Publication database:

The next step involved is configuring the publication for the replication instance. Login to one of the node and create a new publication following the below steps.

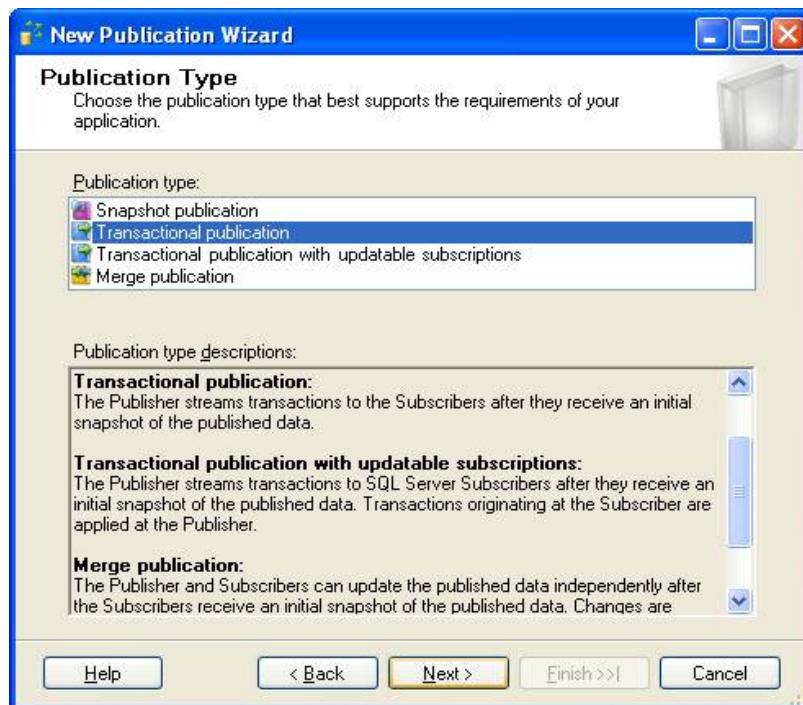
On the welcome screen, click next.



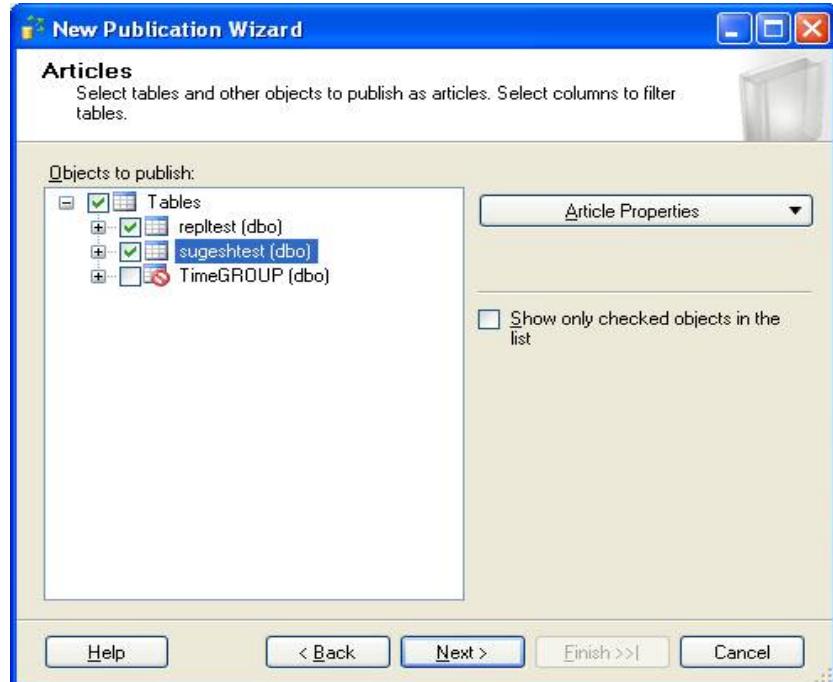
Confirm the database to be published and click next.



Click on transactional replication and click next.



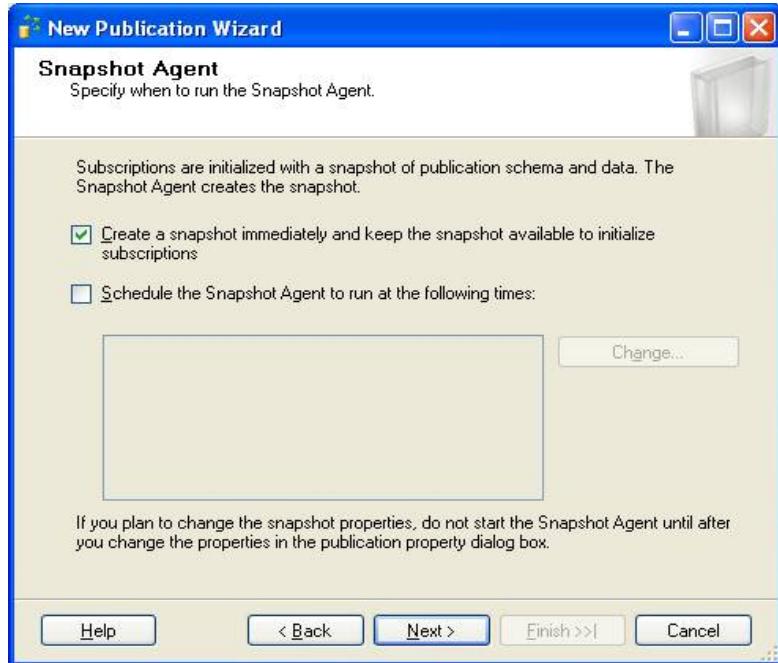
Select the articles to be published and click next. Remember that P2P replication is a kind of transactional replication you can add articles only having a primary key column.



You cannot filter rows in P2P replication so ignore the next screen clicking next.



Next screen is about creating the snapshot for the publication. Choose what you need and click next.



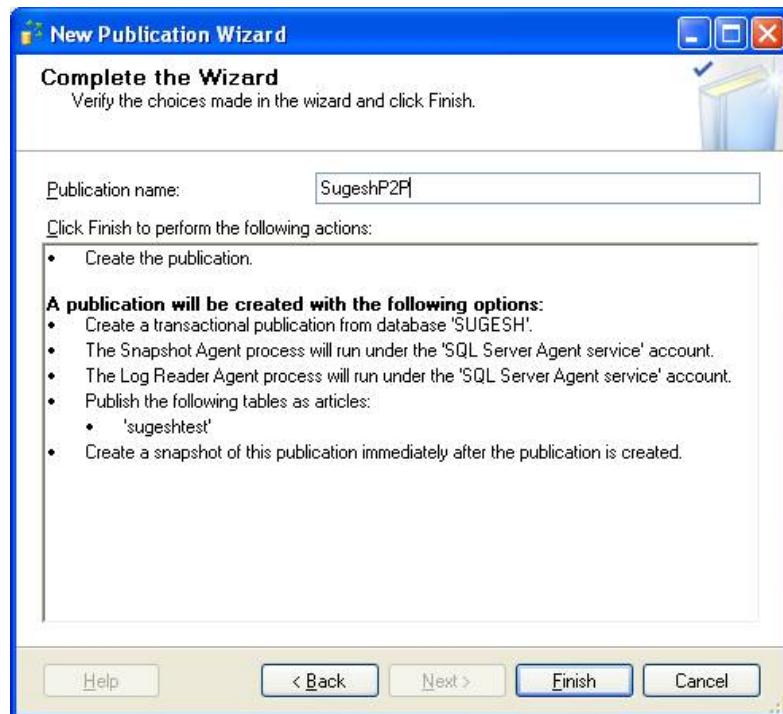
Configure the agent security for the publication and click next. This is the login which the replication agents will use to communicate within themselves.



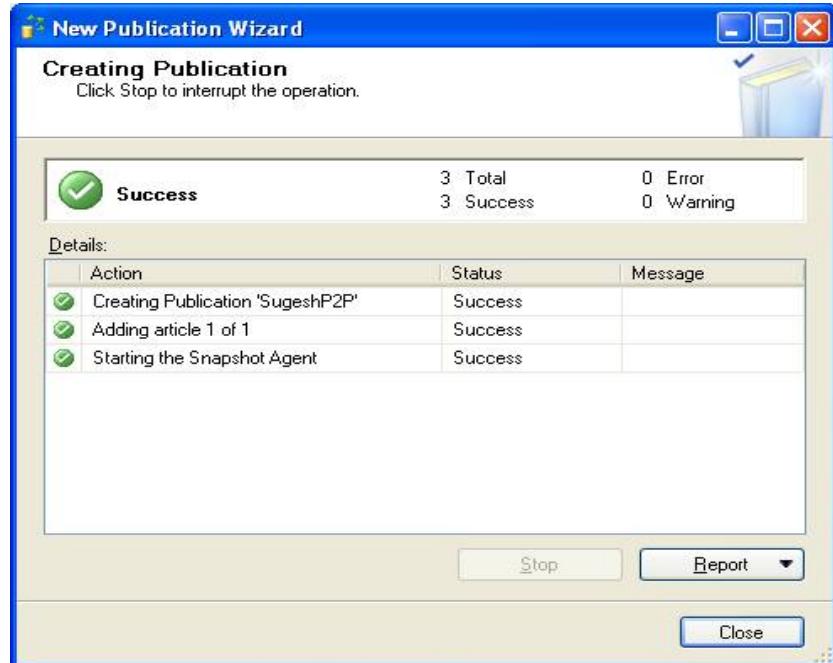
Click on the next screen for creating the publication.



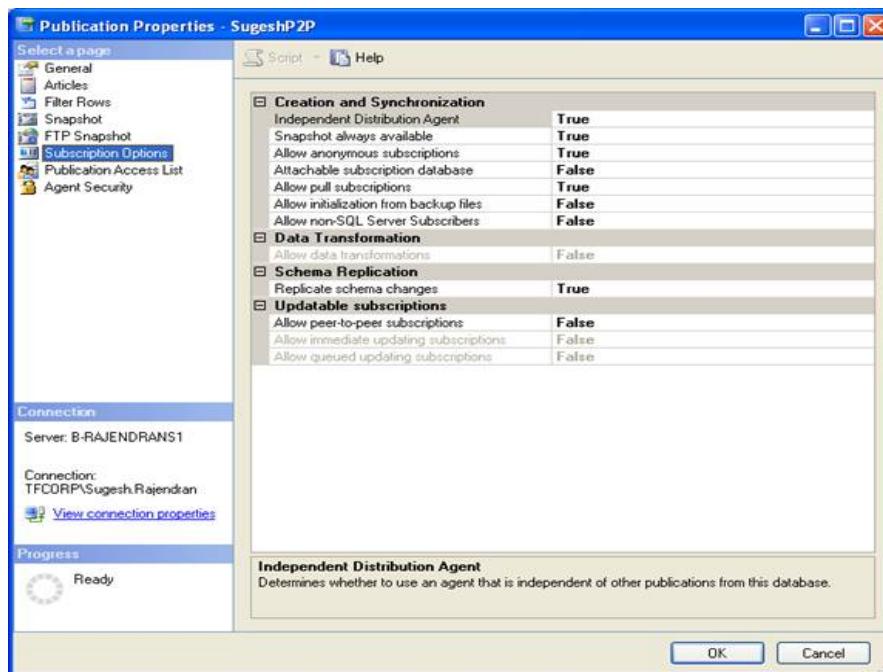
Enter a Name for the publication and click finish.

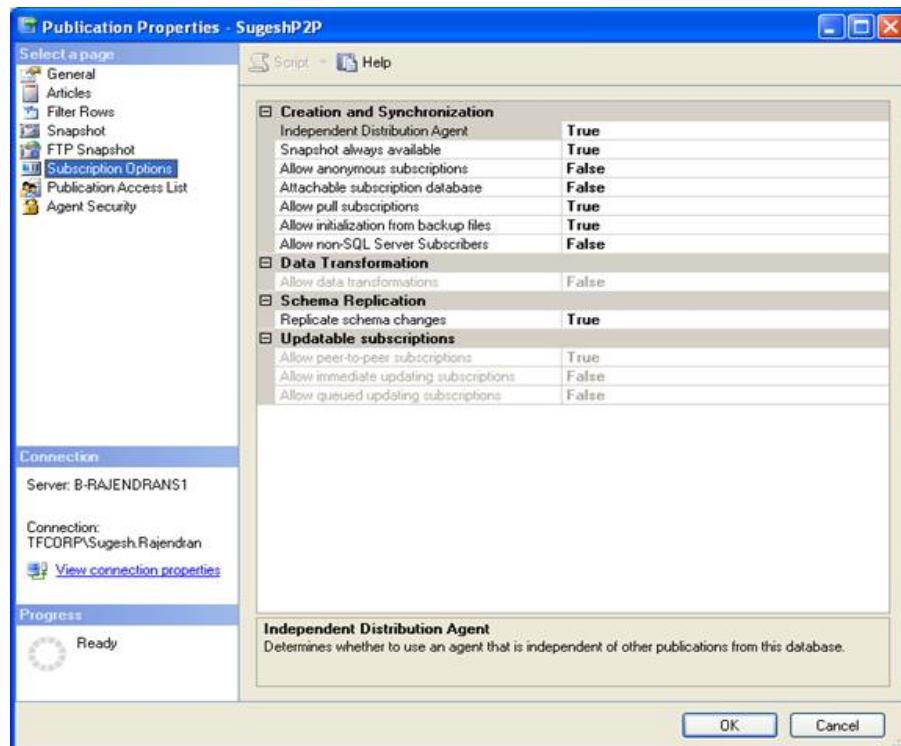


Once you see success in all the steps click close. You have created the publication for this replication instance. The next step is in configuring this publication for P2P replication.



Click on the properties of the publication. Move to the subscriptions options page. You will see an option Allow Peer-to-Peer replication. Change the value from False to True.





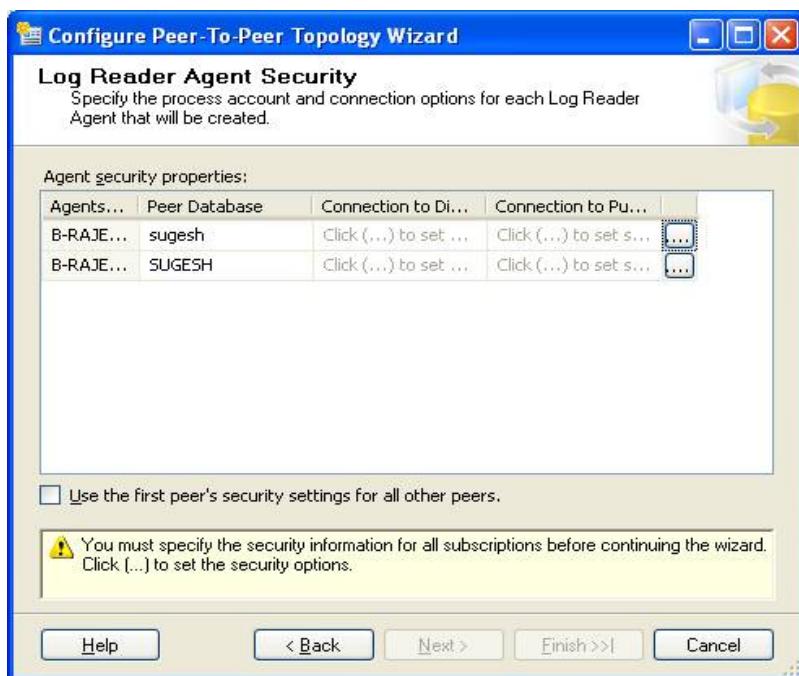
Then Right click the publication and you will see configure peer-to-peer replication. Click that to proceed. Click the publication for which you need to configure the replication instance and click next.



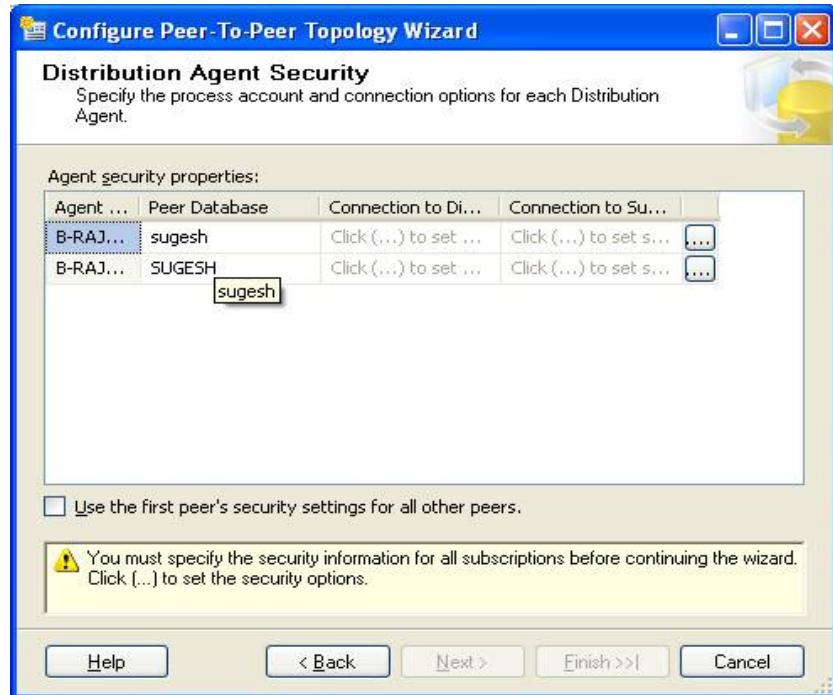
Add the peer instances and click next



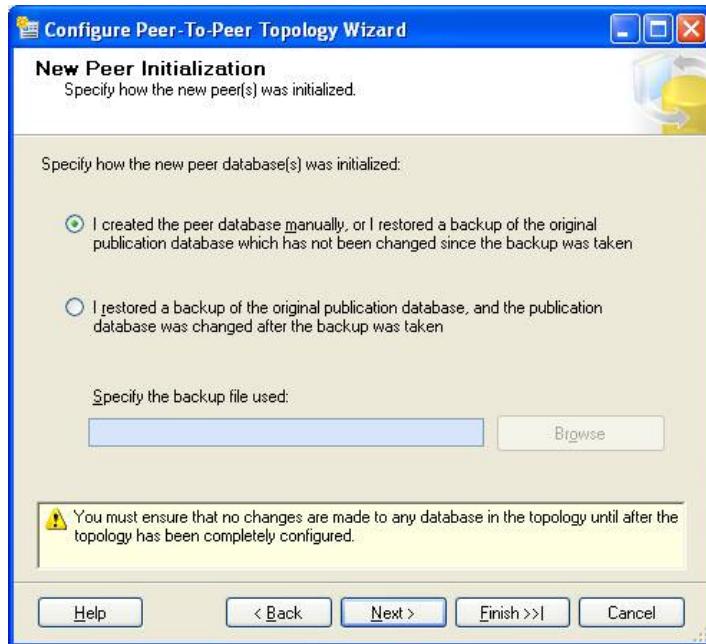
Configure the log reader agent security for the peers and click next.



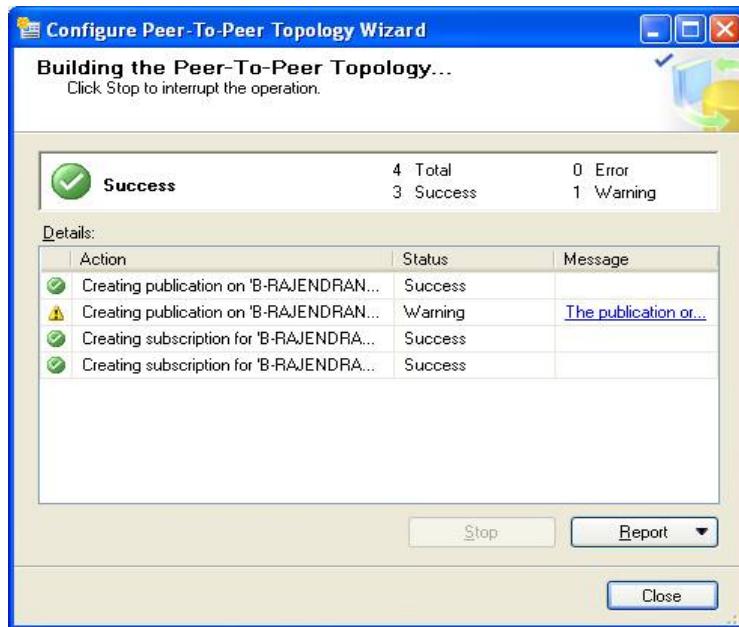
Configure the distributor agent security and click next.



Here you will need to provide the details of how you initialized the database in the peer. You can either create database manually or restore a database. In this case I have restored the database with no changes gone to the database in the publisher. Click next.



Click finish. Check for the errors and warnings in the results screen.



Once all steps have been configured successfully, you are done with configuring P2P replication.

Case study: Initialize from Backup for Transactional Replication

How to initialize the subscriber from backup

1. Create the publication using user interface Replication Wizard.
2. Under subscription options for the publication, set "**allow initialization from backup files**" to true
3. Create a new full backup of the publisher database. If you have existing full backup of the publisher database, you can still use that backup set but we have take a new log backup or differential backup of the publisher database and restore at the subscriber.
4. Replication UI does not allow the option to create the subscription to allow initialization from back. We have to TSQL when creating the subscription.

```
exec sp_addsubscription @publication = N'Repl2000', .....
@sync_type = N'initialize with backup',
@backupdevicetype='Disk',
@backupdevicename='C:\Repl2000_RestoreThis.bak'--this is the last backup used to restore on
the subscriber that was taken after the publication was created
go
```

```
exec sp_addpushsubscription_agent .....,  
go
```

5. This should create the subscription and the Distribution agent should be replicating the commands successfully.

Best Practices on Replication

- Replication needs should be clearly defined before creating a replication topology. Successful replication can be difficult and requires much pre-planning.
- Ideally, publishers, distributors, and subscribers should be on separate physical hardware.
- Create, document, and test a backup and restore strategy. Restoring replicated databases can be complex and requires much planning and practice.
- Script the replication topology as part of your disaster recovery plan so you can easily recreate your replication topology if needed.
- Use default replication settings, unless you can ensure that a non-default setting will actually improve replication performance or other issues. Be sure that you test all changes to ensure that they are as effective as you expect.
- Fully understand the implications of adding or dropping articles, changing publication properties, and changing schema on published databases, before making any of these changes.
- Periodically, validate data between publishers and subscribers.
- Regularly monitor replication processes and jobs to ensure they are working.
- Regularly monitor replication performance, and performance tune as necessary.
- Add alerts to all replication jobs so you are notified of any job failures.

Chapter – 19

SQL Server Clustering

Introduction

Microsoft® SQL Server® 2008 provides a solution for mission-critical applications that require the highest levels of availability. SQL Server 2008 failover clustering is part of the SQL Server high-availability technology toolset and is designed to help businesses meet their availability and uptime goals. This provides protection against both planned as well as unplanned downtime. When a server on one of the node fails SQL Server can continue serving request through other node(s).

SQL Server 2008 includes several changes to the implementation of SQL Server failover clustering, including an entirely new setup process and support for up to 16 nodes. This white paper discusses several new and existing concepts for SQL Server 2008 failover clustering, such as architecture, implementation, administration, and troubleshooting.

Overview of Windows Server Failover Clustering

SQL Server 2008 failover clustering is built upon an established Windows Server® failover cluster. A Windows Server failover cluster aims to provide high availability for services or applications that run within the failover cluster. It contains a group of independent servers that work together to increase the availability of applications and services. Failover clustering can protect against hardware and software failures by *failing over* resources from one server (or *cluster node*) to another as required. Failover is the process of taking a clustered service or application offline on one node and bringing it back online on another node. This process is typically transparent to the users, who should experience a minimal disruption of service when a failover occurs.

Other types of cluster configurations such as Windows® Compute Cluster Server 2003, which Windows® HPC Server 2008 succeeded, and Windows Network Load Balancing (NLB) clusters exist. However, all of them have different implementations and are not the Windows Server failover clustering for which SQL Server 2008 failover clustering is designed. Windows failover cluster architecture provides redundancy and the ability to detect certain failures. It compensates for these failures by moving highly available resources over to redundant working components. This architecture is not suitable for scenarios such as load balancing and its implementations differ from the HPC and NLB cluster solutions.

A failover cluster contains one or more clustered servers (called nodes) and a configuration of shared cluster disks that are set up for use within the cluster. The failover cluster also enlists at least two networks for communication, at least one public and one private. The private network should be configured for internal cluster communication only. The public network, however, is primarily for clients to connect to the applications that are running within the cluster. The cluster nodes communicate with one another over the private network by using *heartbeats* (periodic health detection signals that enable them to assess the status of other nodes in the cluster).

Other failover cluster components are:

- **Cluster service.** The essential software component that controls all aspects of server cluster or failover cluster operations. Each node in a failover cluster or server cluster runs one instance of the Cluster service.
- **Quorum.** The number of elements that must be online for a given failover cluster to continue running. The relevant elements in this context are nodes, a witness/quorum disk, or, in some cases, a witness file share. Each element included in the quorum, except a witness file share, contains a copy of the cluster configuration. The Cluster service works to keep all copies of the cluster configuration synchronized at all times.

Within the cluster, disks, IP addresses, network names, and applications such as SQL Server are resources that can be established as highly available and provide failover capabilities. These resources are designed to be cluster-aware through a resource dynamic-link library (DLL). The DLL contains the resource-specific code that is necessary to provide high availability for one or more resource types. Each resource type exposes two types of health detection checks:

- **LooksAlive** is a lightweight, more frequent check.
- **IsAlive** is run less frequently to do a deeper check that the service is available.

Resources that are related to a specific application instance are installed into a Windows Server failover cluster group, which is known as a resource group.

Note: In the Failover Cluster Management tool in Windows Server 2008, the **Resource Group** is called **Services and Applications**. These terms are used interchangeably throughout this document.

The resource group is the unit of failover, which means that all resources within the same group will fail over as one. At any given time, only one node in the cluster can own each resource group and the resources that it contains. See the “Topology and Architecture of a SQL Server Cluster” section below for more information about resources and resource groups for SQL Server.

SQL Server 2008 failover clustering can be configured on Windows Server 2003 and 2008

- **The cluster validation wizard.** In Windows Server 2003, the cluster hardware solution was validated manually via the Windows Server Catalog. In Windows Server 2008, the cluster validation wizard provides this functionality and validates that the cluster environment is compatible with Windows Server 2008 failover clustering prior to setup.
- **Service security IDs (SIDs).** If you are creating a new SQL Server 2008 failover cluster on Windows Server 2008, you can now bypass the use of domain groups by designating service SIDs during the installation. Service SID functionality was introduced in Windows Vista and Windows Server 2008, and allows the provisioning of access control lists (ACLs) to server resources and permissions directly to a Windows Server service. During the installation of a SQL Server failover cluster, in the **Cluster Security Policy** dialog box, you still have the option to use domain groups. However, for SQL Server 2008 on Windows Server 2008, the recommended

choice is to select **Use service SIDs**. This enables you to bypass provisioning of domain groups and associated service account membership additions prior to installation.

- **Four quorum modes.** Windows Server 2008 failover clustering now supports four quorum modes. To provide maximum availability, the cluster can automatically adjust the quorum mode as nodes are added or removed from the cluster. The four quorum modes are:
 1. **Node Majority.** Each node that is available and in communication can vote. The cluster functions only with a majority of the votes, that is, more than half. This is similar to Majority Node Set (MNS) in Windows Server 2003. This selection would be preferred with an odd number of nodes and if no shared disk/storage is provided.
 2. **Node and Disk Majority.** Each node plus a designated disk in the cluster storage (the "disk witness") can vote whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half. This is the recommended configuration with an even number of nodes and for when shared storage is available.
 3. **Node and File Share Majority.** Each node plus a designated file share created by the administrator (the "file share witness") can vote whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half. This would commonly be implemented in a geographically dispersed failover cluster environment.
 4. **No Majority: Disk Only.** The cluster has quorum if one node is available and in communication with a specific disk in the cluster storage. Only the nodes that are also in communication with that disk can join the cluster. This is equivalent to the quorum disk in Windows Server 2003. The disk is a single point of failure, so only select scenarios should implement this quorum mode.

When you install a Windows Server 2008 cluster, a recommended default quorum mode is selected during the installation. For example, if the cluster has an even number of nodes, the **Node and Disk Majority** quorum mode selected by default; if the cluster has an odd number of nodes, the **Node Majority** quorum mode will be selected. The user is not prompted to select a specific type of quorum mode during the installation. Furthermore, when nodes are added and removed from the cluster, the quorum mode can change automatically. This may surprise users who are accustomed to clusters in Windows Server 2003.

When you set up a cluster, it is important to understand the implications of each quorum model and the number of failures (node and/or disk) that a quorum mode can tolerate. You must also evaluate the requirements of the application to select the right quorum mode.

To maintain the uptime of a system, it is important to align the operational practices with the technology that is used to achieve high availability. For example, if your requirement is to restart two nodes in a three-node cluster, and still be available during that time, it may make sense for you to use the **No Majority: Disk Only** quorum mode. However, make sure that the disk quorum uses highly reliable storage (for example, a mirrored disk on a fully redundant storage area network (SAN)).

Topology and Architecture of a SQL Server Cluster

A SQL Server failover cluster instance that is installed on a Windows Server failover cluster is similar to a stand-alone SQL Server instance that does not use Windows Server failover clustering in terms of the way it runs and appears externally to clients. The key difference is that SQL Server failover clustering

leverages the Windows Server failover clustering platform to monitor and provide high availability of the SQL Server instance across redundant nodes. This section describes the SQL Server failover cluster resources and resource groups, in addition to common configurations.

Failover Cluster Resource Group

A SQL Server failover cluster instance has two parts:

- The local binaries and components that are installed on each possible owner node.
- The shared resources in the resource group that fail over between each possible owner node.

The local binaries and components on each node are similar to those for a stand-alone SQL Server instance. However, at startup, instead of the service being started either automatically or manually by the user, the Windows Server failover cluster service monitors and manages the instance.

Each SQL Server failover cluster instance has a resource group that contains the following set of resources:

- Network Name.
- IP Address.
- One or more shared disks.
- SQL Server Database Engine service.
- SQL Server Agent service.
- SQL Server Analysis Services service, if installed.
- One file share resource, if the FILESTREAM feature is installed.

The network name and IP address resources provide a single identifier that clients can use to connect to the SQL Server instance regardless of which node is currently serving the SQL Server failover cluster instance. When the resource group fails over, these resources register the network name and IP addresses to redirect to the new node that is serving the SQL Server failover cluster instance. This process can be transparent to clients who do not need to change the name or IP address that they are using to connect to SQL Server.

A SQL Server shared disk resource contains all of the system and user data including databases, logs, FILESTREAM files, and integrated full-text search files for the SQL Server failover cluster instance. When a failover occurs, the disks are mounted on a new node. When the SQL Server instance is started on the new node, it goes through recovery as part of the SQL Server startup and maintains access to the same database files that existed when it was running on the previous node.

The shared disks are the single point of failure for a failover cluster instance. Windows Server and SQL Server failover clustering provides redundancy for machines, operating systems, and SQL Server binaries, but requires reliable storage to assure availability of the shared storage. The shared disk resource type can either be the built-in shared disk resource or a custom shared disk resource provided by your storage vendor for more advanced storage configurations such as Storage Area Network (SAN) replication technologies.

The SQL Server, SQL Server Agent, and Analysis Services resources are responsible for bringing their respective services online and offline on each node. The Windows Server failover cluster directs this process in response to health detection messages that relate to each resource.

The SQL Server and SQL Server Agent resources are part of any SQL Server failover cluster's resource group where the Database Engine feature is installed. The SQL Server Analysis Services resource is a part of any SQL Server failover cluster's resource group where Analysis Services is installed. Although it is possible to install both Database Engine and Analysis Services in the same resource group, it is generally recommended to install them as separate instances. Analysis Services is not dependent on SQL Server, and it should be installed to a separate resource group for maximum availability and performance.

Health Detection

A SQL Server failover cluster has two levels of health detection:

- The Windows failover cluster detects that the nodes are online and able to communicate
- Each resource also provides its own specific health detection

SQL Server and Analysis Services rely on the built-in Windows Server failover clustering resources for network name and IP addresses which perform health checks to ensure the network name and IP address are online and properly registered. If the network name or IP address resource is offline, the SQL Server or Analysis Services resources that depend on them will also be in an offline state.

Shared disk health checks are built in to either the Windows failover cluster resource checks or the resource checks provided by a third-party cluster-aware storage vendor. In both cases, these resources will verify that the disk is attached and available.

For the SQL Server resource, the **LooksAlive** check (also referred to as **Basic resource health check** in Windows Server 2008) will verify that the SQL Server service is running on the online node every 5 seconds by default. If the **LooksAlive** check fails, the Windows Server cluster service performs an **IsAlive** check to confirm the failure.

The **IsAlive** check (also referred to as **Thorough resource health check** in Windows Server 2008) runs every 60 seconds and verifies the cached result of an internal **IsAlive** process in the SQL Server resource DLL.

An internal process in the SQL Server resource DLL handles all connection and execution of the underlying **IsAlive** logic. It contains extensive retry logic to verify any connectivity or execution failures. Upon successful connection to the SQL Server, the internal **IsAlive** check executes `SELECT @@SERVERNAME` every 60 seconds to determine whether SQL Server is online. If this query fails, the internal process runs additional retry logic to avoid stress-related failures. If the retry logic also fails, the internal process shuts down the SQL Server service and updates a cached result that causes the next **LooksAlive** and **IsAlive** checks to fail.

The SQL Server resource DLL is loaded by the Windows cluster service when the SQL Server resource is started. Therefore, the calls that the resource DLL makes are made in the context of the Cluster service. Consequently, the Cluster service must have access to the SQL Server's **master** database. It is unnecessary to add the Cluster service account or the `BUILTIN\Administrators` group to the **sysadmin**

role. However, the Cluster service account must be a user in the master database which has permissions to execute SELECT @@SERVERNAME. By default, the **public** role has this permission and the **guest** role is enabled for **master**, so simply adding the Cluster service account as a login with no additional permissions is usually sufficient. The SQL Server 2008 installation automatically adds the Cluster service startup account (or the service SID for the Cluster service) as a login during setup.

The Analysis Services service is installed as a generic service and its state (**LooksAlive/ IsAlive**) is determined by using standard logic for generic clustered applications.

Resource and Resource Group Properties

The SQL Server resource group and each resource within the resource group expose their failover properties through the Windows Server Failover Cluster Management Microsoft Management Console (MMC) snap-in. In addition, they expose their failover properties through cluster application programming interfaces (APIs) and the cluster.exe command-line tool.

Resource Properties

Each resource may expose several types of properties through Windows Server Failover Cluster Management. For Windows Server 2008, these include:

- **General.** The basic properties for this resource type.
- **Dependencies.** These define what other resources in the group must be online before a particular resource can be brought online.
- **Policies.** These define how the resource responds to a failure and what impact a failure of this resource has on the entire resource group.
- **Advanced policies.** These policies control the possible owners and the health-check intervals.
- **Permissions.** These define share and file permissions for a file-share resource. For example, SQL Server Setup will install a file-share cluster resource to support the FILESTREAM feature, if it is enabled.
- **Registry replication.** These are the registry keys that are replicated across all nodes via the checkpoint process.
- **Properties.** These are custom properties that are defined for the resource type.

For Windows Server 2003, a subset of the list above is included.

Each resource has a set of possible owners, which are the nodes that the resource is allowed to run on. In most cases, you should not change the possible owners manually. However, one exception to this is during the execution of a rolling update where ownership should be changed (see the "Rolling Updates" section below for more information). Otherwise, SQL Server Setup will handle adding and removing possible owners itself during installation, node addition, node removal, and major version upgrades.

Resource dependencies enable you to customize what the resource depends on. Windows Server failover clustering ensures the correct dependency order of startup and shutdown, and terminates any resource that is dependent on a resource that fails or is offline. SQL Server and SQL Server Analysis Services are dependent on the shared disk resources and the network name resource. The network name resource is

dependent on the IP address resource, and SQL Server Agent is dependent on the SQL Server Database Engine resource.

Windows Server 2008 adds the ability to specify **OR** dependencies in addition to **AND** dependencies. SQL Server 2008 and earlier versions do not support **OR** dependencies between IP addresses, which is necessary to enable nodes to run on different subnets.

Each resource also enables you to configure failover policies that determine:

- Possible owners. All resources within the same group should have the same possible owners.
- Whether the resource should attempt to restart if it fails.
- The maximum number of restarts to attempt on a given node within the specified period.
- Whether failure of this resource should cause the entire resource group to fail over to another node.
- Pending time-out (the time-out value to let the resource change states).
- The **LooksAlive** check (a basic resource health-check interval).
- The **IsAlive** check (a thorough resource health-check interval).
Note: Changing this interval does not affect the standard detection, which runs every 60 seconds. It does, however, change the interval at which the Windows Server Failover Cluster service checks the cached result of the SQL Server resource DLL. It is not recommended to change the interval from the default selection.

SQL Server 2008 uses the default policy of attempting to restart once on the same node on the first failure within a 15-minute interval. If a second failure occurs within the 15-minute interval, the entire resource group fails over. Each resource is also configured to affect the group if a restart on the node is unsuccessful. Each resource in the SQL Server resource group should be evaluated to ensure that its failover policy aligns with business requirements. Some customers may decide to disable the "**Affect the group**" setting (also referred to as "**if restart is unsuccessful, fail over all resources in this service or application**" in Windows Server 2008) for resources such as the ones listed below, if they do not want a failure of these resources to cause a failover, resulting in downtime, for SQL Server.

- The SQL Server Agent resource.
- The FILESTREAM file share.
- Microsoft Distributed Transaction Coordinator (MSDTC), if it is installed to the SQL Server resource group.
- SQL Server Analysis Services if it is installed in the same failover cluster instance as the Database Engine.

The pending time-out generally remains at the default value of three minutes; it is not usually necessary to change this value.

All resources have a default **LooksAlive** interval of five seconds. You can alter this value if you want to check whether the service looks alive more frequently. It is recommended to leave the default setting for this value.

SQL Server uses an internal process in the custom resource DLL to manage **IsAlive** logic for SQL Server failover clusters. The internal **IsAlive** logic maintains cached values to indicate the current state of the

SQL Server service. The internal timings for this logic cannot be changed. Changing the resource properties for the SQL Server resource within the Cluster Administrator changes the frequency that the Cluster service checks the cached results of the SQL Server resource DLL.

The SQL Server resource exposes custom SQL Server-specific properties for troubleshooting. In Windows Server 2008, these are visible on the **Properties** tab. In Windows Server 2003, you can also view these through the cluster.exe command:

```
cluster.exe RESOURCE [SQL Engine Resource Name] /PRIVPROPERTIES
```

For additional information about these properties see the "Properties for the SQL Server Resource" section in the "Troubleshooting SQL Server Failover Clusters" section of this paper.

Resource Group Properties

Each resource group exposes the following configurable properties:

- Preferred owners order.
- **AntiAffinityClassNames** (this property is not exposed via the Cluster Administrator graphical user interface (GUI)).
- Automatic fallback policies.
- Maximum number of retries or failures to attempt within a specified period.

You can use the **preferred owners** setting to specify nodes that are more desirable to run the application on. If you select multiple preferred owners, you can also configure the order of preference. On clusters that have more than two nodes, you should set your preferred owners list to provide a more balanced failover across the cluster.

The **AntiAffinityClassNames** setting enables the user to specify applications that they do not want to run on the same node. The property consists of zero or more arbitrary user-defined strings. Groups that have string names in common are said to be "anti-affined" from each other, which means that they should not be hosted on the same node if possible.

The resource groups also enable you to configure a policy to fail back to a specific "most preferred" node. You can configure the policy to attempt failback immediately after the node is available again, or within a maintenance range period, which is specified in hours. By default, automatic failback is not configured for SQL Server 2008. When you configure failback, using immediate failback can result in SQL Server failover that has less granular control than when allowing an administrator to schedule when to move the group.

You can also specify the maximum number of failover attempts or restarts for each resource group within a specified time before leaving the resource group in a failed state. By default, the SQL Server 2008 resource group is configured to tolerate two failures every six hours.

Windows Server 2012 Failover Clustering

Failover clusters provide high availability and scalability to many server workloads. These include server applications such as Microsoft Exchange Server, Hyper-V, Microsoft SQL Server, and file servers. The server applications can run on physical servers or virtual machines. In a failover cluster, if one or more of the clustered servers (nodes) fails, other nodes begin to provide service (a process known as failover). In SQL Server 2012 DBA

addition, the clustered roles (formerly called clustered services and applications) are proactively monitored to verify that they are working properly. If they are not working, they restart or move to another node. Failover clusters also provide Cluster Shared Volume (CSV) functionality that provides a consistent, distributed namespace that clustered roles can use to access shared storage from all nodes.

Scalability – With Windows Server 2012 you will be able to have the industry leading most scalable private cloud, with Failover Clustering having four times the scale over Windows Server 2008 R2 Failover Clustering. There is now support for 64-nodes in a single cluster, as well as 4,000 virtual machines running on a cluster.

Manageability – One of the major themes in Windows Server 2012 is multi-machine management. You will see a new Server Manager which will enable managing your private cloud and along with cluster integration. To manage a cluster of this scale you will see new management paradigm's in the Failover Cluster Manager snap-in to search, sort, and filter views in the Failover Cluster Manager snap-in. This will deliver a highly scalable easy to manage platform.

VM Mobility – In Windows Server 2012 a virtual machine can seamless move anywhere in your datacenter. You will be able to migrate VMs from one cluster to another, and between clusters and stand-alone hosts. Virtualization and high availability go hand-in-hand, so you will see tight integration with all the new Hyper-V features on a cluster. This will give you incredible flexibility and allow you to rethink your cluster deployment models.

Monitoring Applications in your Private Cloud – With Windows Server 2012 you will be able to monitor application health and have application mobility in new and more flexible ways. VM Monitoring will enable you to monitor the health of applications running inside of VMs in a lightweight way, and bubble the health state down to the host layer to take recovery actions. With Guest Clustering there will greater flexibility in how you configure a solution that achieves not only application health monitoring, but also application mobility. You will have greater flexibility to create Guest Clusters with fibre channel, iSCSI, or File (SMB) based storage.

Dynamic Clusters – With Windows Server 2012 how clustering determines quorum and resiliency is dynamic to the state of the cluster. This will deliver a private cloud that is flexible, dynamic, and more resilient.

Cluster Shared Volumes – First introduced in Windows Server 2008 R2, CSV enables all nodes in a cluster to access a common volume. With Windows Server 2012 CSV has undergone many innovations that enable it to provide a highly scalable, increased performance, security, and flexible shared storage infrastructure for your private cloud. Additionally, CSV will be supported with more workloads beyond Hyper-V... such as with a new Scale-out File Server.

Installing the Failover Cluster Feature and Tools in Windows Server 2012

Windows Server 2012 continues with the Roles and Features model. All clustering technologies are considered Features, as they are infrastructure that enables Roles to be made highly available.

Installing the Failover Clustering feature using Server Manager

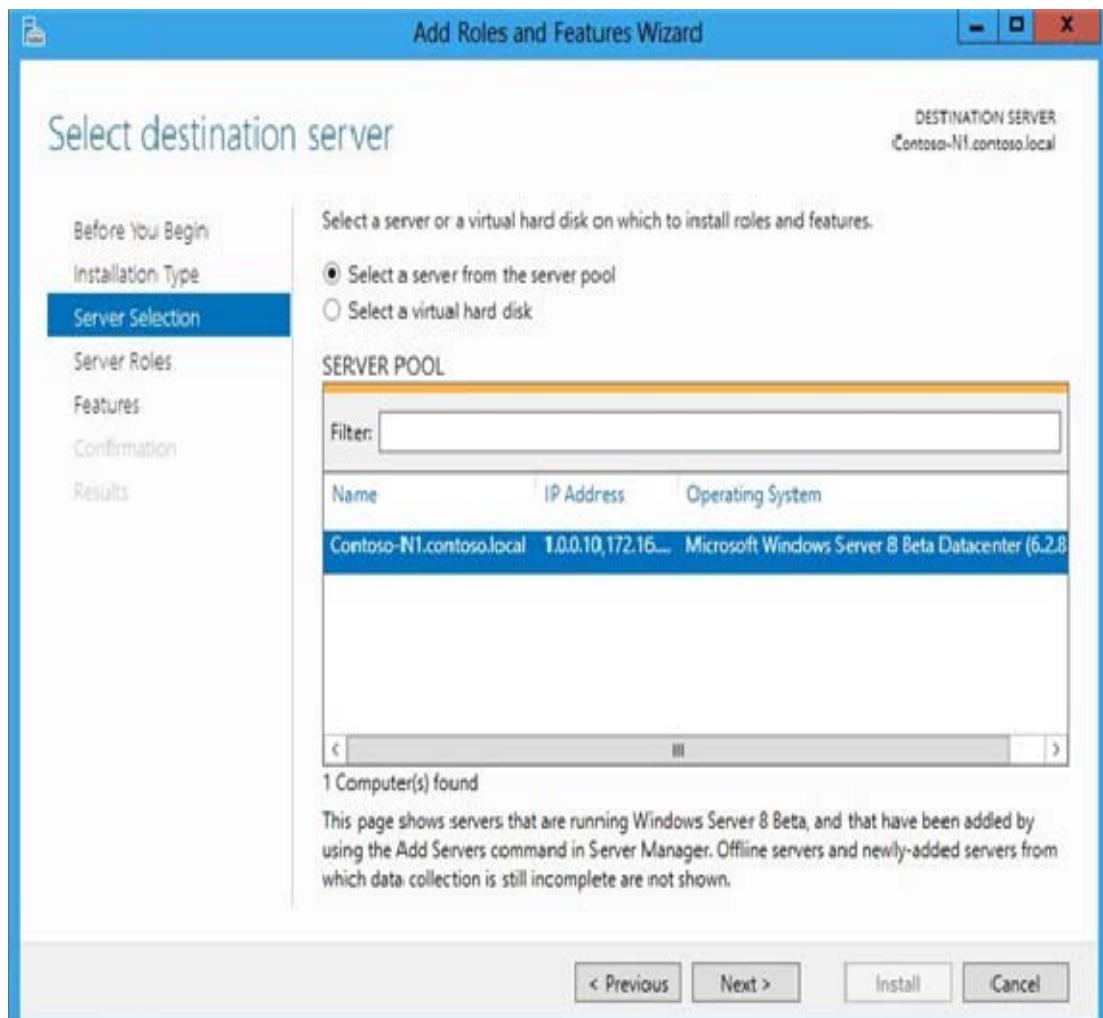
The Failover Clustering feature can be installed with either **Server Manager** or Windows PowerShell cmdlets. In Server Manager, the Add Roles and Features Wizard is used to add roles and\or features. The Add Roles and Features Wizard are accessed in the Server Manager **Menu** bar by choosing **Add Roles and Features** from the list.



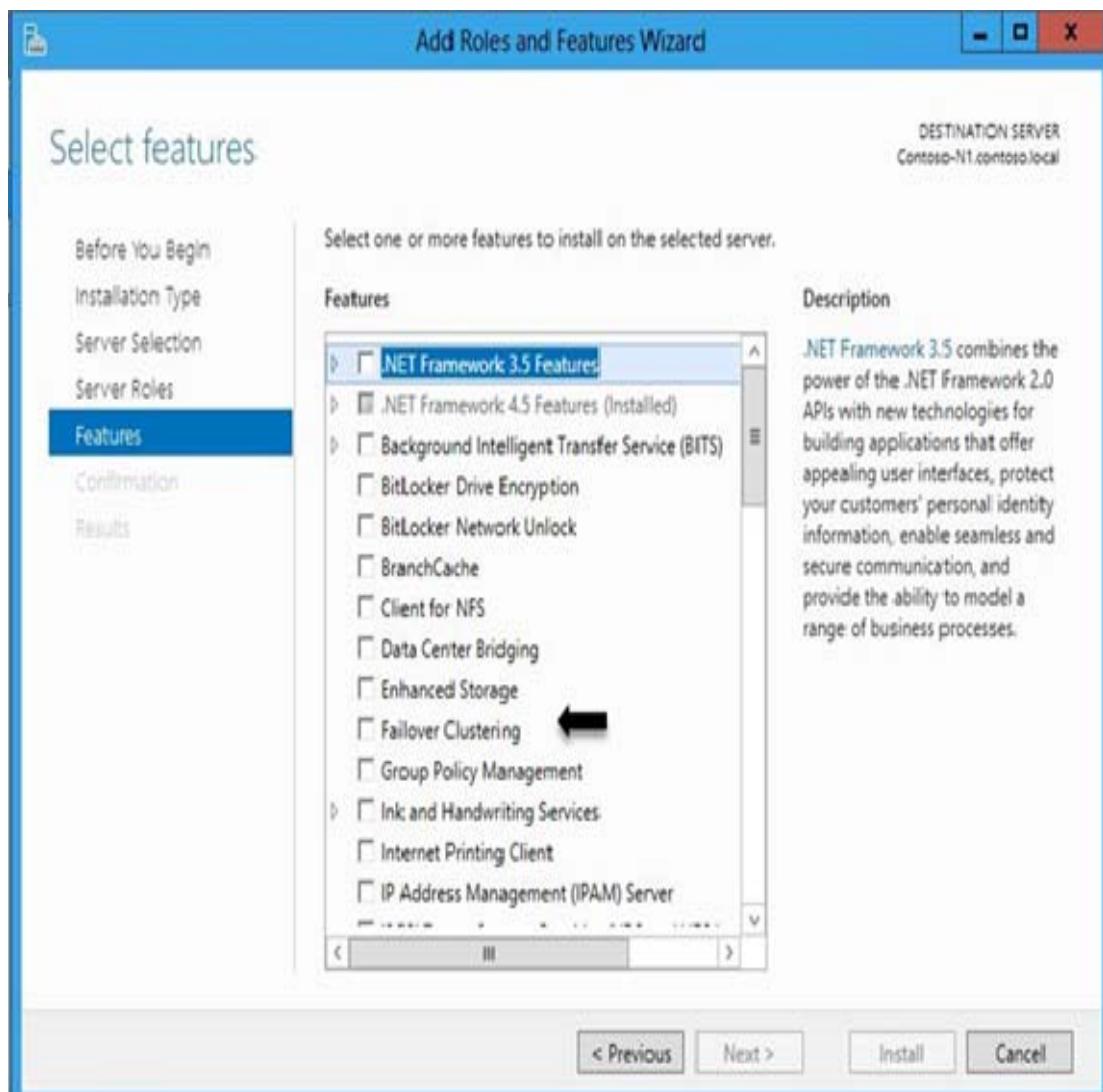
This starts the Add Roles and Features Wizard. The **Installation Type** is **Role-based or feature-based installation**.



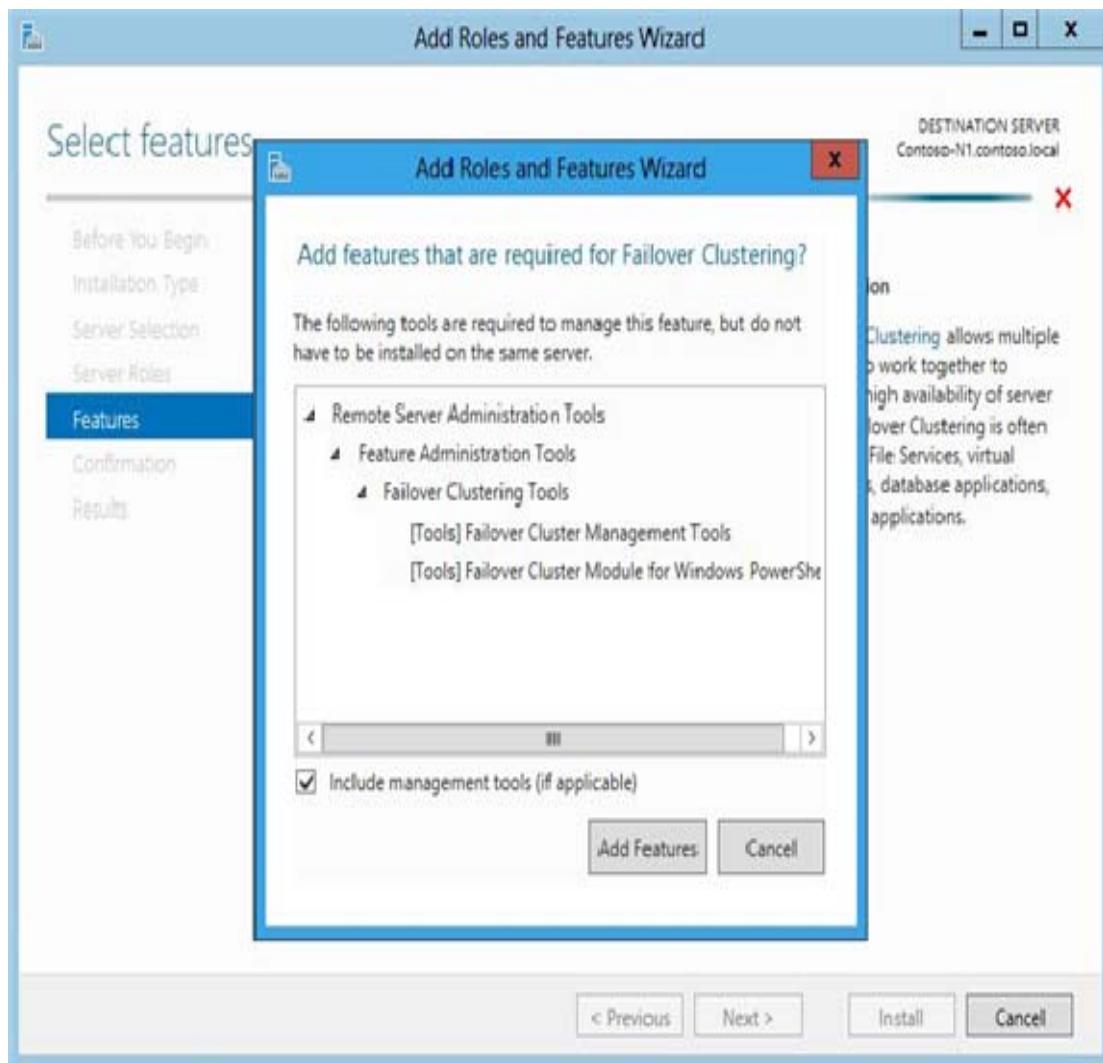
Be sure the correct server is selected in the **Server Selection** screen



In the **Features** screen, select **Failover Clustering**

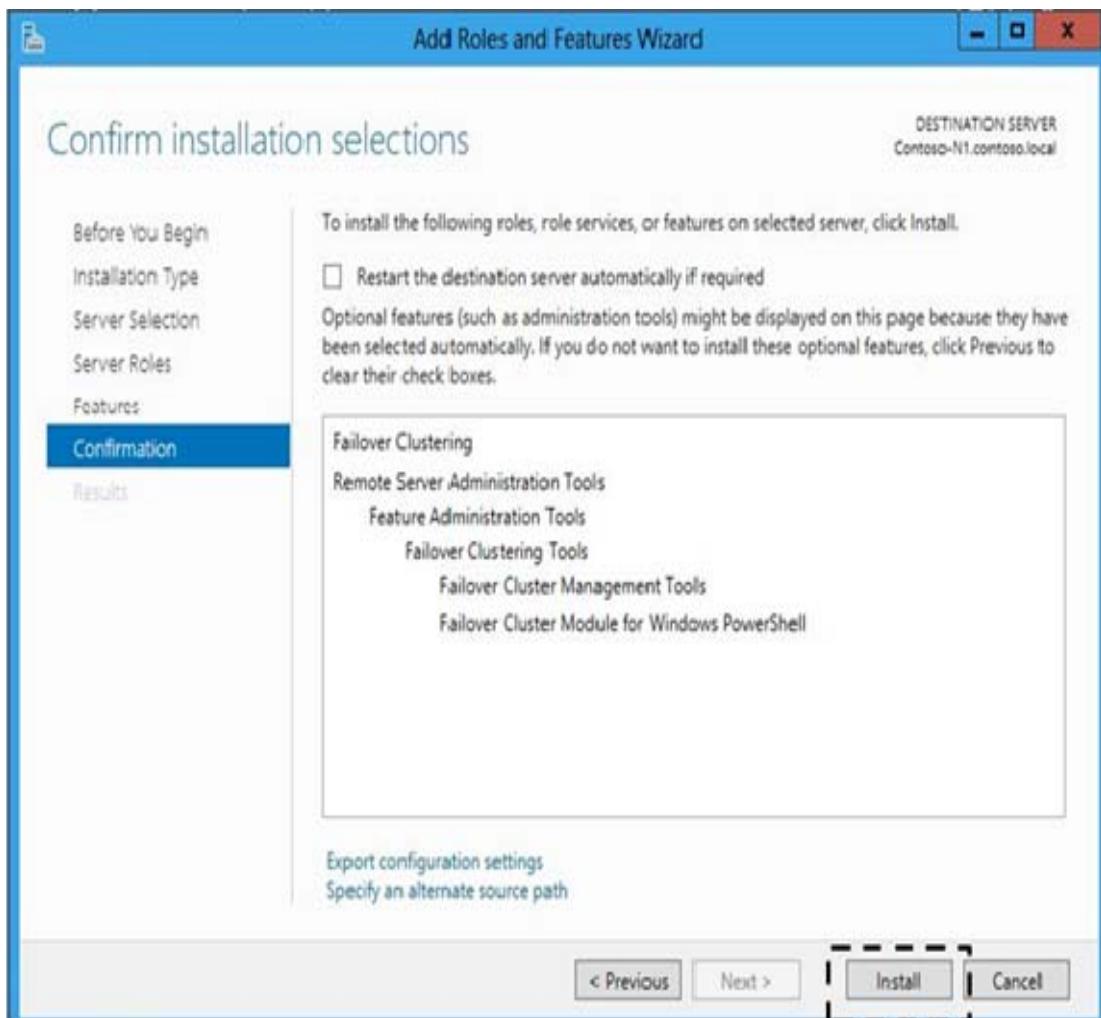


A pop-up screen appears listing additional requirements for the feature



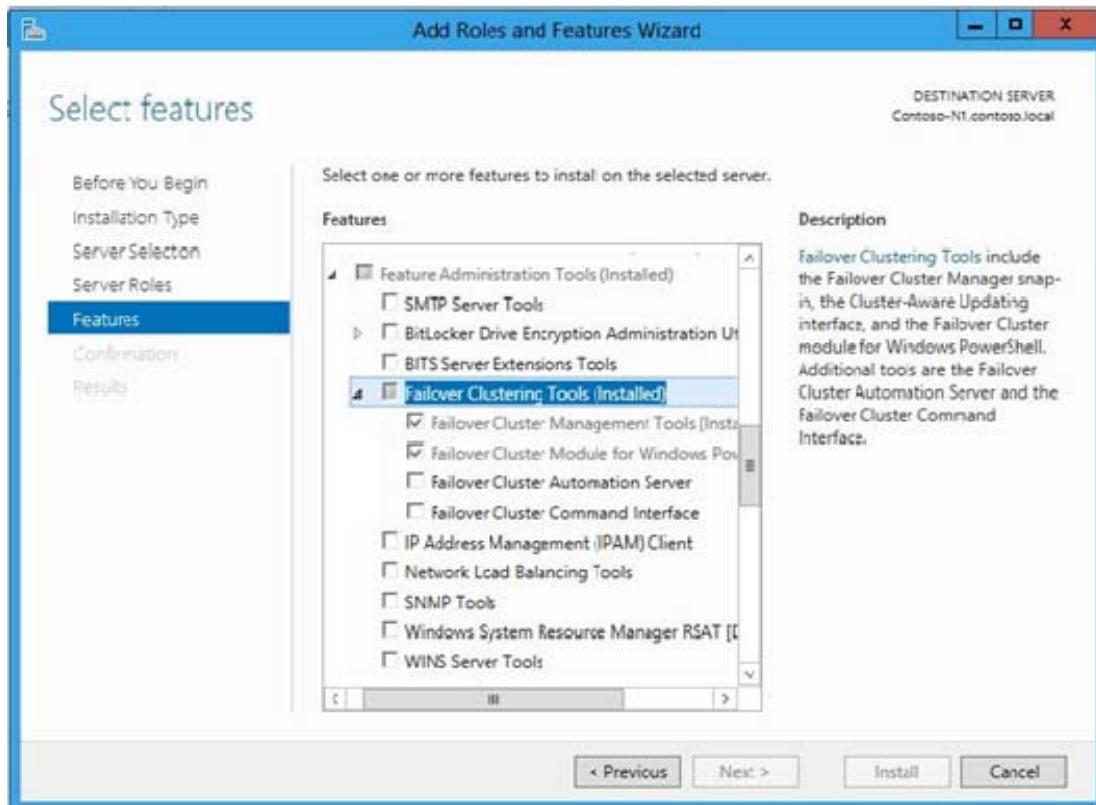
If you wish to install the Failover Cluster Manager snap-in and Failover Cluster PowerShell cmdlets, management tools, then click **Add Features**

Confirm the selections and click **Install**



The installation of the Failover Clustering feature does not require a reboot, checking the **Restart the destination server automatically if required** check box is not necessary.

There are optional features available for the Failover Clustering feature administration tools.

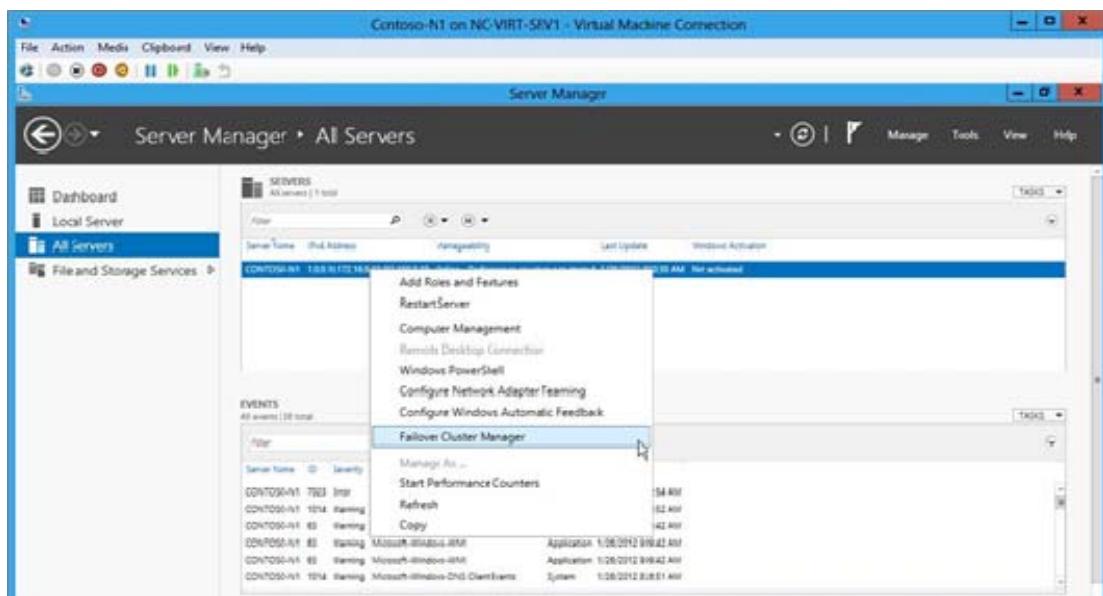


These are deprecated features (Failover Cluster Command Interface (cluster.exe) and Failover Cluster Automation Server) in Windows Server 2012 but are made available, as there are still some applications that may need them, SQL Server being one of them. Installing it may be necessary for any legacy scripts you have built on the old Cluster.exe command line interface.

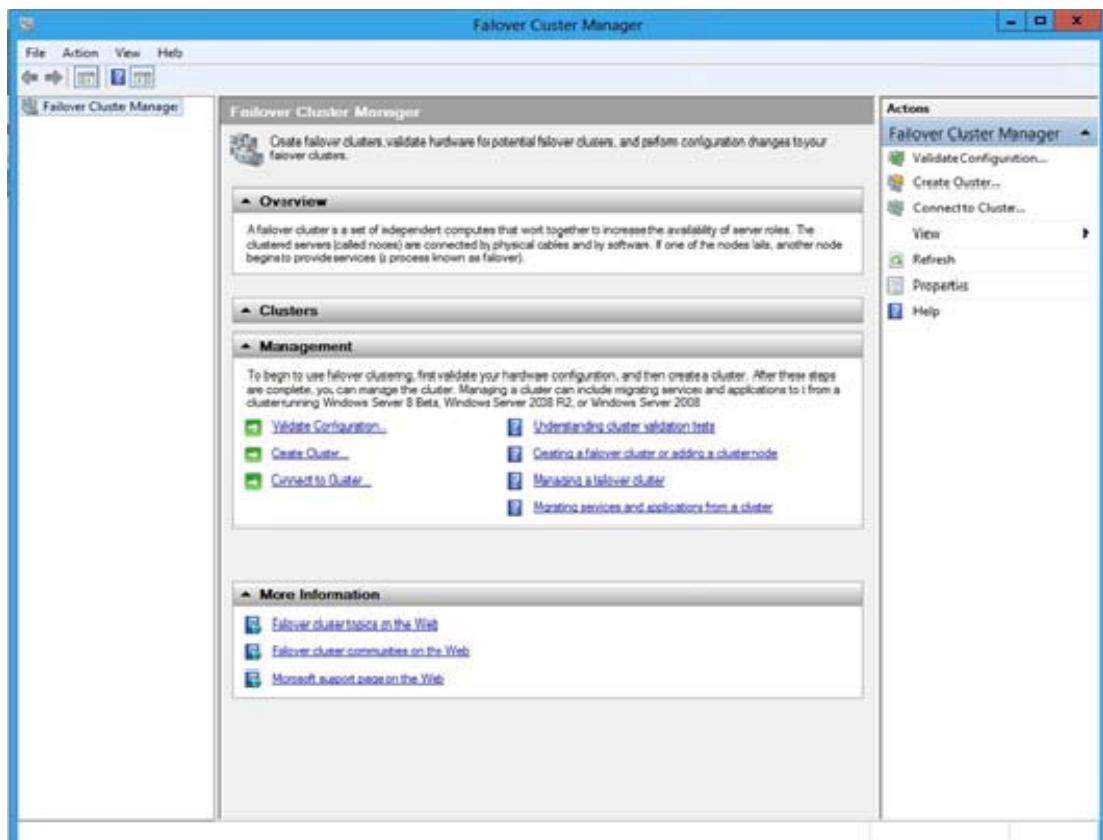
Once the Failover cluster feature is installed, the **Failover Cluster Manager** interface is available in the **Tools** category in the **Menu** bar.



Failover Cluster Manager is also available by right clicking on a node in the cluster in the **All Servers** view.



Choosing Failover Cluster Manager opens the snap-in.



Creating a Windows Server 2012 Failover Cluster

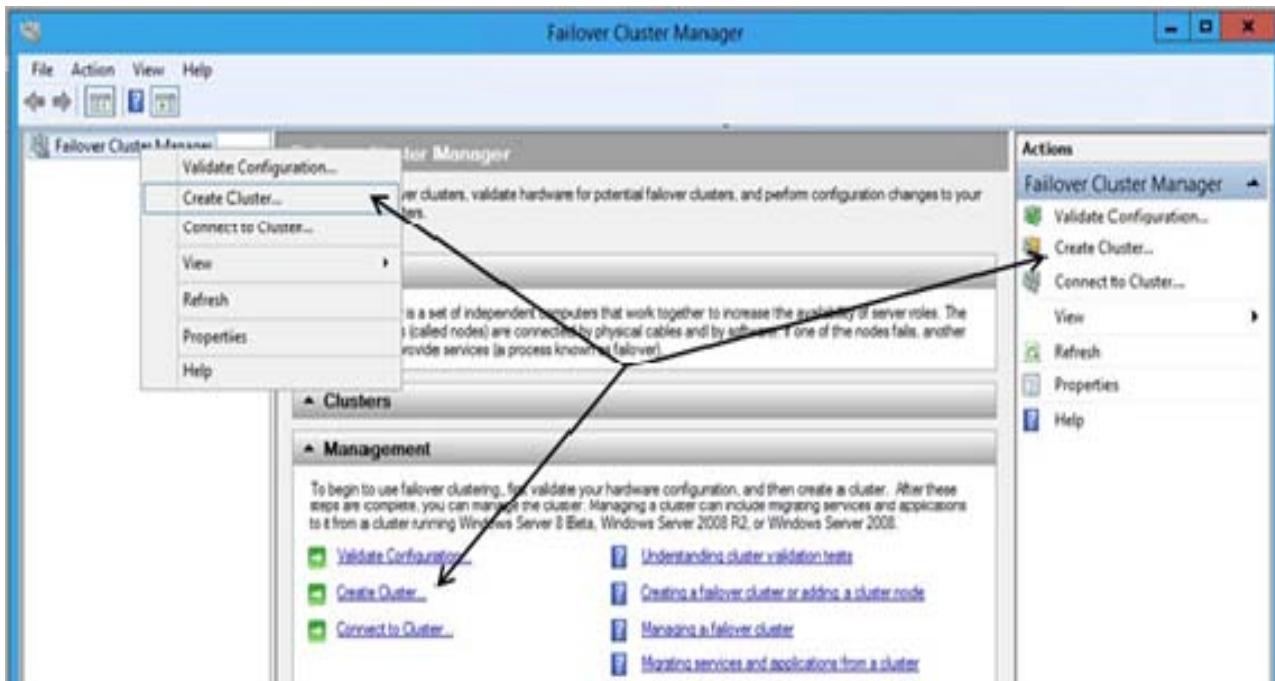
After installing the Failover Clustering feature, and validating a configuration, the next step is to create a new cluster. There are some minor changes in the create cluster experience in Windows Server 2012, but it is very similar to creating a cluster in Windows Server 2008 R2.

Creating a Failover Cluster using Failover Cluster Manager

1. Open **Failover Cluster Manager** - it can be opened from Server Manager using the Tools menu:



2. In the **Failover Cluster Manager**, choose the "Create Cluster..." action, which can be found in 3 places:

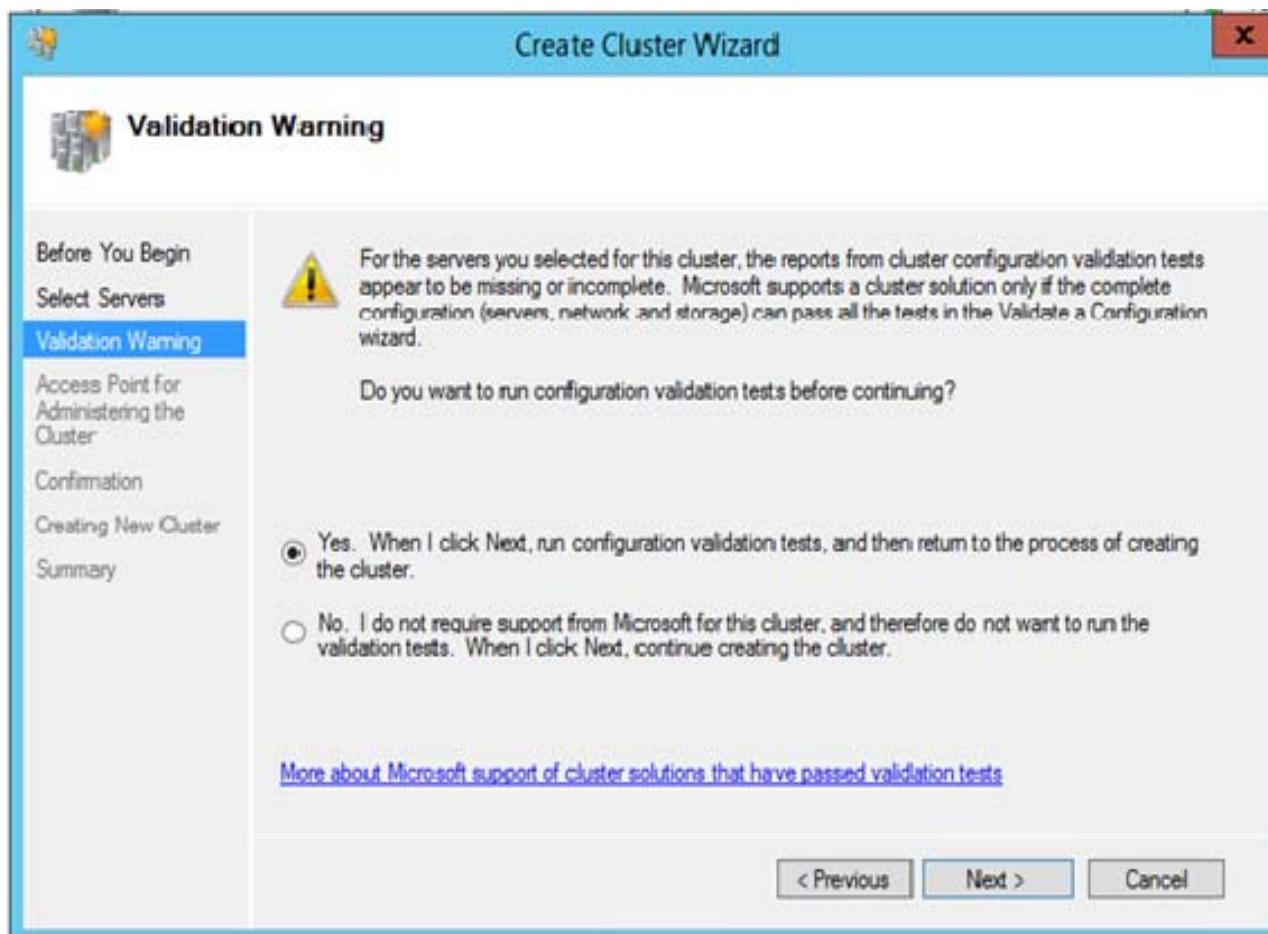


3. The **Create Cluster Wizard** initializes. Review the information on the **Before You Begin** screen. Click **Next**

4. Enter the names of all the servers that will be part of the cluster. **Note:** More than one node can be specified at a time using comma separation.

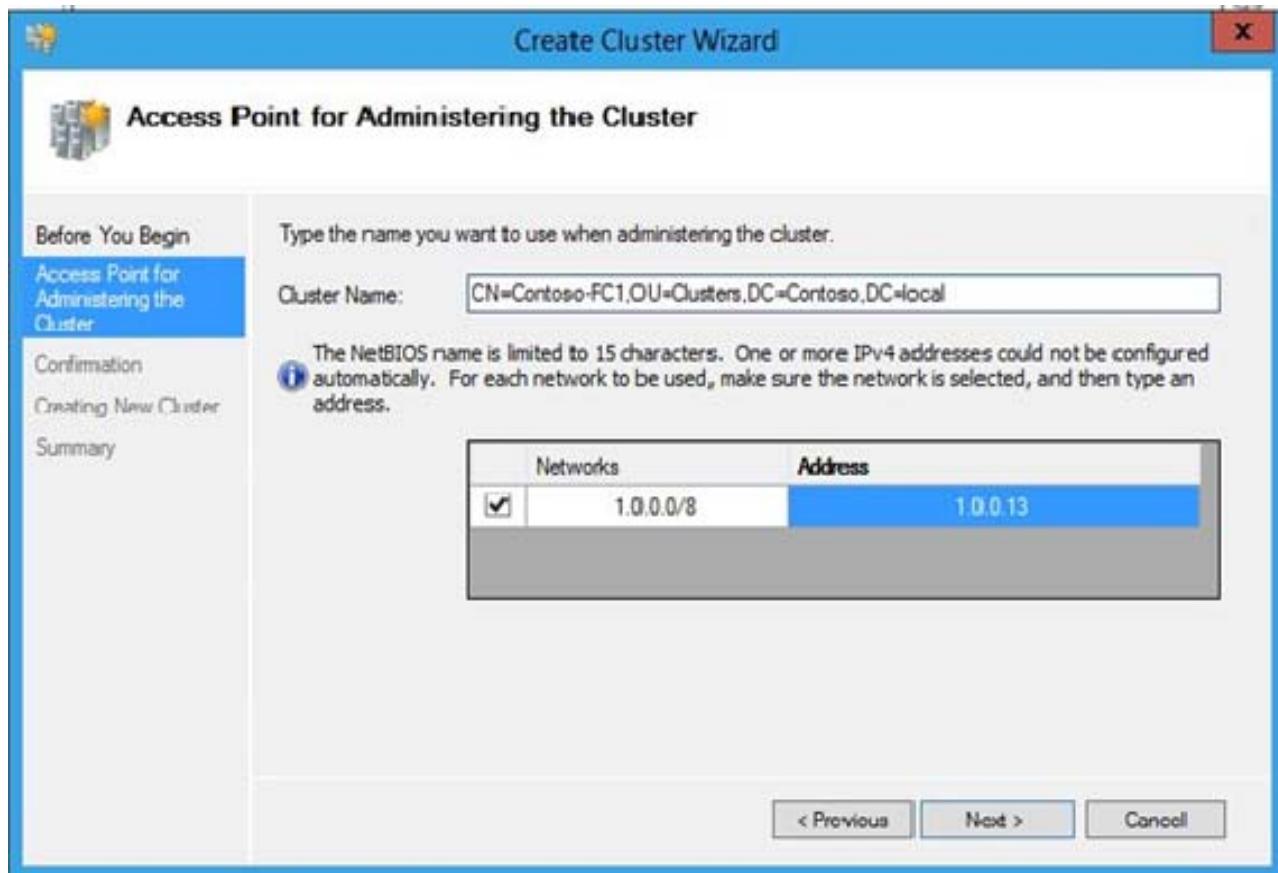
Example: **MyServer1, MyServer2, MyServer3**

5. If the nodes specified have not been validated, the following page in the wizard will be shown. It's highly recommended to validate the configuration before you create the cluster. This will help ensure that the servers are connected and configured correctly and that it can be supported by Microsoft:



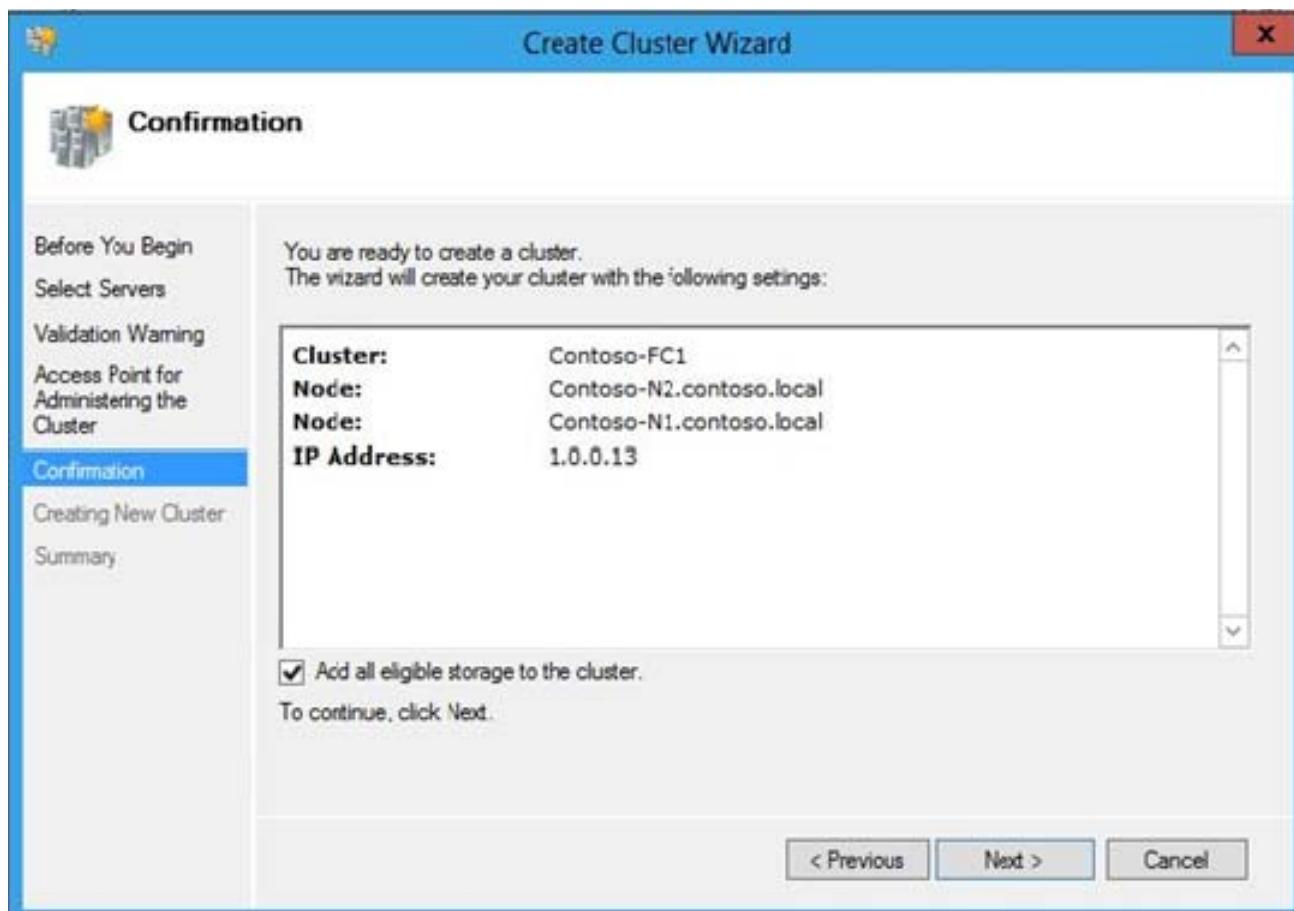
6. In the "Cluster Name" field, provide a NetBIOS name to be used as the cluster name. This cluster name is also the name that can be used to connect to the cluster to manage it. During cluster creation, a computer object will also be created in the Active Directory domain and Organizational Unit where the cluster nodes computer objects are located. If the servers have no NICs configured for DHCP, then this page will also prompt for a static IP address. If any of the networks are configured for DHCP, then this will not be shown and an IPv4 DHCP assigned address will be used. Click **Next**:

Note: If you do not want the Active Directory object for the cluster to be placed in the same Organizational Unit (OU) as the servers, the specific OU can be designated by specifying the full distinguished name like screen shot below:

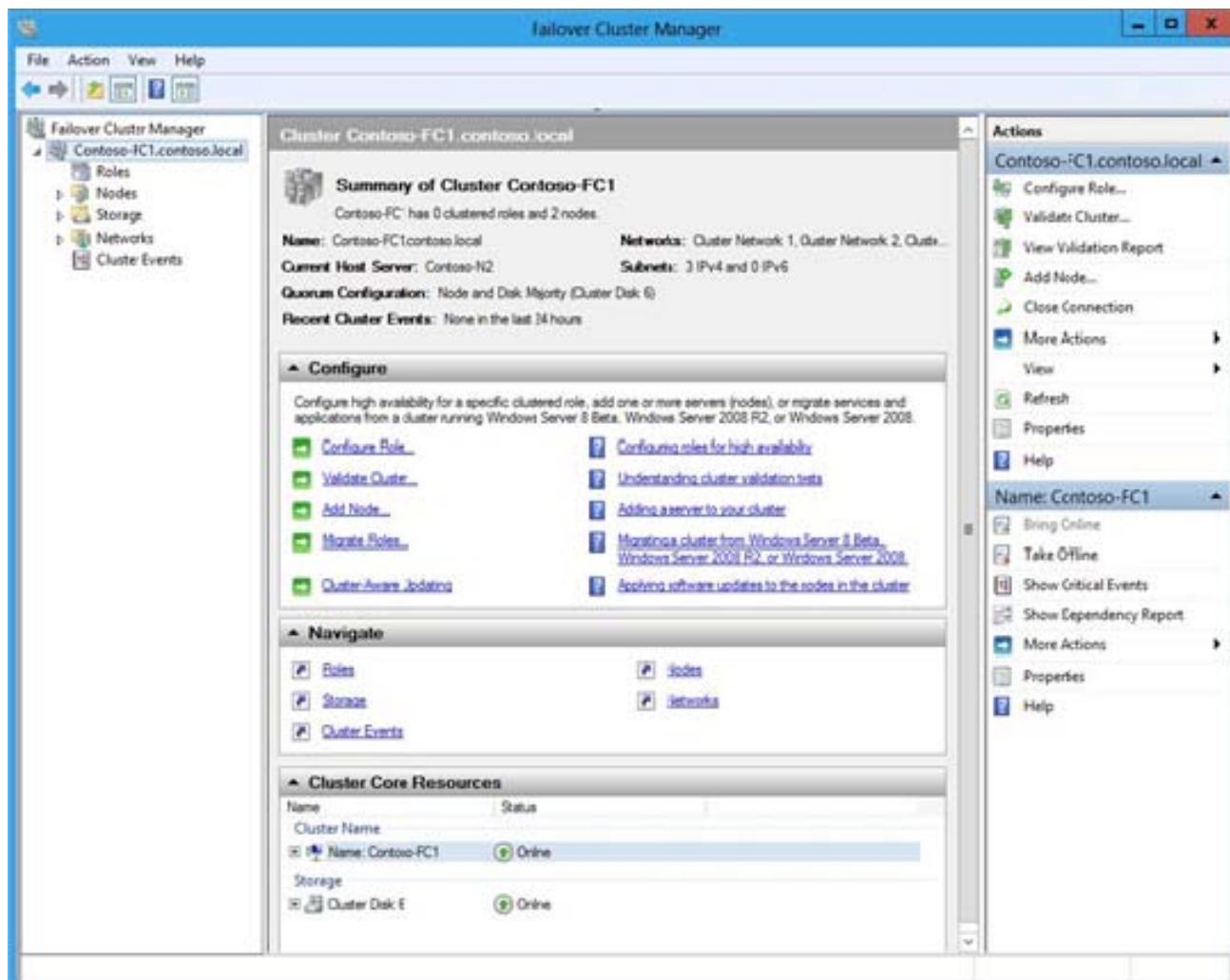


Review the **Confirmation** screen. If all eligible storage will be added to the cluster, check the box **Add all eligible storage to the cluster**. Click **Next**

Note: This ability to choose whether all eligible storage will be added to the cluster or not is new for Windows Server 2012. In previous versions all storage would always be added to the cluster. If you choose not to add all eligible storage to the cluster, you can add specific disks after the cluster is created:



7. The cluster should be successfully created. Review the **Summary** report if desired. Click **Finish**
8. A Failover Cluster Manager will automatically connect to the cluster when the wizard finishes:



Common SQL Server Failover Deployments

There are various architectural deployment scenarios for SQL Server failover clustering. This section describes the most common failover cluster configurations:

- Single failover cluster instance two-node cluster.
- Multi-instance cluster:
 - All nodes with active instances.
- Multi-site failover cluster instance.

Note: The terms “Active/Passive,” “Active/Active,” or some other combination of “Active” and “Passive” (such as “Active/Active/Passive”), are often used in conjunction with SQL Server failover clustering. This was an accurate definition when you could only install a single SQL Server instance on each node in SQL Server (on versions that predated SQL Server 2000). Therefore, from the application (SQL Server) perspective, any node could be “active” or “passive” in terms of running SQL Server. Today, however, with multiple failover cluster instances able to run on a single node, and the potential of many more

nodes within the cluster, these terms can be imprecise and ambiguous when used to describe the configuration of the SQL Server instances and the nodes they are running on. It is more accurate to describe each failover cluster instance separately in terms of where it is currently running and what its possible owners are. Furthermore, the “Active/Active” term can give the misleading impression that some load balancing is occurring across nodes or across SQL Server instances.

Single Failover Cluster Instance

The most common failover cluster configuration is a two-node cluster with a single SQL Server failover cluster instance. This arrangement provides a dedicated machine to protect against the failure of a single node. It is strongly recommended that both machines in this configuration have the same hardware provisioned to ensure that your workload continues to run after a failover without undesirable performance constraints.

If there is a need to tolerate additional machine failures you can increase the number of nodes within the cluster. SQL Server 2008 Enterprise Edition will be required if you want the SQL Server failover cluster to be able to fail over to more than two nodes. You should also consider the quorum type of the underlying Windows Server cluster to ensure that it can tolerate the desired number of failures.

Sometimes provisioning dedicated spare machines for each SQL Server failover cluster instance is too costly and you may be willing to trade off availability for hardware utilization by combining multiple SQL Server failover cluster instances on a single failover cluster. This is referred to as a multi-instance failover cluster.

Multi-Instance Failover Cluster

There are two common types of multi-instance clusters. The first is where all nodes are running one or more failover cluster instances. When a node fails, all of the failover cluster instances that it is hosting will fail over to another node. The primary consideration for this configuration is whether each node has enough hardware capacity to run the peak workloads for the failover cluster instances that it already hosts, plus the failover cluster instances that it will take over in the event of the failure of another node. This issue can be compounded if you need to tolerate multiple node failures. You can use the preferred owners and **AntiAffinityClassNames** resource group properties for greater control of the balancing of SQL Server instances during a failure scenario. As in any availability solution, you should ensure that there is enough spare hardware capacity to continue serving the application workloads in the event of machine failure(s). Alternatively, you should be prepared to run under a modified service level agreement (SLA) for the duration that the node is unavailable.

The second common multi-instance cluster configuration is what is often referred to as an “ $n+1$ ” configuration. In a number of nodes (n), each node is running one or more active failover cluster instances, and a dedicated spare node ($+1$) is available to take on the workload of any other node upon failure. This configuration reduces potential capacity considerations during failure that arise when all nodes are active. It also generates greater machine utilization because you can use the dedicated spare machine across multiple failover cluster instances.

You can, of course, increase the number of spare nodes to any number, as long as you stay within the SQL Server supported limits of nodes for your operating system version. Using preferred owners and SQL Server 2012 DBA

AntiAffinityClassNames on the resource group gives greater control over SQL Server failover cluster instance allocation or balancing during multiple failures.

Multi-Site Failover Cluster

A multi-site failover cluster includes nodes that are dispersed across two physical sites or data centers, with the aim of providing a disaster recovery solution to protect against site failure. Sometimes multi-site failover clusters are called *geographically dispersed failover clusters* or *stretch clusters*. In addition to Windows Server failover clustering, SQL Server relies on two key pieces of technology in a multi-site failover cluster:

- Storage replication technology (for example, SAN replication) to ensure that the dependent disks can fail over.
- Stretch virtual LAN (VLAN) to provide a single subnet across all nodes.

When you choose a storage replication technology for multi-site failover clustering, ensure that it meets SQL Server storage requirements and that your databases and their logs are synchronously and consistently replicated in order to avoid data loss. If your database and log files reside on different volumes, you should work with your storage solution to ensure consistency between the data and log file when SQL Server recovers the database after a failover. The key concern is that SQL Server write ordering requirements are adhered to and that the data is consistent even when network interruptions or failures occur.

Although features of Windows Server 2008 added functionality to support multi-subnet failover clusters, SQL Server 2008 does not support this additional functionality and requires that all nodes reside on the same subnet. Different sites or data centers often do not share a single subnet, so implementing a multi-site failover cluster for SQL Server 2008 generally requires the use of a stretch VLAN solution to expose a single subnet on all nodes of a SQL Server failover cluster instance. This is an appropriate configuration (single subnet) for a multi-site SQL Server failover cluster.

By using the storage replication and stretch VLAN technologies to provide common network and storage access to all nodes, a multi-site failover cluster will appear to SQL Server 2008 to be similar to a standard single-site failover cluster.

When you configure a multi-site failover cluster, you must also consider:

- The effect that site failures or a loss of communication can have on quorum. Configuring quorum settings on a multi-site failover cluster is the same as configuring them for a single-site failover cluster. However, a multi-site cluster configuration is designed to protect against a site failure and is also susceptible to a loss of communication between physical sites. Therefore, typically, the quorum is configured so that either site can run should the other fail.
- The configuration of heartbeat settings to tolerate increased latency. Depending on the latency between the sites in your multi-site failover cluster, you may want to increase the tolerance for Windows Server failover clustering inter-node heartbeat to avoid latency falsely being detected as stress.

You can, of course, combine multi-site failover cluster instances into the multi-instance failover cluster.

Note: Some multi-site failover cluster implementations have experienced failures of SQL Server Setup during disk qualification. This is due to how SQL Server 2008 Setup investigates disk resources and the resources that are dependent on those disks. See the Developer Note below for further technical information about this. If you believe you are experiencing this problem, consider:

1. Using the Failover Cluster Management MMC to verify that all disks are part of the same group and that there are no nondisk resource dependencies to any of the disks.
2. Contact the hardware partner for advice on the appropriate course of action.

Developer Note: SQL Server 2008 Setup does not look at the physical storage resource type. Instead, the CLUS_RESCLASS_STORAGE flag is investigated when it looks for storage resources and checks dependencies. If you are getting blocked during disk qualification, it is likely that a resource in the dependency chain does not have that flag set. If the resources on both sides of the dependencies are CLUS_RESCLASS_STORAGE, then they will not be blocked by SQL Server Setup. Please note that this applies to the entire storage dependency graph. SQL Server Setup walks the entire graph to make sure that it has only CLUS_RESCLASS_STORAGE. Even if the dependencies that are directly connected to the disk resource are all valid, the whole chain is disqualified if the replication solution is referenced by, for example, a service. Any graph that contains just CLUS_RESCLASS_STORAGE should pass the check.

Advantages of SQL Server 2008 on Windows Server 2008 Clustering

Although a Windows Server 2003 failover cluster supports the installation of SQL Server 2008 failover clustering, Windows Server 2008 failover clustering introduces several important new features that SQL Server 2008 failover clustering takes advantage of:

- **The removal of the validation requirement for cluster solution hardware and components with the Hardware Compatibility List (HCL) and Windows Server Catalog**
Before Windows Server 2008, Microsoft server failover clusters were only deemed “supported” if the entire cluster solution was listed in the Windows Server Catalog under the **Cluster Solutions** category. Matching a given cluster hardware solution against the Windows Server Catalog could be challenging, and often required further communication and validation with the solution vendor to determine supportability.
Windows Server 2008 removes this requirement. Instead of checking with the Windows Server Catalog, your Windows Server 2008 cluster solution must instead pass validation by using the Windows Server 2008 Cluster Validation Tool. Before cluster configuration, this tool scans the server nodes and storage that you intend to use for your cluster solution. The tool validates hardware and reports on any blocking issues that may impact the support of the failover cluster, detailing issues across storage, the operating system, hardware, and network components. If all hardware components are certified for Windows Server 2008 and the validation passes all tests, the failover cluster solution is supported from Microsoft’s point of view.
- **Internet small computer system interface (iSCSI) support**
Expanding SQL Server 2008 failover cluster storage options beyond Fiber Channel and Serial Attached SCSI (SAS) storage connections, Windows Server 2008 failover clusters support iSCSI storage connections.

- **IP version 6 (IPv6) support**
Windows Server 2008 DNS servers support IPv6, which expands IP address size to 128 bits (rather than 32 bits with IPv4). SQL Server 2008 failover clustering natively supports IPv6.
- **Dynamic Host Configuration Protocol (DHCP) support**
DHCP is now supported for Windows Server 2008 failover cluster server nodes. The cluster service is responsible for managing the IP addresses, and SQL Server 2008 failover clustering supports this scenario. However, from a system stability perspective, it is still recommended to use static IP addresses for the Windows Server 2008 failover cluster server nodes rather than DHCP when hosting SQL Server 2008 failover clusters. Any service or application dependencies on an IP address that fails to renew could cause a failure in the IP address resource, in addition to a failure with the clustered service or application.
- **Service SIDs**
SQL Server 2005 failover clustering introduced the requirement of designating domain groups during installation. Domain groups were used in the installation process to provision ACLs and required permissions. These designated domain groups required that the SQL Server service accounts already be members of the domain groups. If this was not the case, permissions were needed to add them as members. This requirement added administrative difficulty to the overall installation process because it became hard to create a common separation of duty between database administrator (DBA) teams and Active Directory® administrators.

When you install a SQL Server 2008 failover cluster on a Windows Server 2008 failover cluster, domain groups are no longer required. Instead, you can choose to designate the use of service SIDs. When you install a SQL Server 2008 failover cluster, you designate “Use service SIDS” to bypass the need to provision domain groups. Service SIDs, introduced in Windows Server 2008 and Windows Vista, enables you to bind permissions directly to a Windows service.

- **Supported number of cluster nodes**
The 64-bit edition of Windows Server 2008 Enterprise expands the number of nodes that are supported in a single cluster to 16. On Windows Server 2003, only eight nodes were supported.

Although many of the new features that were introduced in Windows Server 2008 failover clustering are available with SQL Server 2008, a few Windows Server 2008 features are not currently supported or exposed to SQL Server 2008:

- **Installation on Windows Server 2008 Server Core**
SQL Server 2008 failover clusters and stand-alone instances of SQL Server 2008 are *not* supported on Windows Server 2008 Server Core.
- **Separate subnets for cluster nodes**
Although Windows Server 2008 failover clusters support separate subnets for cluster nodes, SQL Server 2008 failover clusters do not. Due to the popular demand for this feature, it is expected that multi-subnet support will be added in a future version of SQL Server.

SQL Server Installation and Setup

The installation process for SQL Server 2008 failover clusters has changed significantly. The focused and discrete installation steps within the setup of the failover cluster help to minimize failures. They also ensure that you can accomplish more discrete tasks and installation options during SQL Server 2008 failover cluster setup. Unlike earlier versions, when you install or upgrade a SQL Server 2008 failover

cluster, you must run the setup process for SQL Server individually on each node of the failover cluster. To add a node to an existing SQL Server failover cluster, you must run SQL Server Setup on the node that is to be added to the SQL Server failover cluster instance. Unlike SQL Server 2005, the node that needs to be managed or modified is where the SQL Server Setup should be executed from.

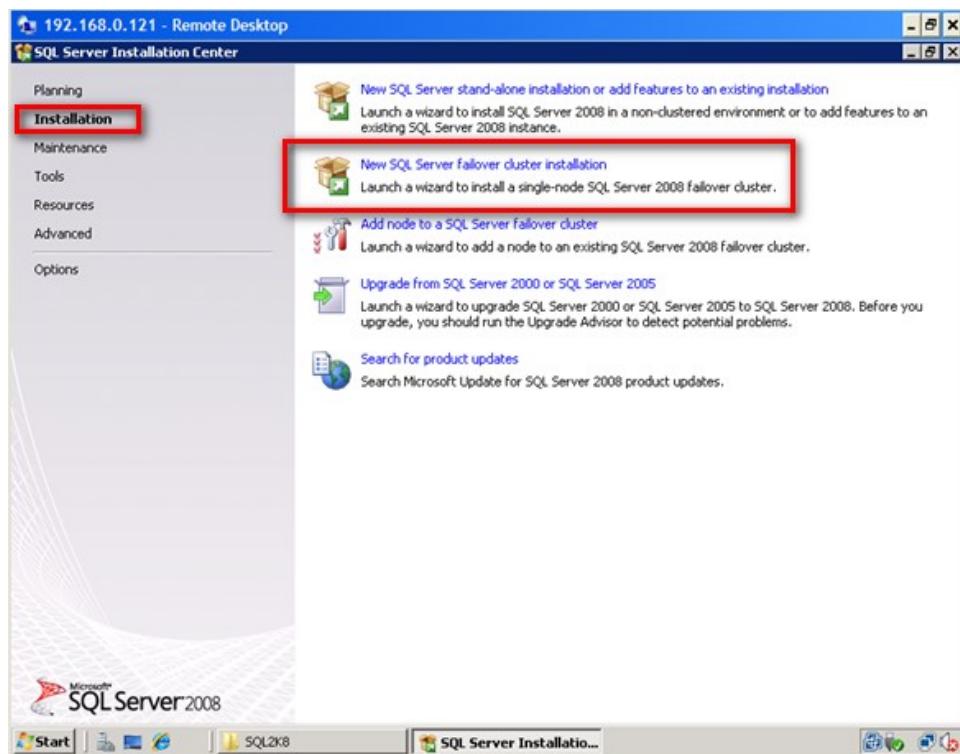
Before you begin the SQL Server installation, validate the hardware configuration and the functionality of the underlying Windows Server failover cluster by confirming that all resources can fail over and come online.

Installing SQL Server 2008 on a Windows Server 2008 cluster

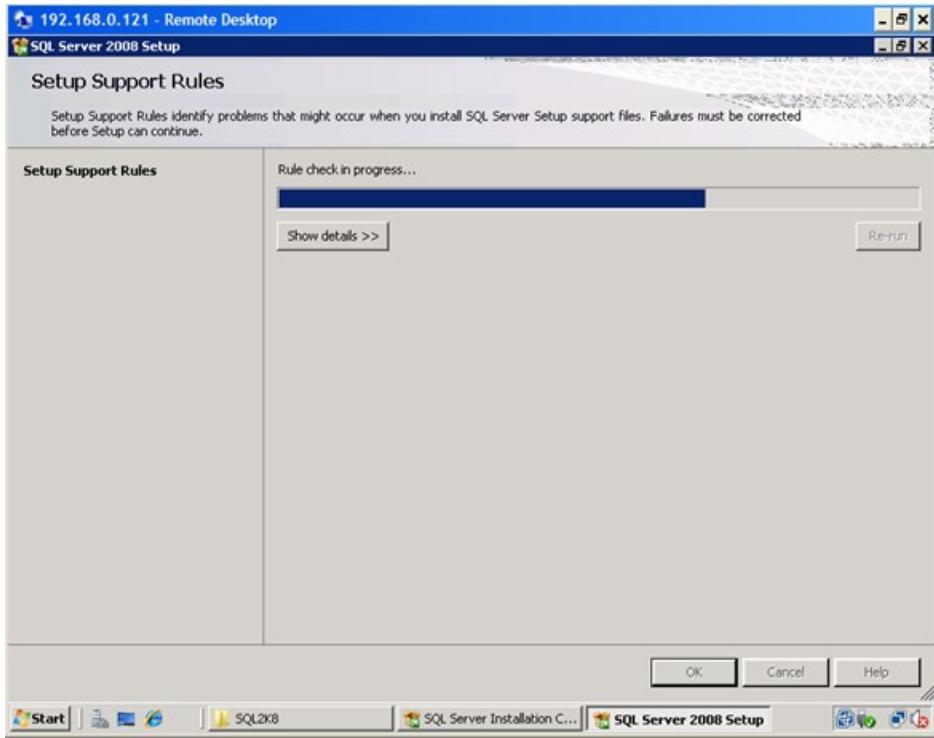
There are two options to install SQL Server 2008 on a cluster. The first one is by using the integrated failover cluster install with Add Node option and the second one is the Advanced/Enterprise installation option. The process outlined below will take into account the first option.

To install SQL Server 2008:

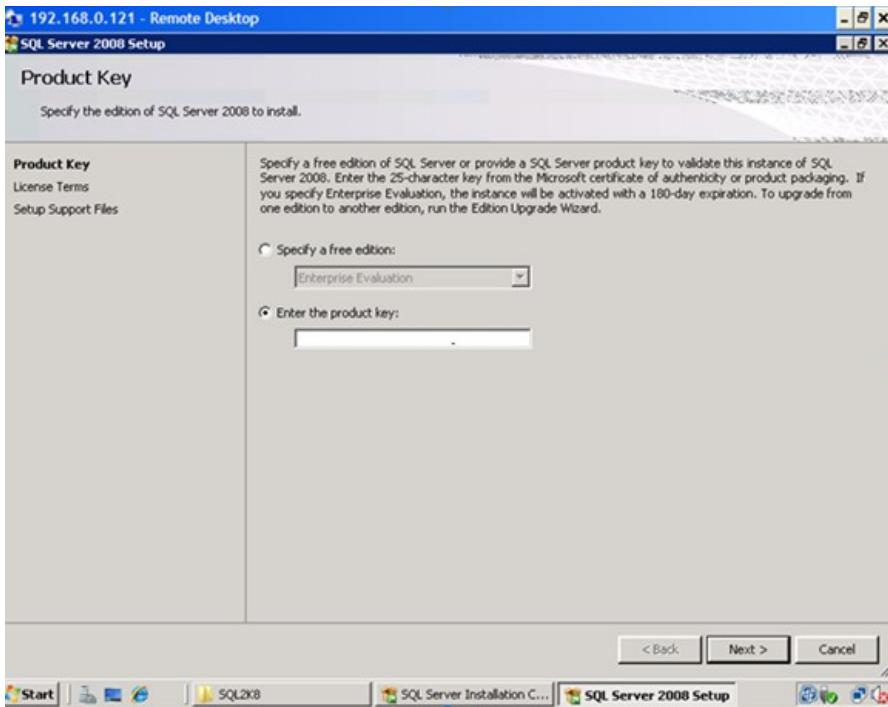
1. Run **setup.exe** from the installation media to launch **SQL Server Installation Center**. Click on the **Installation** link on the left-hand side
2. Click the **New SQL Server failover cluster installation** link. This will run the **SQL Server 2008 Setup** wizard



3. In the **Setup Support Rules** dialog box, validate that the checks return successful results and click **Next**.



4. In the **Product Key** dialog box, enter the product key that came with your installation media and click **Next**.

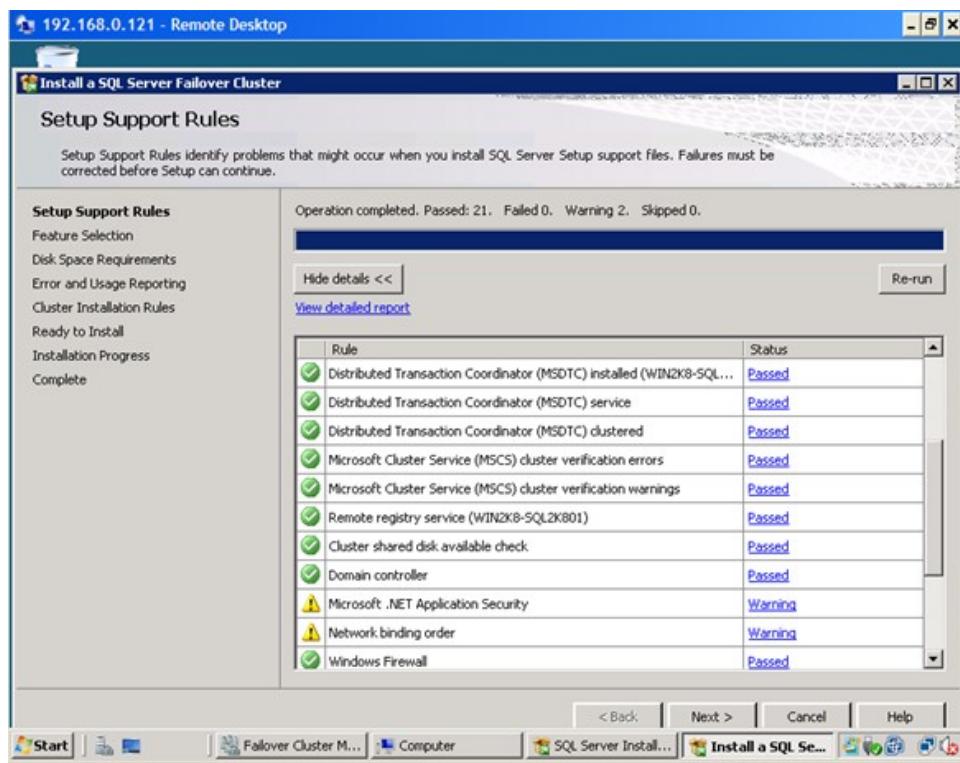


5. In the **License Terms** dialog box, click the **I accept the license terms** check box and click **Next**. You probably haven't read one of these, but if you feel inclined go for it.

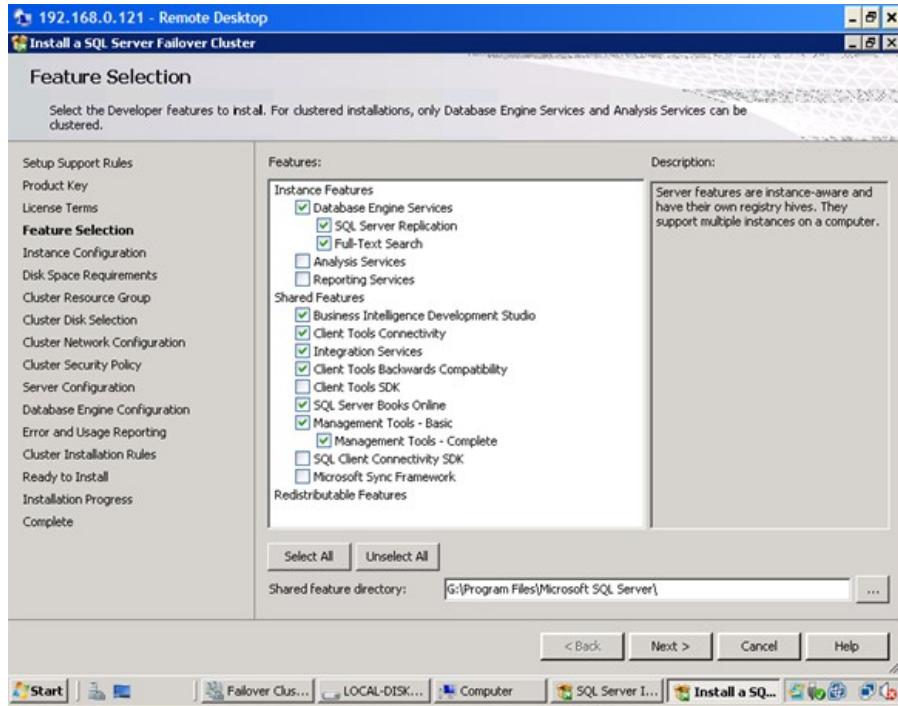


6. In the **Setup Support Rules** dialog box, click **Install**. Validate that the checks return successful results. If the checks returned a few warnings, make sure you fix them before proceeding with the installation. An example of this is the **Network binding order**. The public network cards should be first on both nodes. Also, you can disable NETBIOS and DNS registration on the network cards to avoid network overhead. Be sure to check your binding order as well.

For the Windows Firewall, make sure that you open the appropriate port number on which SQL Server will communicate. You can do this after the installation. Alternatively, you can disable Windows Firewall during the installation and enable it later with the proper configuration. Click **Next** to proceed.



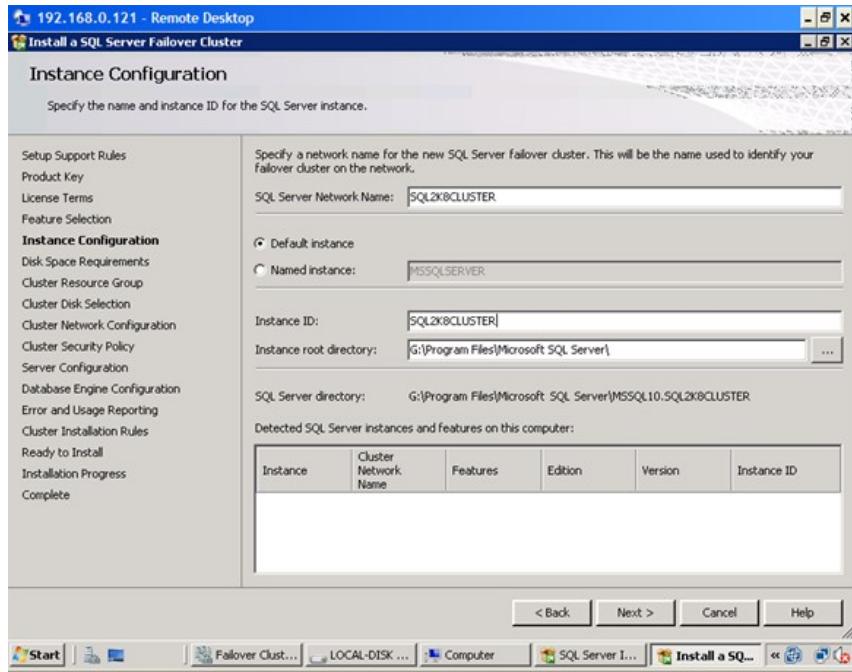
7. In the **Feature Selection** dialog box, select only the components that you want installed. For the **Shared feature directory**, you can keep the default path if you have sufficient disk space on your C:\ drive or anywhere that is a **local disk** as this will be used by the SQL Server installation process later on. The directory for the clustered database engine will be different. Click **Next**.



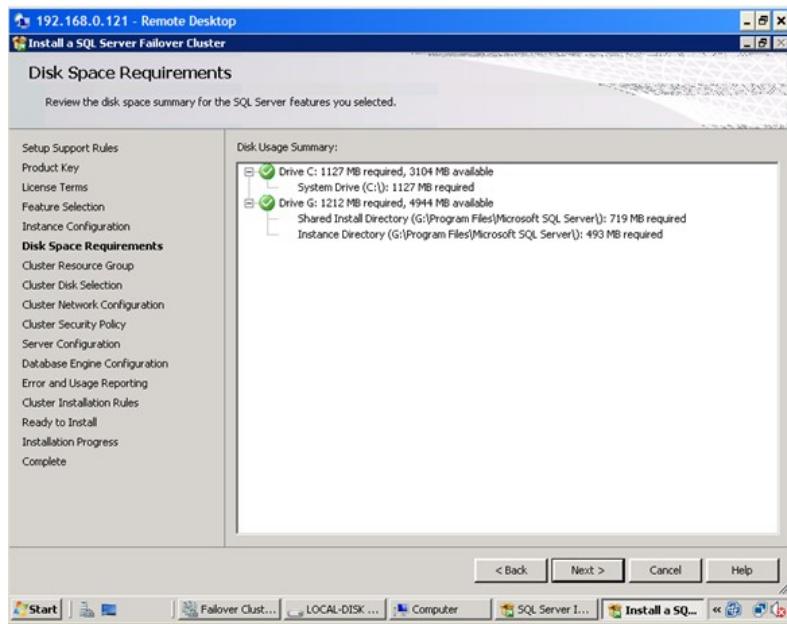
8. In the **Instance Configuration** dialog box, enter the SQL Server Network Name. This is the name that will be available on the network for the clients. This will vary depending on your selection of whether it is a default or named instance. In this example, default instance is selected.

A couple of things need highlighting in this section. By default, the instance name is used as the **Instance ID**. This is used to identify installation directories and registry keys for your instance of SQL Server and is helpful when you want to run multiple instances in a cluster. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be **MSSQLSERVER**. To use a non-default instance ID, you should select the **Instance ID** box and specify a value.

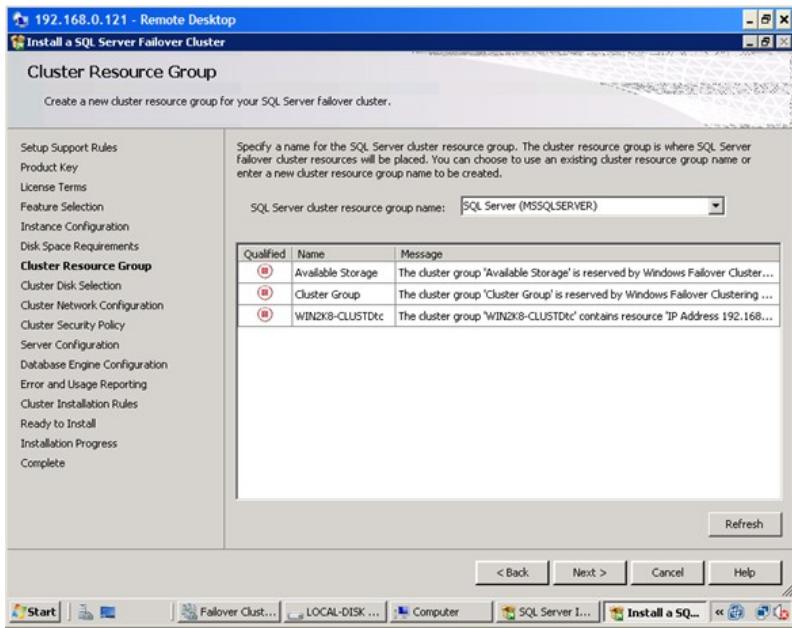
The section on **Detected SQL Server instances and features on this computer** would make sense if there are other SQL Server instances running on your server.



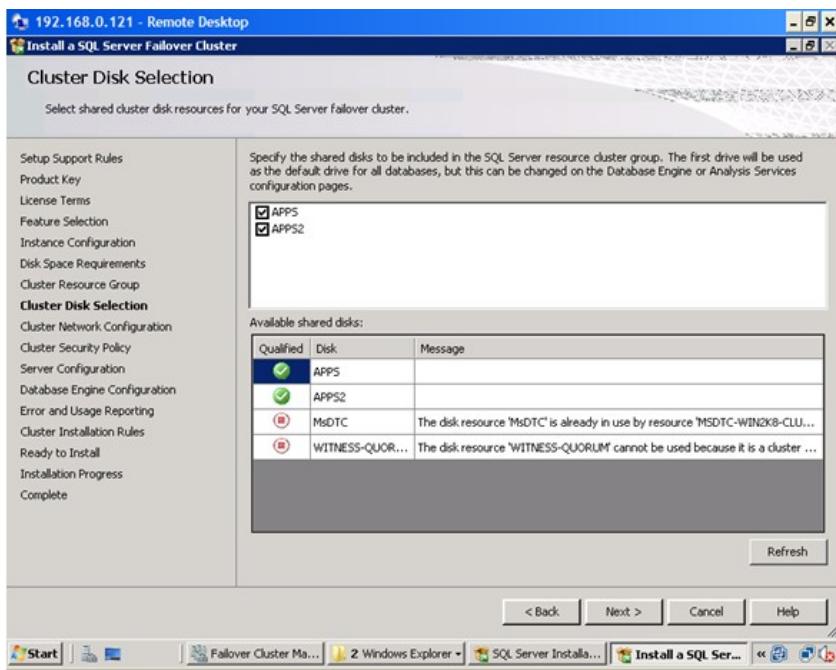
9. In the **Disk Space Requirements** dialog box, check that you have enough space on your **local disks** to install the SQL Server 2008 binaries and click **Next**.



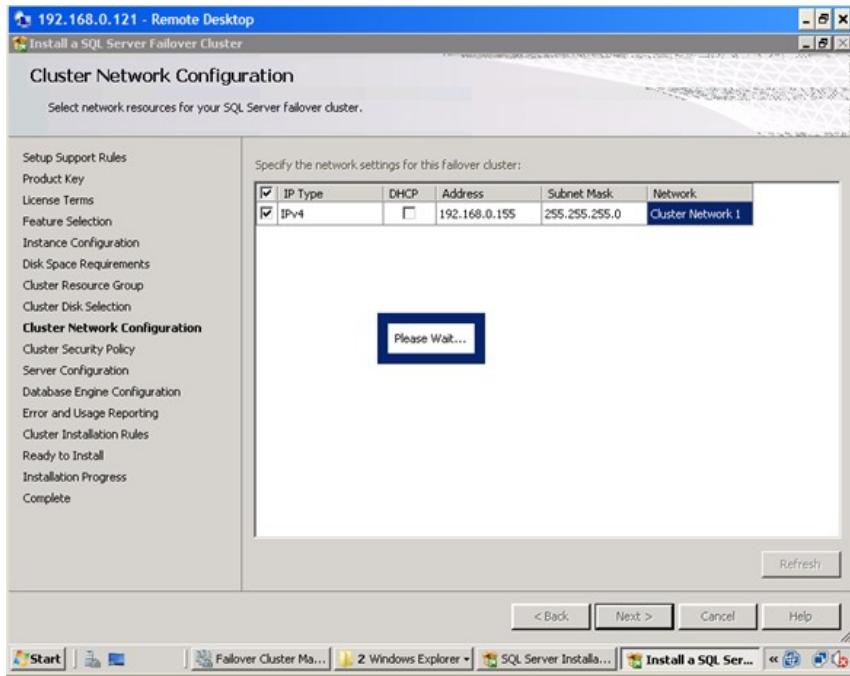
10. In the **Cluster Resource Group** dialog box, check the resources available on your Windows Server 2008 cluster. This will tell you that a new Resource Group will be created on your cluster for SQL Server. To specify the SQL Server cluster resource group name, you can either use the drop-down box to specify an existing group to use or type the name of a new group to create it. Click **Next**.



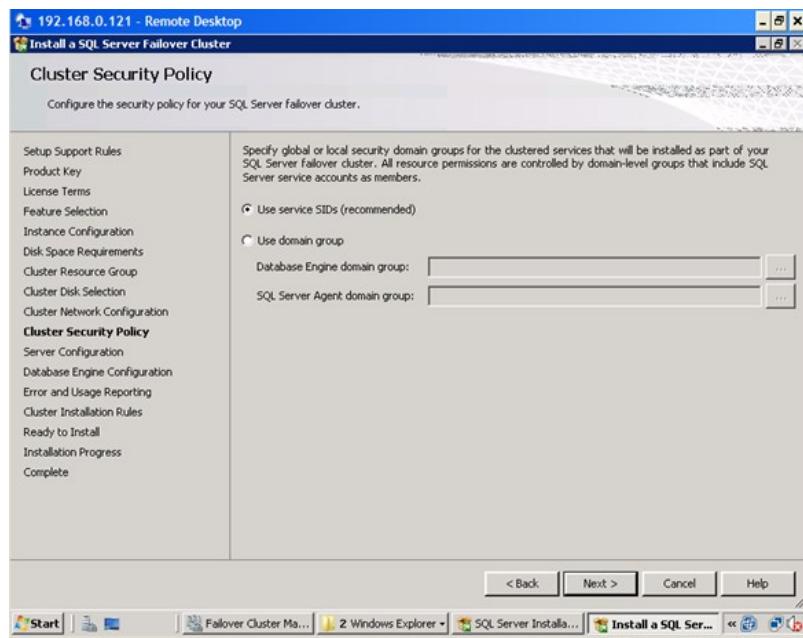
11. In the **Cluster Disk Selection** dialog box, select the available disk groups that are on the cluster for SQL Server 2008 to use. In this example, two clustered disk groups – **APPS** and **APPS2** – have been selected to be used by SQL Server 2008. I will be using one disk resource for the system databases while the other one for the user databases. Click **Next**.



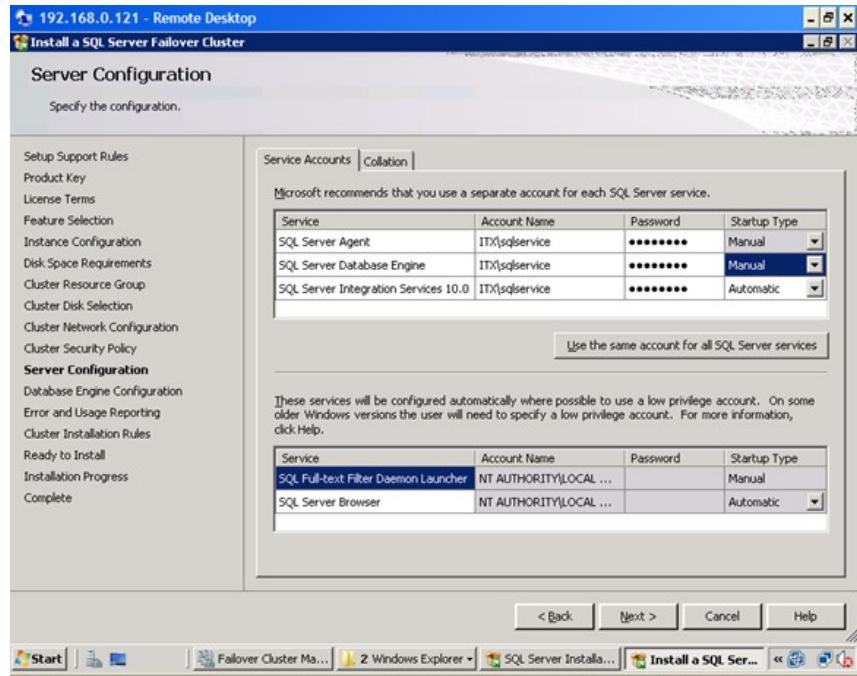
12. In the **Cluster Network Configuration** dialog box, enter the IP address and subnet mask that your SQL Server 2008 cluster will use. Deselect the checkbox under the **DHCP** column as you will be using static IP addresses. If you have not disabled your IPv6 adapters and protocols, it would be better to uncheck the row for **IPv6**



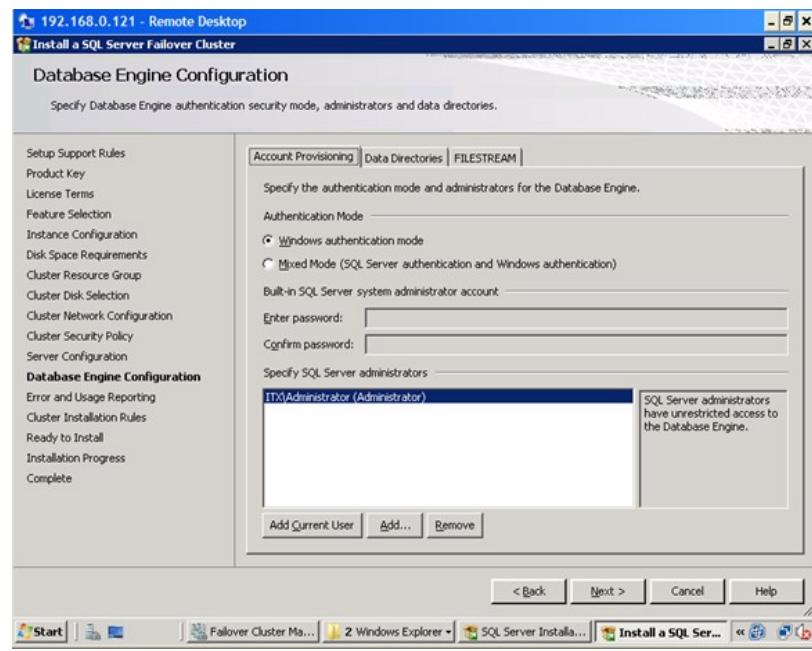
13. In the **Cluster Security Policy** dialog box, accept the default value of **Use service SIDs (recommended)**. In Windows Server 2003, we specify domain groups for all SQL Server services but in Windows Server 2008, this is the recommended option.



14. In the **Server Configuration** dialog box, enter the credentials that you will use for your SQL Server service accounts in the **Service Accounts** tab. In the **Collation** tab, select the appropriate collation to be used by SQL Server. Note that the startup type is set to manual for all cluster-aware services and cannot be changed during the installation process. Click **Next**.

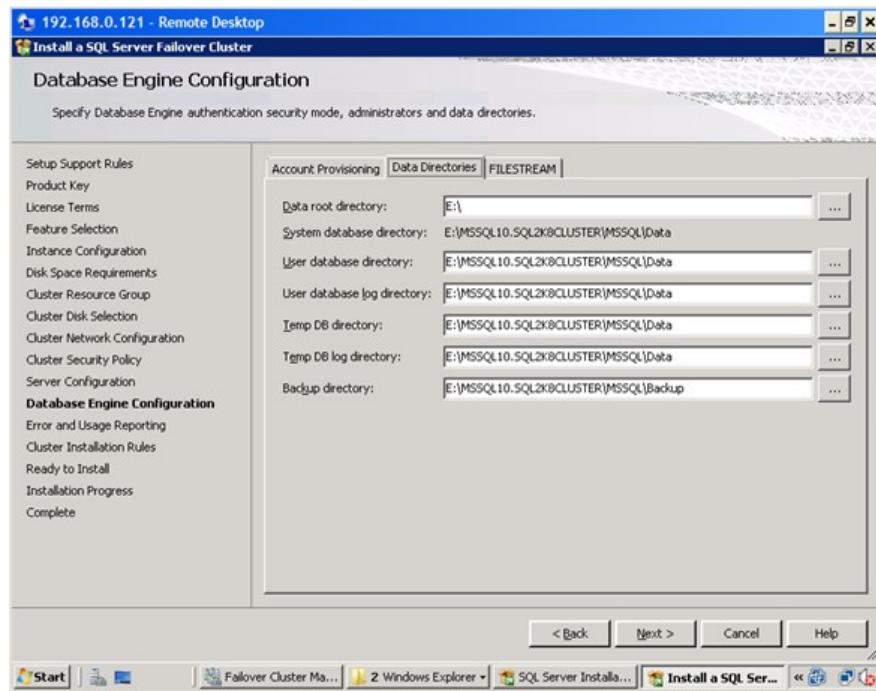


15. In the **Database Engine Configuration** dialog box, select the appropriate **Authentication Mode**. If you want to add the currently logged on user to be a part of the SQL Server administrators group, click the **Add Current User** button.

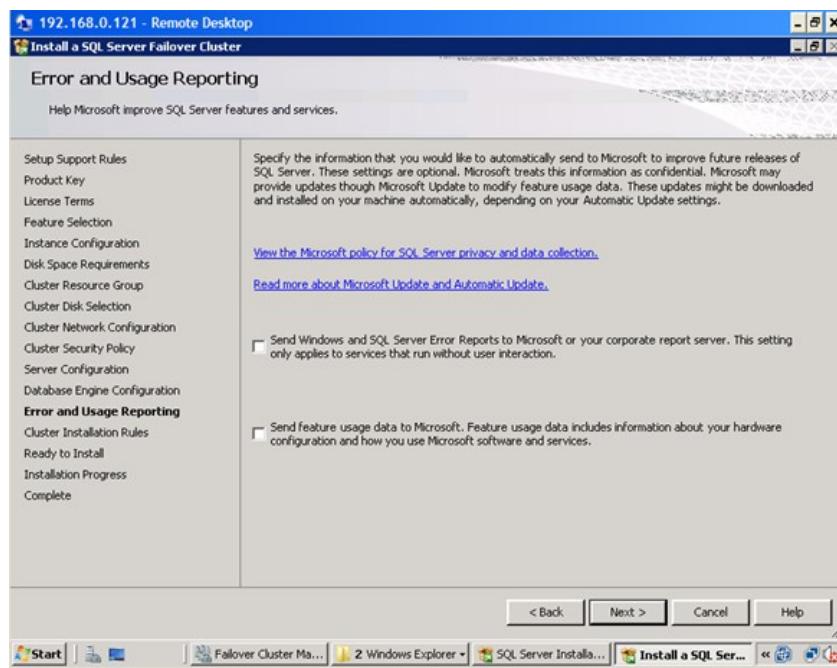


On the **Data Directories** tab, enter the path where your system and user database files will be created. This will default to the first shared disk in the cluster so in case you want to change it to the other shared disks to be used by SQL Server 2008, modify accordingly. If you intend to use

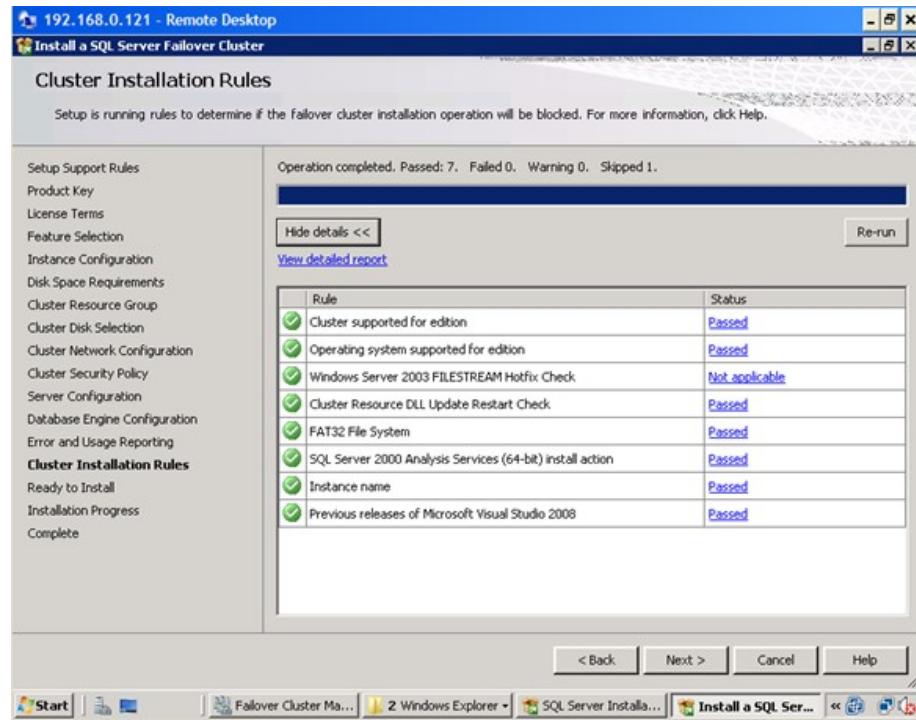
the new FILESTREAM feature, click the **FILESTREAM** tab and set the appropriate configurations. Click **Next**.



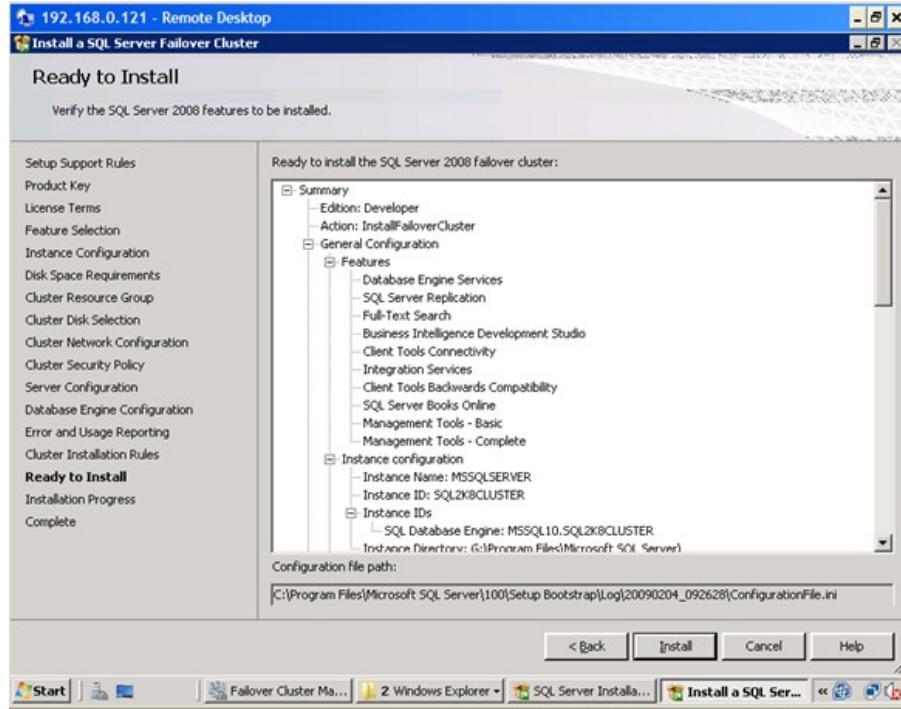
16. In the **Error and Usage Reporting** dialog box, click **Next**.



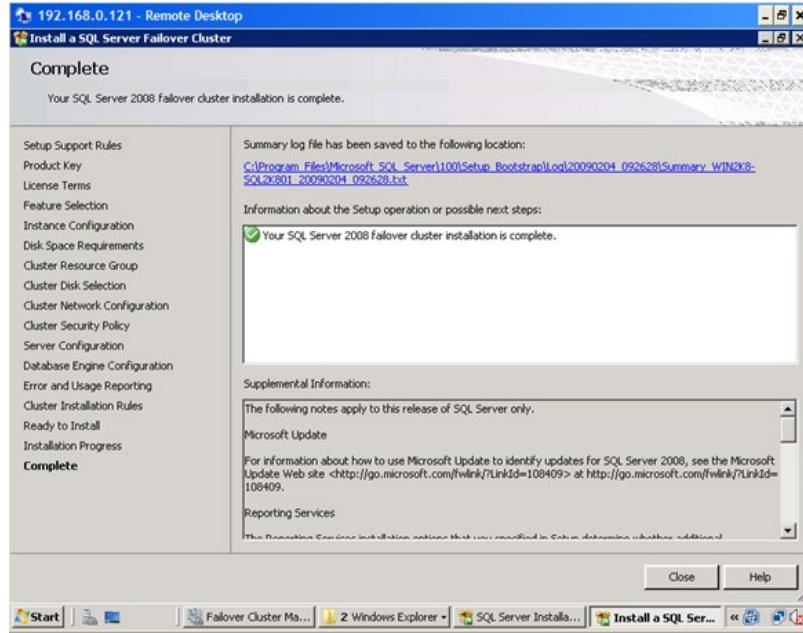
17. In the **Cluster Installation Rules** dialog box, verify that all checks are successful and click **Next**.



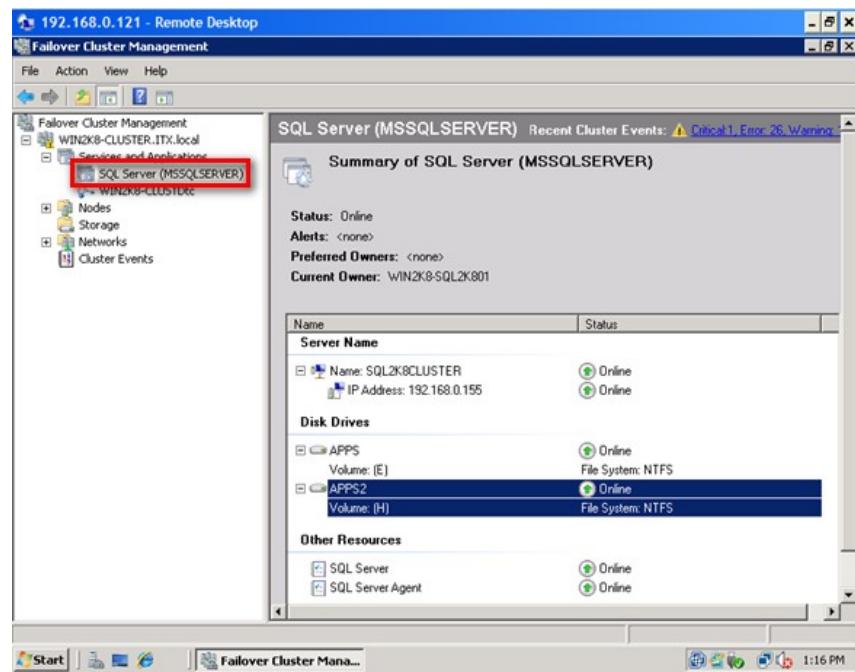
18. In the **Ready to Install** dialog box, verify that all configurations are correct. Click **Next**.



19. In the **Complete** dialog box, click **Close**. This concludes the installation of a SQL Server 2008 Failover Cluster



At the completion of a successful installation and configuration of the node, you now have a fully functional failover cluster instance. To validate, open the **Failover Cluster Management** console, and click on **SQL Server (MSSQLSERVER)** under **Services and Applications**. Make sure that all dependencies are online

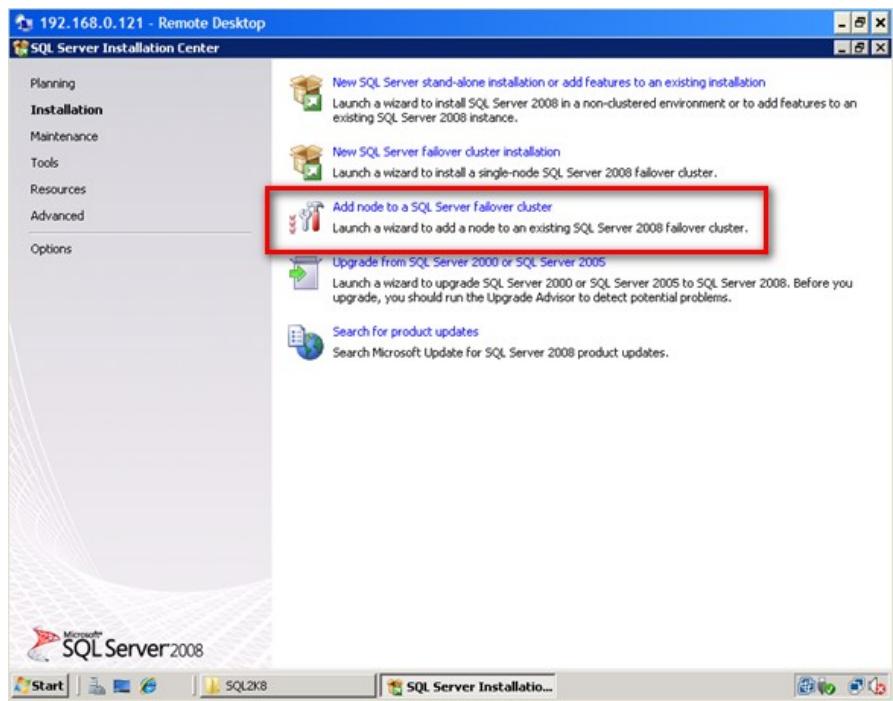


Although we do have a fully functioning SQL Server 2008 failover cluster, it does not have high-availability at this point in time because there is only one node in the failover cluster. We still have to add the second node to the SQL Server 2008 cluster. In the last part of this series, we will add the second node in the failover cluster and install the latest cumulative update.

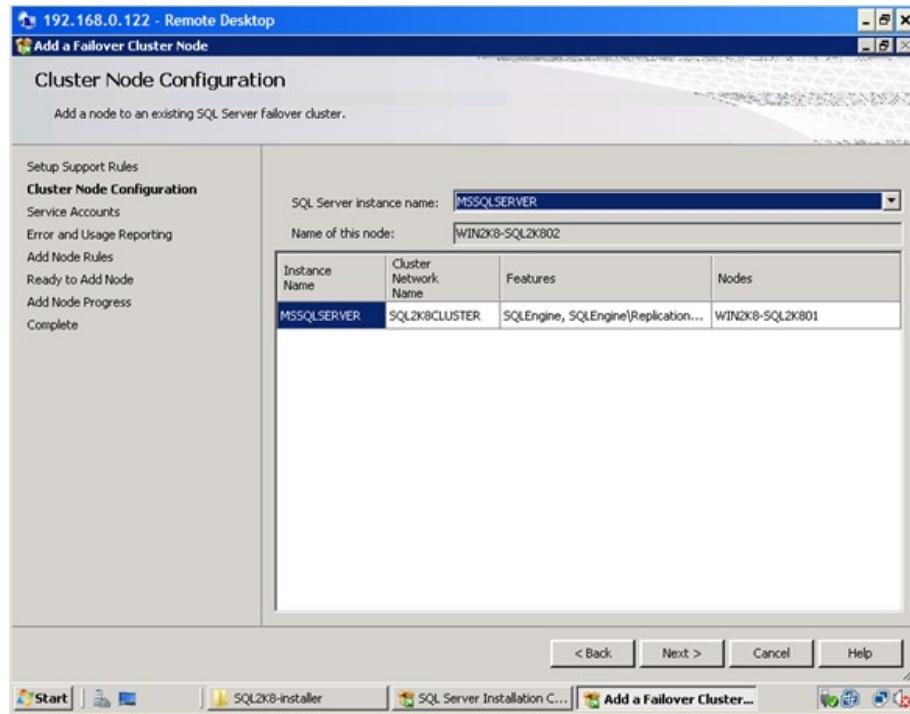
Adding a node on a SQL Server 2008 Failover Cluster

To add a node on a SQL Server 2008 failover cluster:

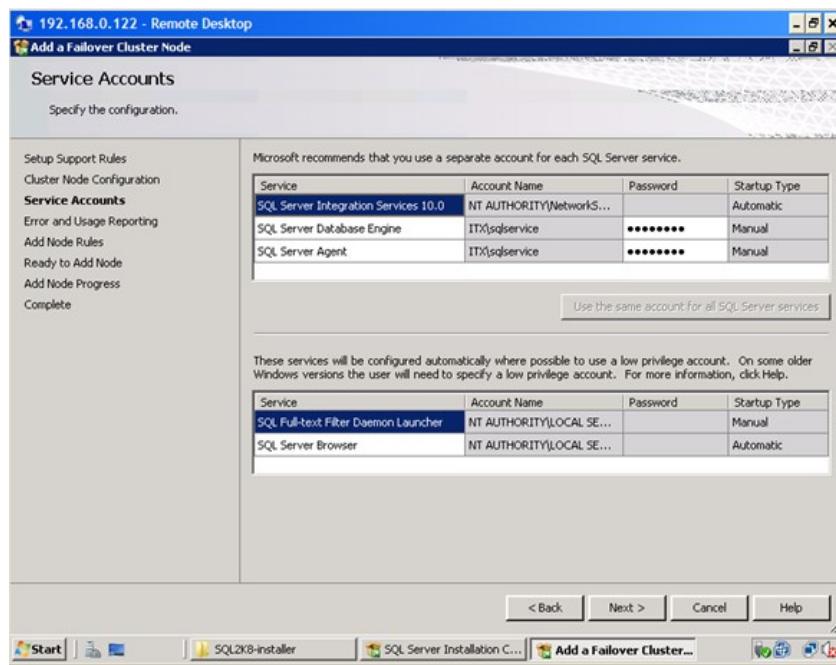
1. Run **setup.exe** from the installation media to launch **SQL Server Installation Center**
2. Click on the **Installation** link on the left-hand side. Click the **Add node to a SQL Server failover cluster** link. This will run the **SQL Server 2008 Setup** wizard. There are a couple of glitches when you get to this point. One of them is a popup error with an error message "*failed to retrieve data for this request*" while in this step. I've seen a [Microsoft Connect](#) item on this but refers to CTP6 so I was thinking it has already been resolved. After a few searches and questions asked, [SQL Server MVP Geoff Hiten](#) advised that prior to adding another node in the cluster, any cumulative update should be pre-applied to the node before the main installation as the cluster install of the RTM version has some bugs. This creates a patched install script for the RTM installer to use. The fix started with cumulative update 1 so, technically, you can apply any cumulative update. Sounds weird, but it works. You still have to apply the patch after the installation.



3. In the **Setup Support Rules** dialog box, validate that the checks return successful results and click **OK**.
4. In the **Product Key** dialog box, enter the product key that came with your installation media and click **Next**. Again, a few glitches on this step. This might seem unusual as you are only being asked about the Product Key. There is also a Microsoft Connect item for this which basically asks you to run the **setup.exe** in command prompt. There is a popup error with an error message "*The current SKU is invalid*" while in this step. This usually happens when you use a media with a supplied product key, like the one that comes with an MSDN subscription. What worked for me was to copy the installation media on a local disk locate the file **DefaultSetup.ini** file from the installation files and delete it or move it to different location. If you opt to delete the file, make sure you note down the product key written on this file as you will need to manually key this in during the installation process.
5. In the **License Terms** dialog box, click the **I accept the license terms** check box and click **Next**.
6. In the **Setup Support Rules** dialog box, click **Install**. Validate that the checks return successful results. Again, make sure to fix any errors returned by this check before proceeding with the installation.
7. In the **Cluster Node Configuration** dialog box, validate that the information for the existing SQL Server 2008 cluster is correct.

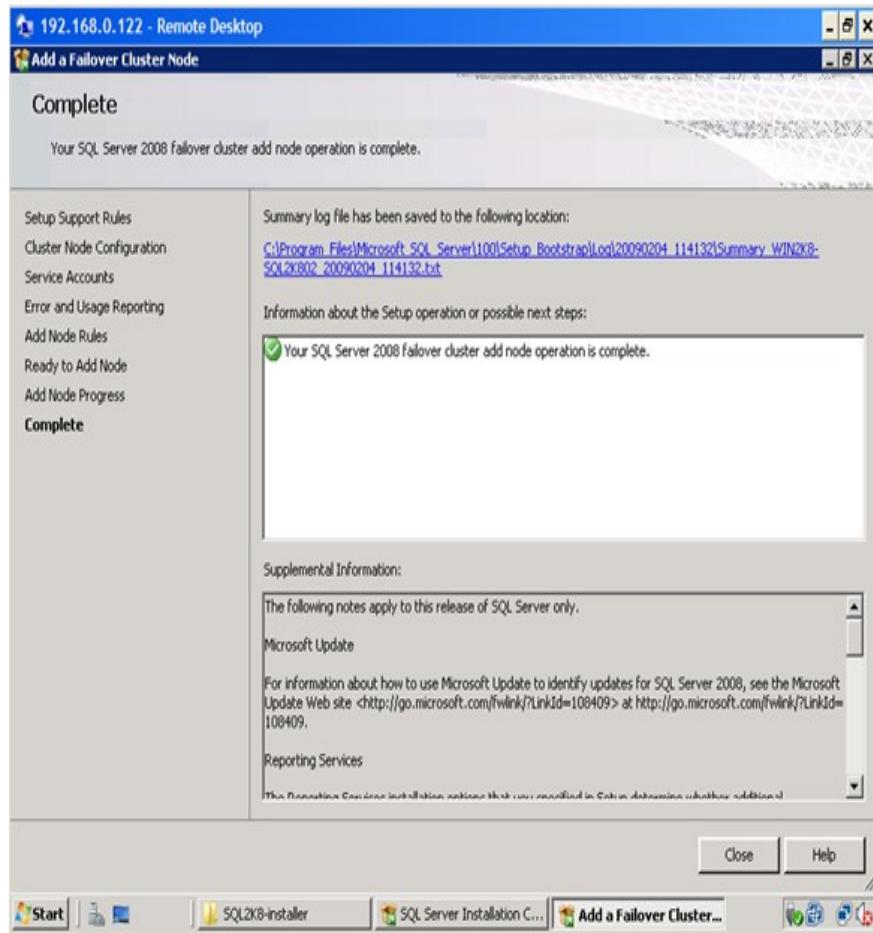


- In the **Service Accounts** dialog box, verify that the information is the same as what you have used to configure the first node.

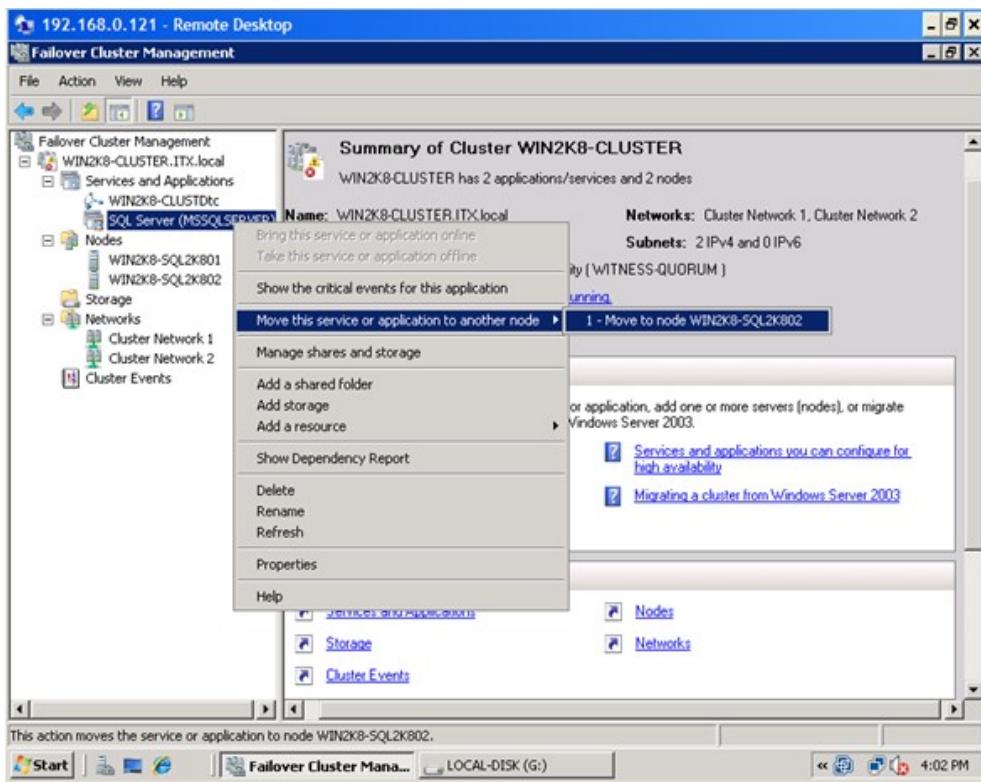


- In the **Error and Usage Reporting** dialog box, click **Next**
- In the **Add Node Rules** dialog box, verify that all checks are successful and click **Next**
- In the **Ready to Add Node** dialog box, verify that all configurations are correct and click **Install**

12. In the **Complete** dialog box, click **Close**. This concludes adding a node to a SQL Server 2008 Failover Cluster



You can validate your cluster installation by expanding the **Services and Applications** node and check the cluster name of your SQL Server instance. You can now see an option to move the service to another node, in this case, the node you've just added in your failover cluster

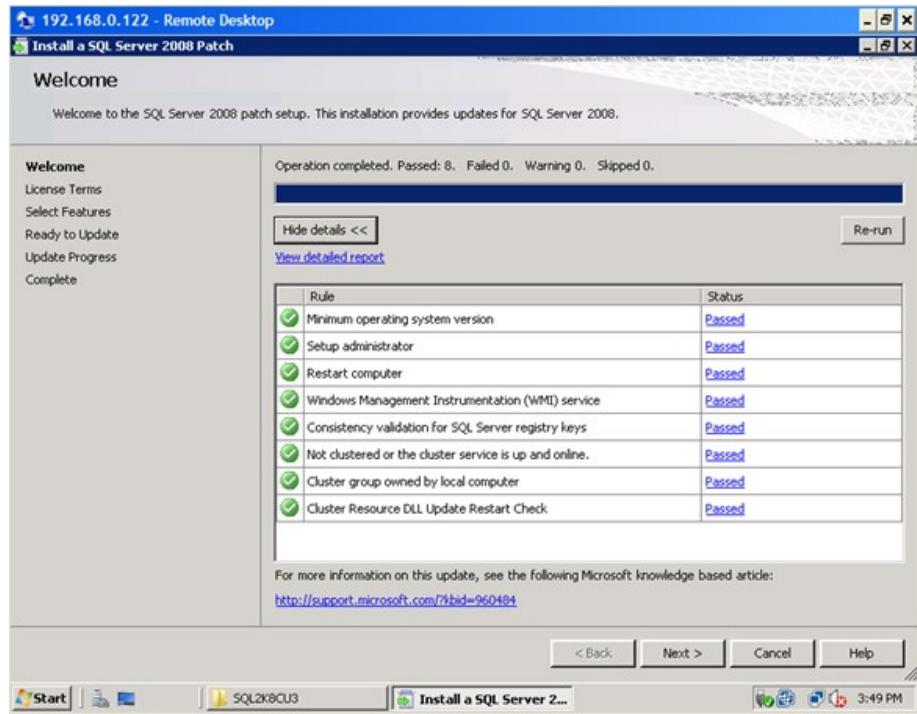


Applying patches on a SQL Server 2008 cluster

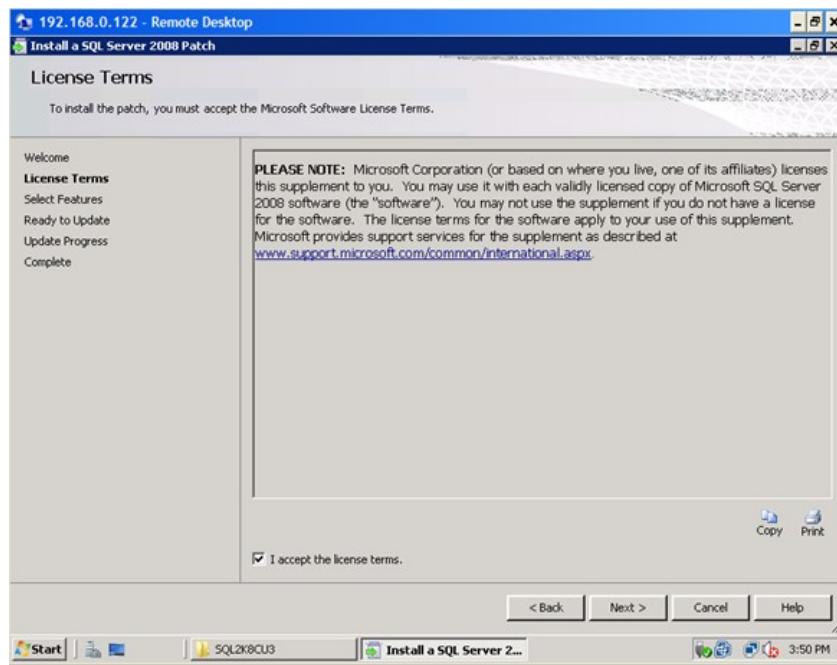
Part of the tasks of a DBA is to apply patches on the database engine and a SQL Server 2008 failover cluster is no exception. In fact, it is not as straight-forward as applying patches and service packs on a stand-alone server. It is important to note that when applying patches or service packs to a SQL Server failover cluster, you should apply them first on the passive node. After completing the installation on the passive node, failover the SQL Server 2008 cluster resource to this node making it the active node. Once the SQL Server service and all other dependencies are up, you can, then, apply the patches on the new passive node. The latest available patch for SQL Server 2008 is cumulative update 4 and is available for request from Microsoft. For more information, check out this [Microsoft KB article](#). You will have to request for the patch from Microsoft as it is not available from the Microsoft Download Center. The screenshots below show cumulative update 3 (version **10.0.1600.22**) but the process is basically the same. Also, note that even though you may have already applied the cumulative update due to the bug mentioned above for adding a node in a failover cluster, you still have to apply the patch on both nodes

To apply patches on a SQL Server 2008 failover cluster node:

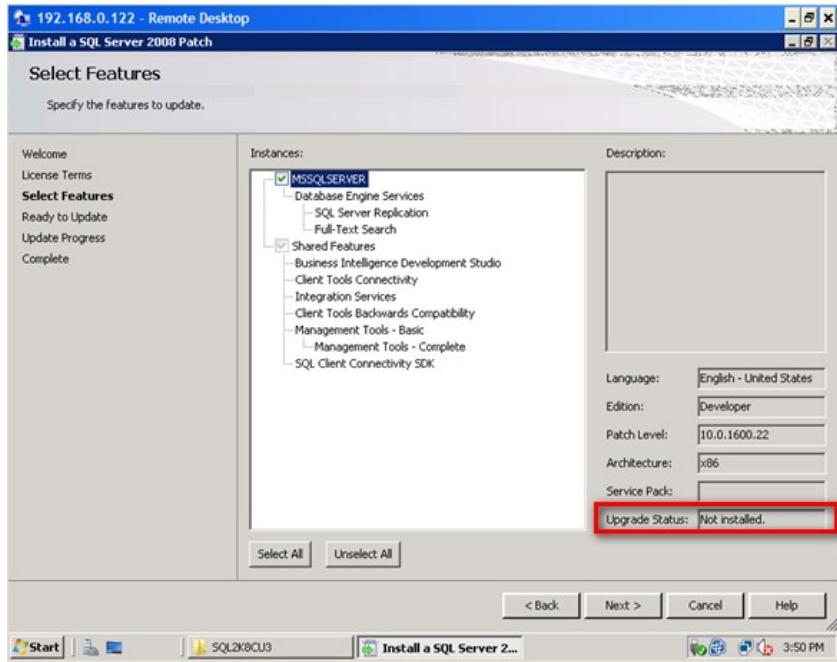
1. Run **SQLServer2008-KB960484-x86.exe** (this would depend on the cumulative update that you want to apply) from the hotfix package you have requested from Microsoft
2. In the **Welcome** dialog box, validate that the checks return successful results.



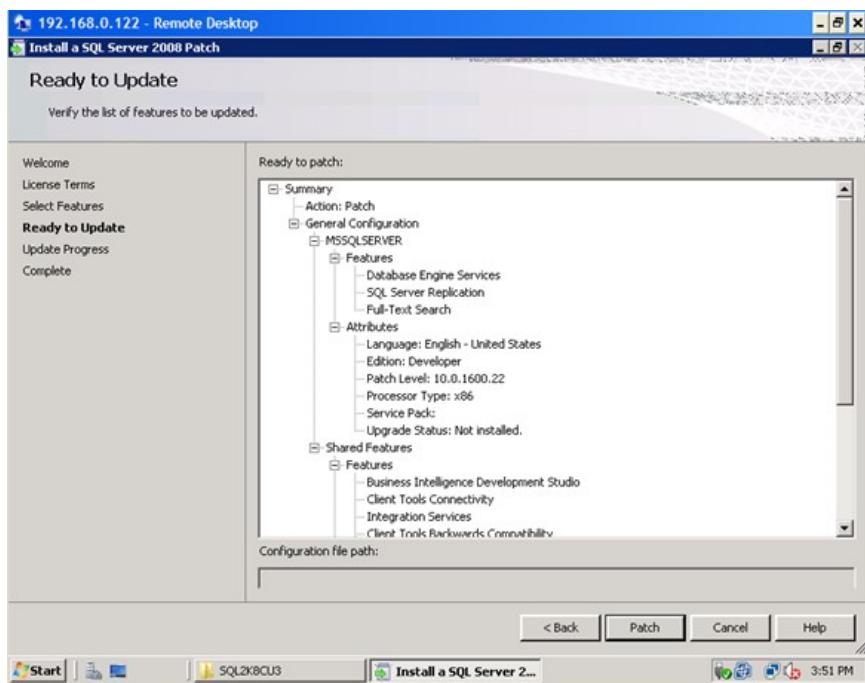
- In the **License Terms** dialog box, click the **I accept the license terms** check box and click **Next**



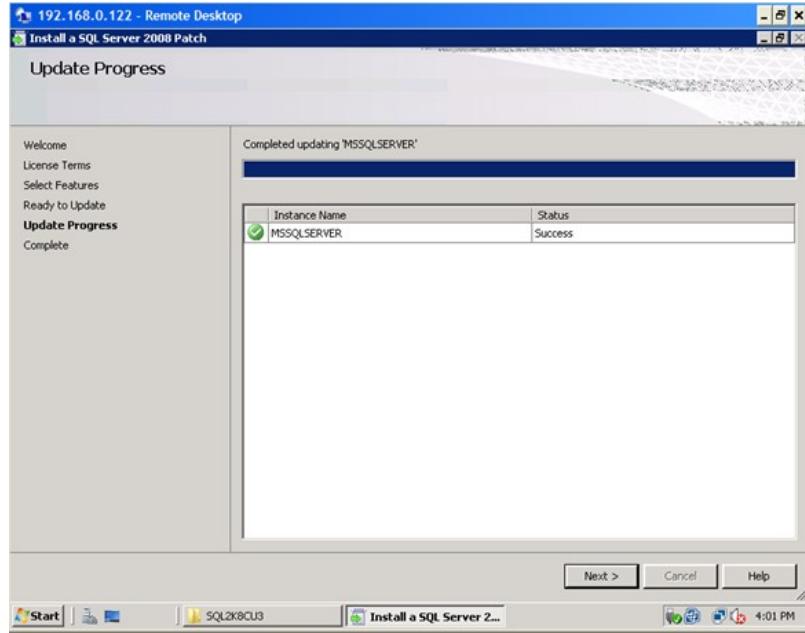
- In the **Select Features** dialog box, validate the SQL Server 2008 components by clicking on the check box. The **Upgrade Status** field will tell you whether or not the patch has already been applied. Click **Next**



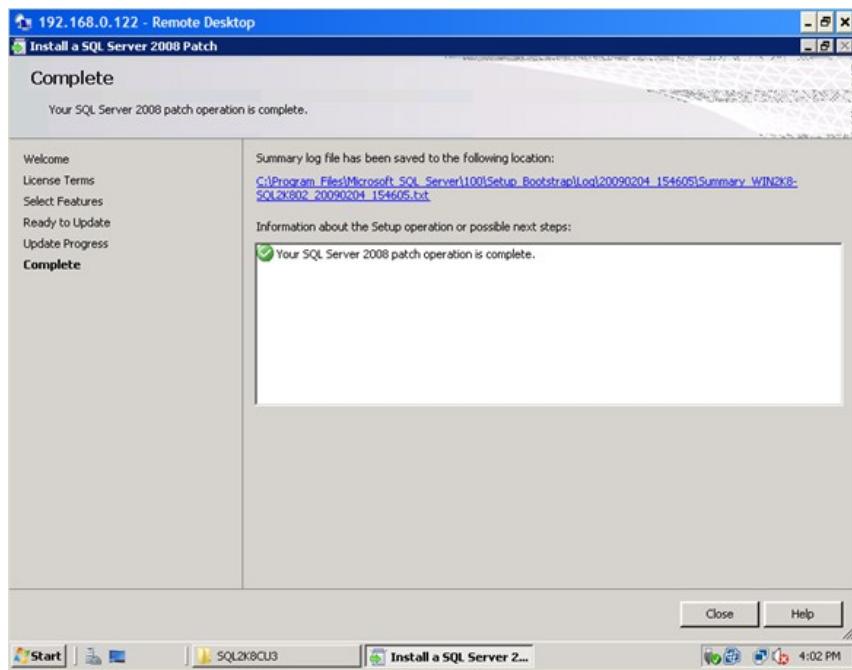
5. In the **Ready to Update** dialog box, verify that all configurations are correct and click **Patch**



6. In the **Update Progress** dialog box, validate that the installation was successful.



7. In the **Complete** dialog box, click Close. This concludes patching the passive node of a SQL Server 2008 Failover Cluster



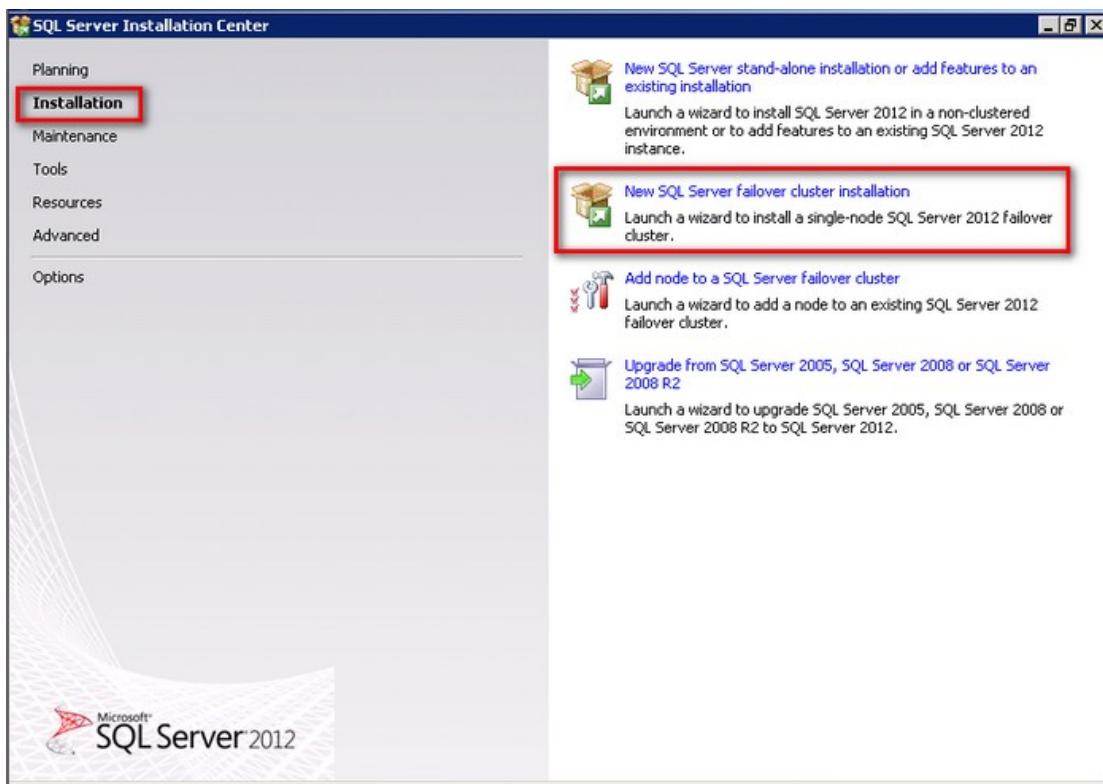
After successfully installing the patch on the passive node, move the SQL Server 2008 cluster resource to this node so it will become the new active node. Make sure that all the SQL Server 2008 cluster dependencies are online prior to applying the patch on the other node. Repeat the process outlined above to the new passive node.

Install SQL Server 2012 on a Multi-Subnet Failover Cluster

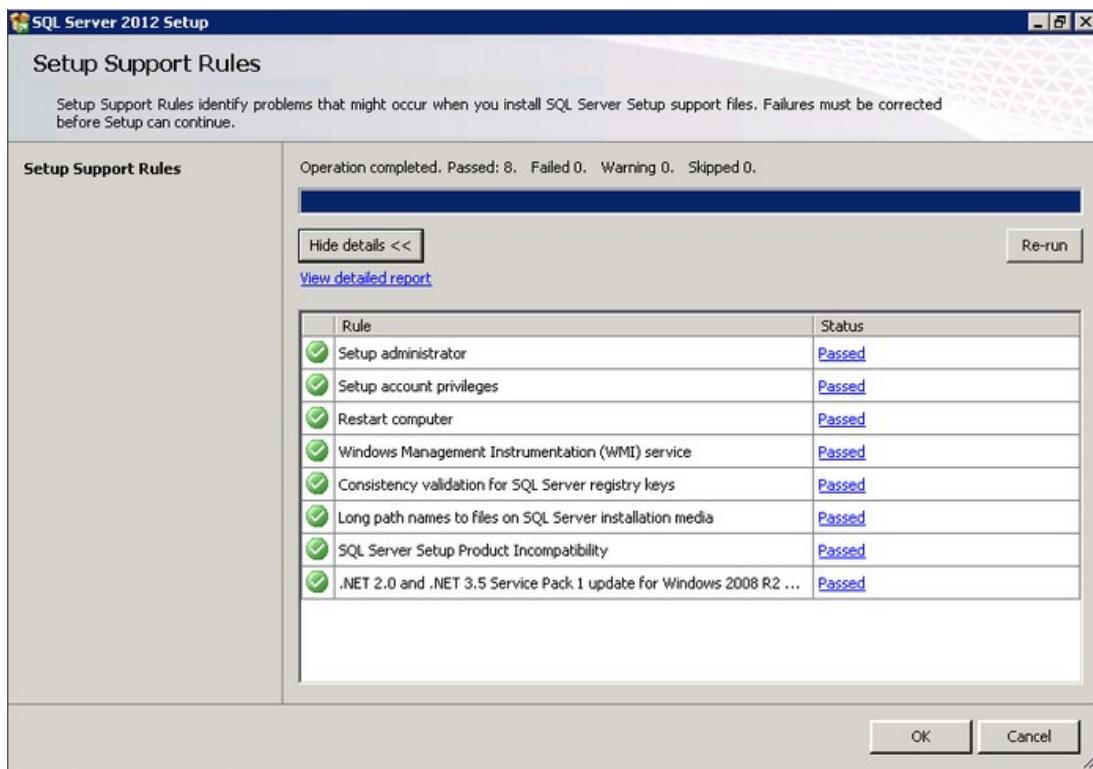
Similar to SQL Server 2008 Failover Clustering, there are two options to install SQL Server 2012 on a multi-subnet cluster. The first one is by using the Integrated failover cluster install with the Add Node option and the second one is the Advanced/Enterprise installation option. The process outlined below will take into account the first option.

Here is how to install SQL Server 2012 on a multi-subnet cluster:

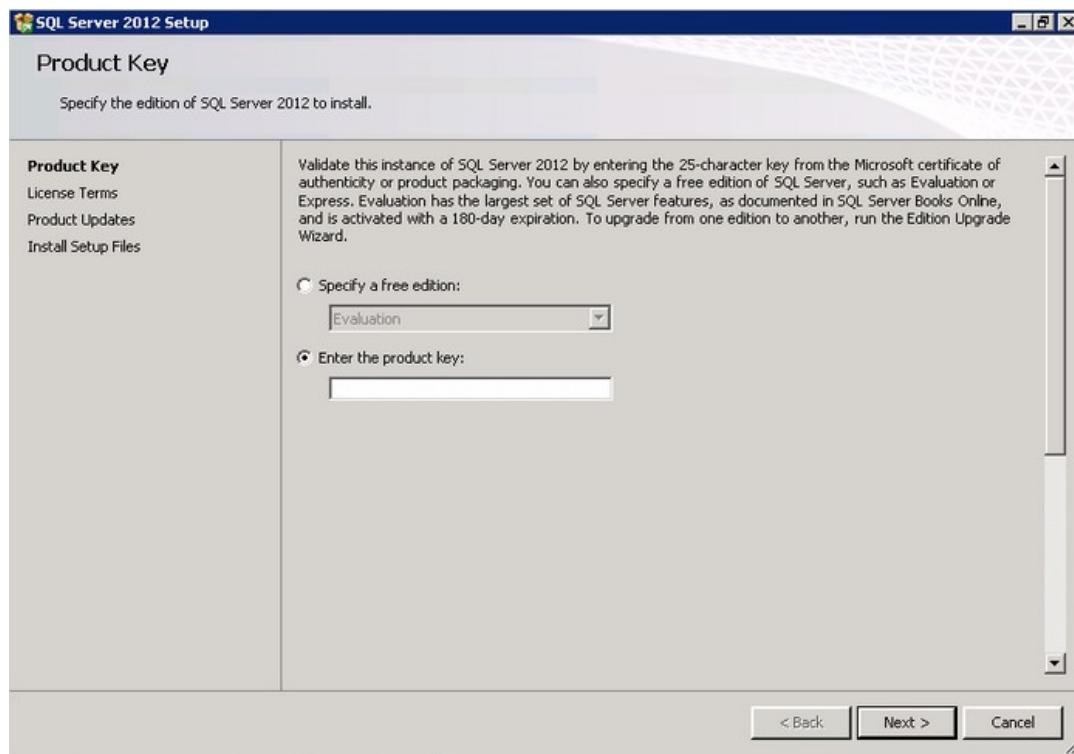
1. Run **setup.exe** from the installation media to launch SQL Server Installation Center. Click on the **Installation** link on the left-hand side
2. Click the **New SQL Server failover cluster installation** link. This will run the SQL Server 2012 Setup wizard



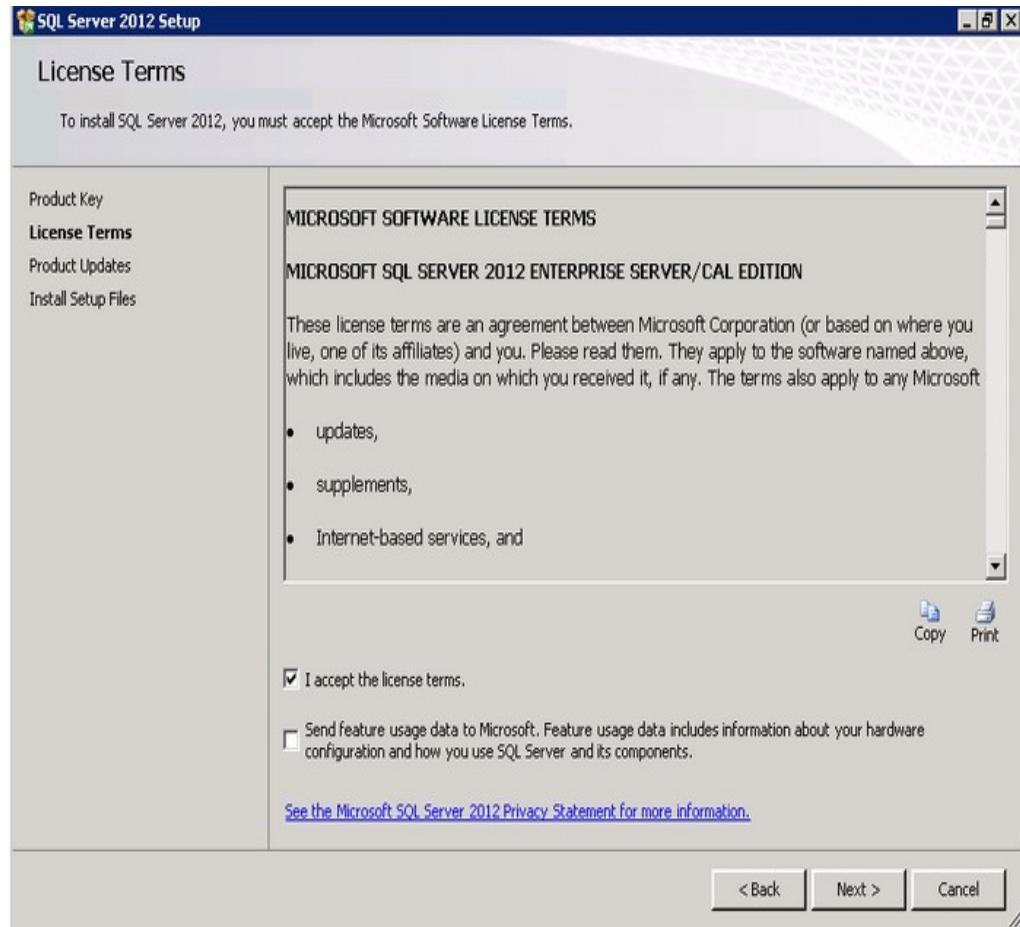
3. In the **Setup Support Rules** dialog box, validate that the checks return successful results and click **Next**.



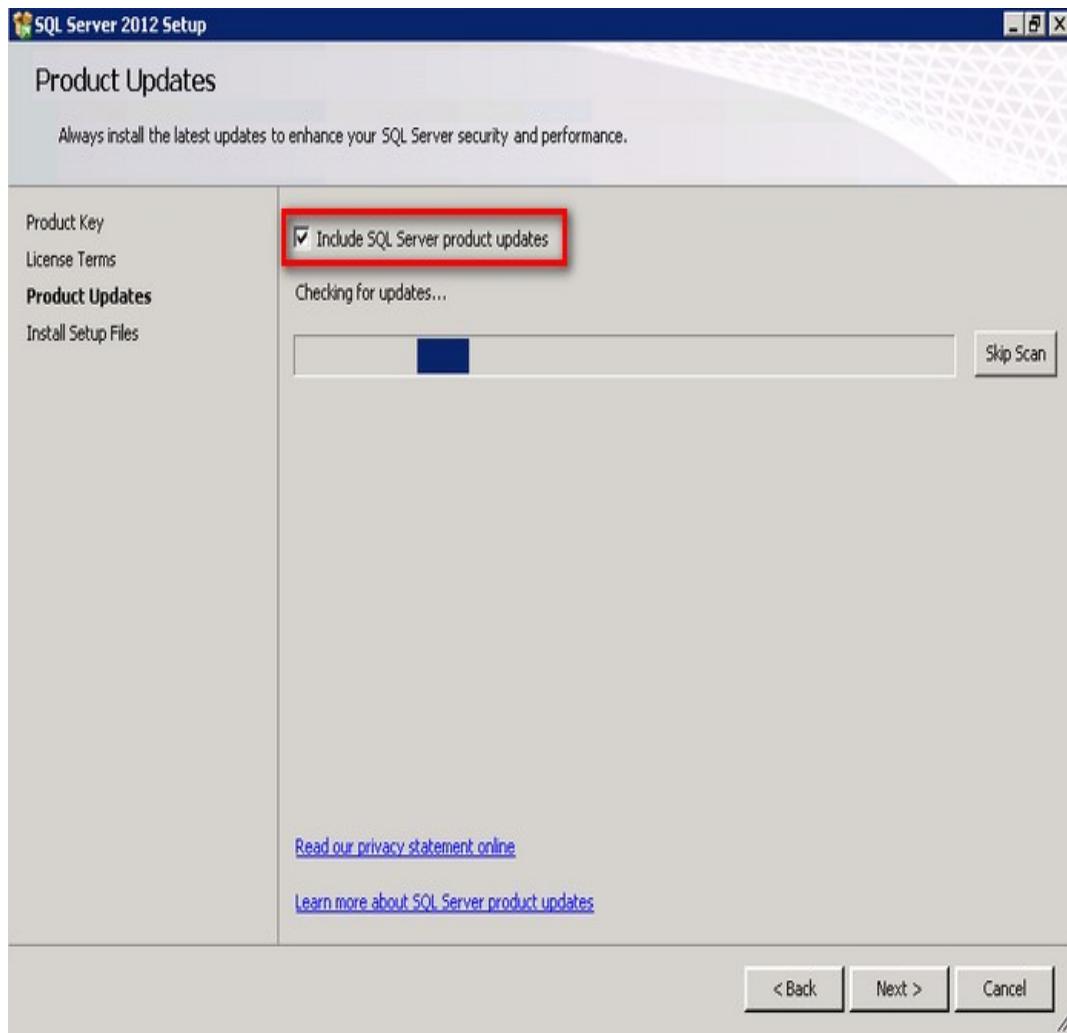
4. In the **Product Key** dialog box, enter the product key that came with your installation media and click **Next**.



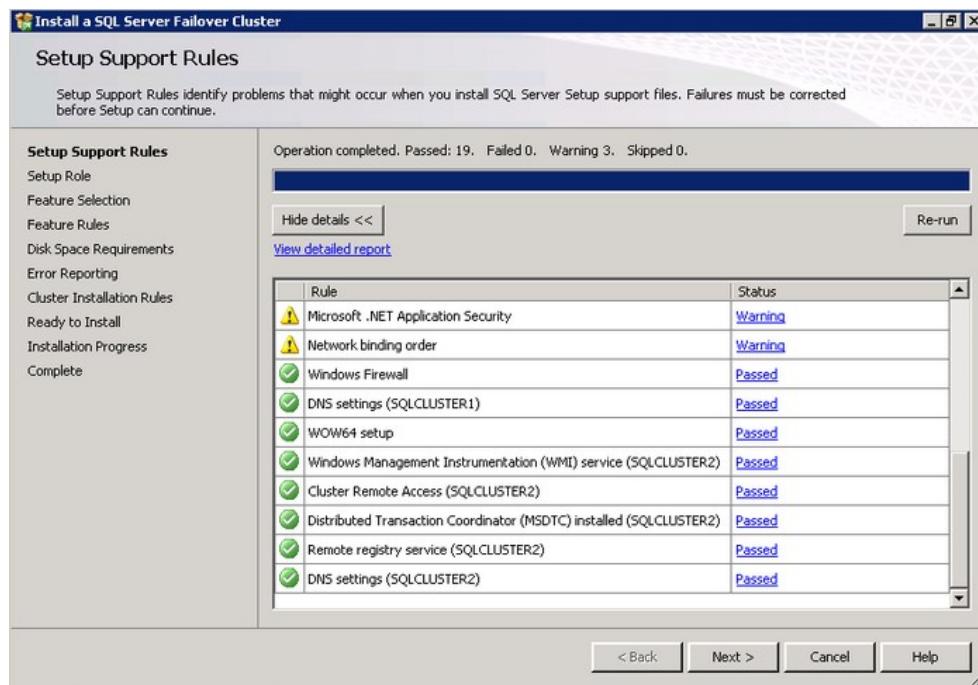
5. In the **License Terms** dialog box, click the **I accept the license terms** check box and click **Next**. And, don't worry, you're not the only one NOT reading the EULA.



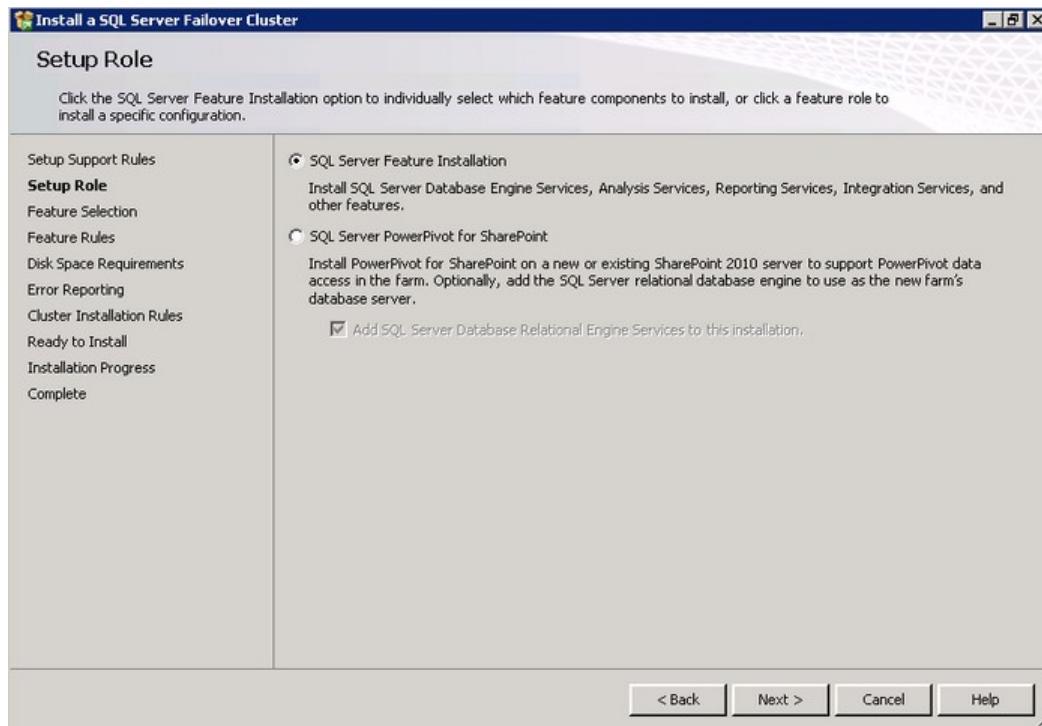
6. In the **Product Updates** dialog box, you have the option to include SQL Server product updates like service packs and cumulative updates in the installation process. This is a new feature in SQL Server 2012 that integrates the latest updates in the installation process. By default, it searches for product updates thru the Windows Updates service online, assuming that the server has access to the Internet. In cases where your servers do not have access to the internet, you can manually download the updates and store them on a network shared folder. You can, then, point the installation media to search the network shared folder instead. A more detailed approach to using this feature is outlined in this [Microsoft TechNet article](#). Click **Next**.



7. In the **Setup Support Rules** dialog box, validate that the checks return successful results. If the checks returned a few warnings, make sure you fix them before proceeding with the installation. An example of this is the Network binding order. The public network cards should be first on both nodes. You can also disable NETBIOS and DNS registration on the heartbeat and iSCSI network cards to avoid additional overhead. Be sure to check your binding order as well. For more details on the network binding order warning, see [Microsoft KB 955963](#). This [blog](#) post also explains what the network binding order rule is for and how you can further configure it. Also, if you decide not to configure a clustered MSDTC resource, it will be flagged as a warning in this dialog box. However, it's not a show stopper and you can proceed with the installation. Click **Next**.

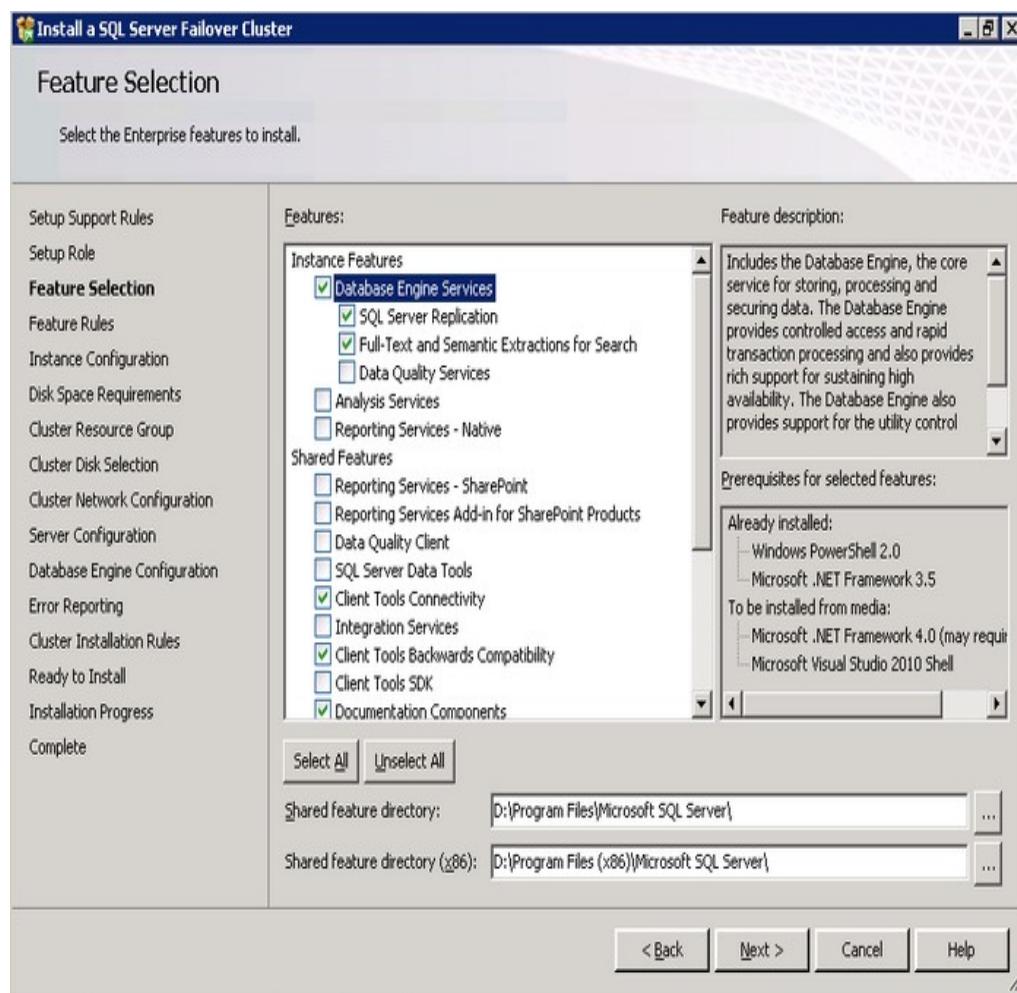


8. In the **Setup Role** dialog box, select the **SQL Server Feature Installation** option and click **Next**.

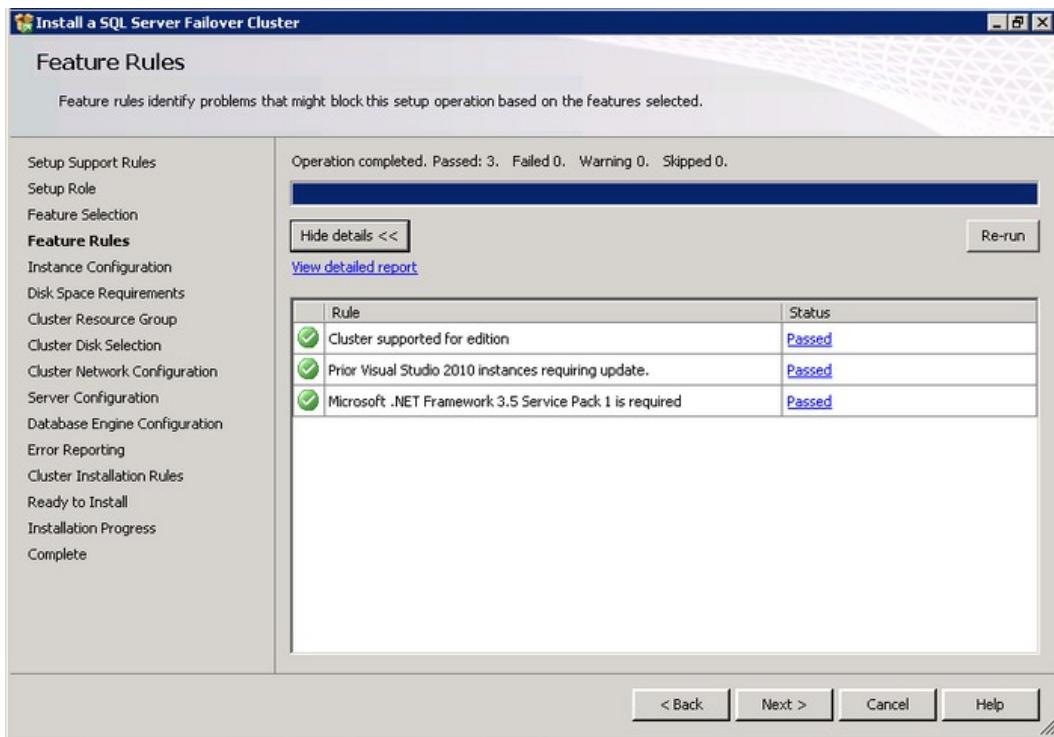


9. In the **Feature Selection** dialog box, select only the components that you want installed. For the **Shared feature directory**, you can keep the default path if you have sufficient disk space on your C:\ drive or anywhere that is a local disk as this will be used by the SQL Server

installation process later on. The directory for the clustered database engine will be different. In my example, I have another local drive available - drive D:\ - where I will store the **tempdb** database. We'll discuss more about that later. Click **Next**.

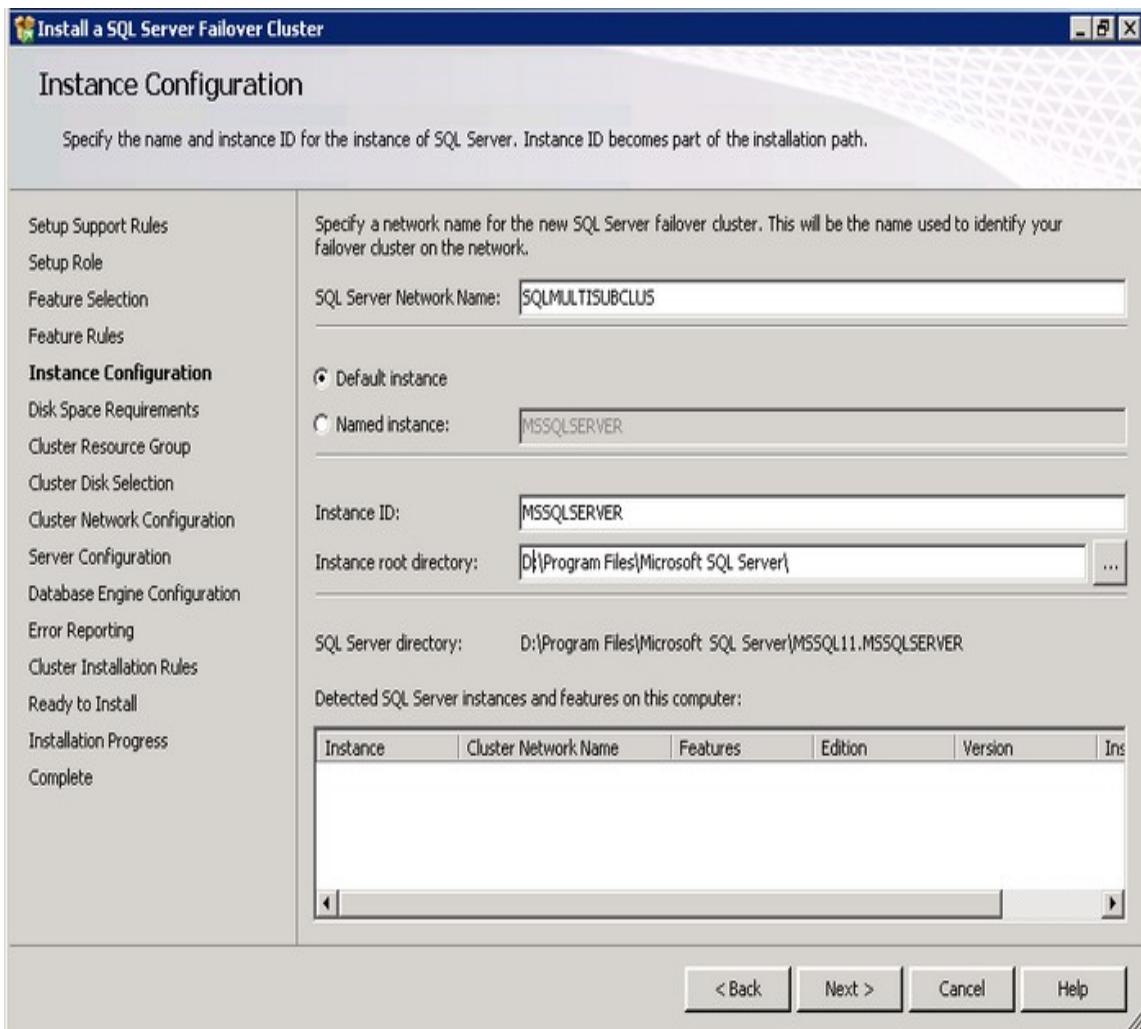


10. In the **Feature Rules** dialog box, verify that all the rules have passed. If the rules returned a few warnings, make sure you fix them before proceeding with the installation. Click **Next**.

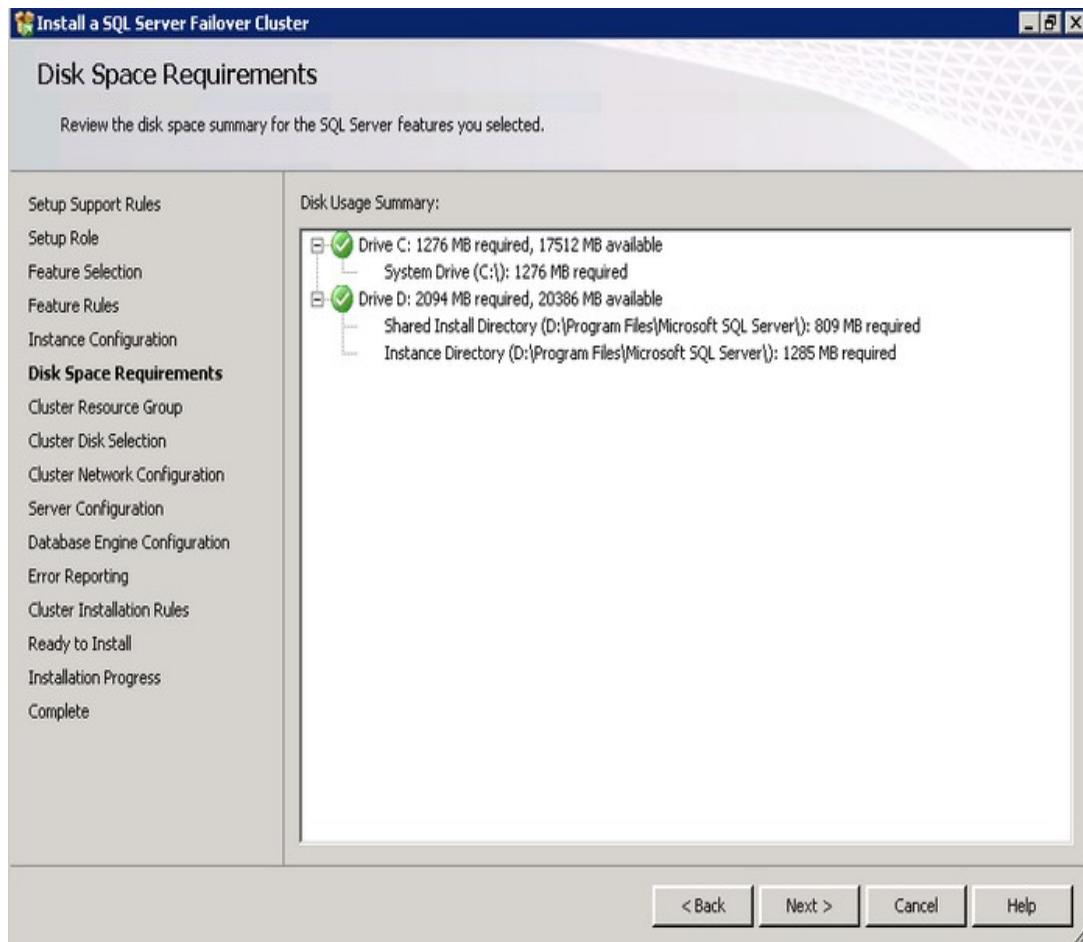


11. In the **Instance Configuration** dialog box, enter the SQL Server Network Name. This is the name that will be available on the network for the clients to access. This will vary depending on your selection of whether it is a default or named instance. In this example, default instance is selected. A couple of things need highlighting in this section. By default, the instance name is used as the Instance ID. This is used to identify installation directories and registry keys for your instance of SQL Server and is helpful when you want to run multiple instances in a cluster. This is the case for default instances and named instances. For a default instance, the instance name and instance ID would be MSSQLSERVER. To use a non-default instance ID, you should select the **Instance ID** box and specify a value.

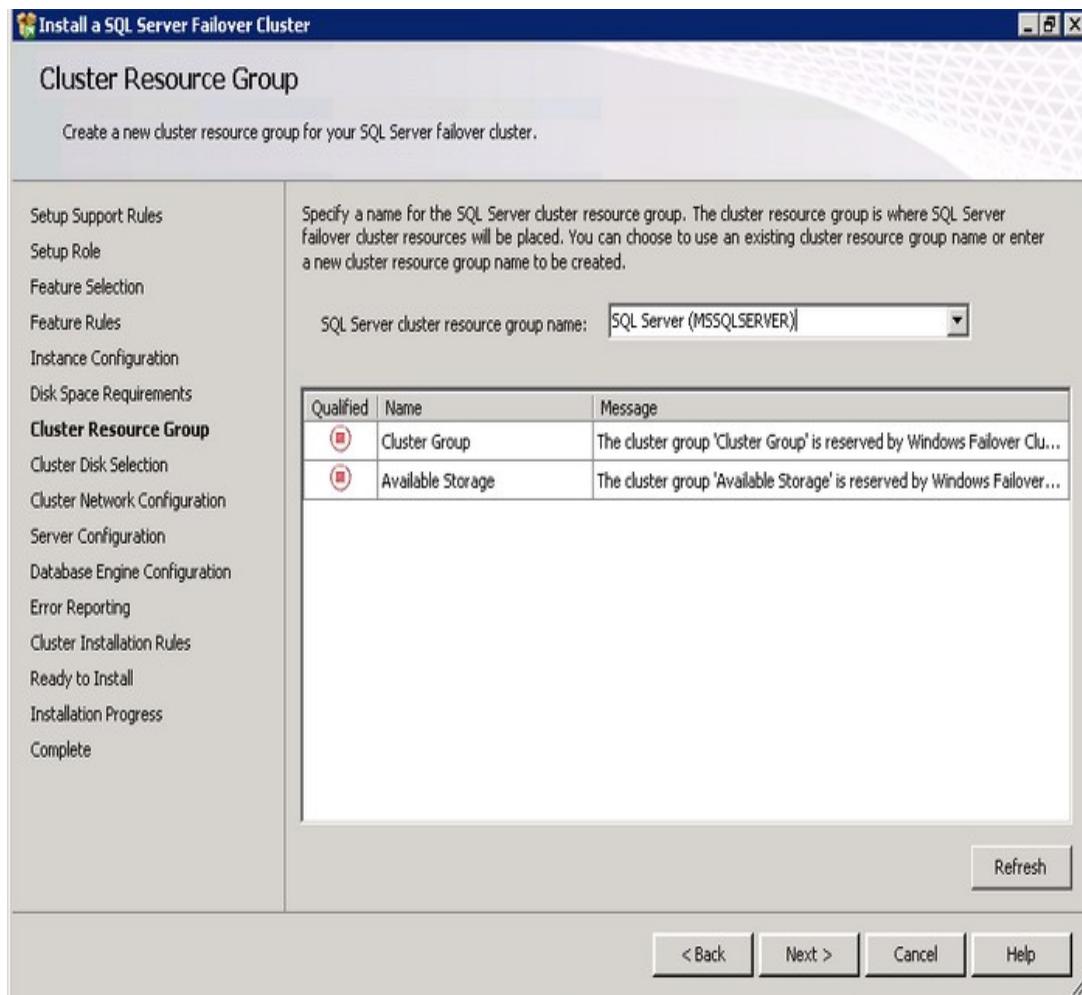
The section on **Detected SQL Server instances and features on this computer** would make sense if there are other SQL Server instances running on your server. Click **Next**.



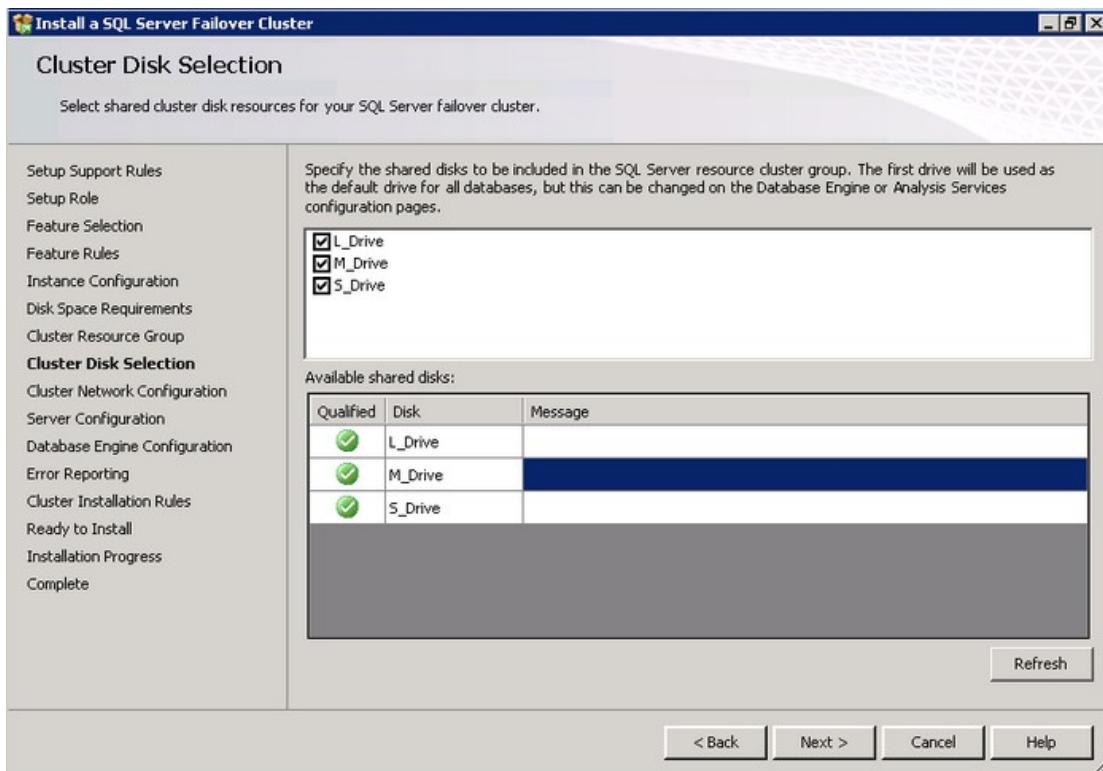
12. In the **Disk Space Requirements** dialog box, check that you have enough space on your local disks to install the SQL Server 2012 binaries. Click **Next**.



13. In the **Cluster Resource Group** dialog box, check the resources available on your Windows Server 2008 R2 cluster. This tells you that a new Resource Group will be created on your cluster for the SQL Server instance. To specify the SQL Server cluster resource group name, you can either use the drop-down box to specify an existing group to use or type the name of a new group to create it. Click **Next**.



14. In the **Cluster Disk Selection** dialog box, select the available disk groups that are on the cluster for SQL Server 2012 to use. In this example, three clustered disk groups – L_Drive, M_Drive and S_Drive – have been selected to be used by SQL Server 2012. I will be using one disk resource for the system databases, one for the data files and one for the log files. Click **Next**.

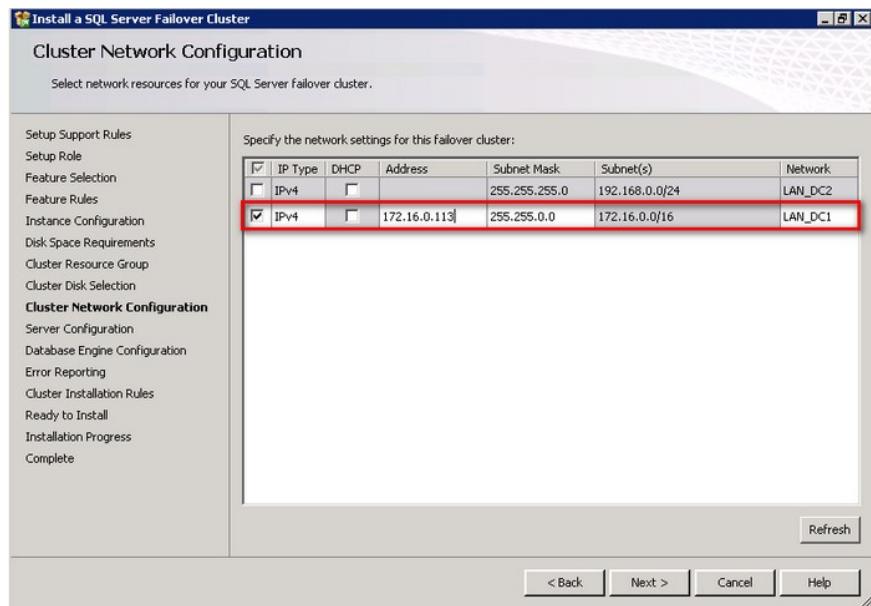


15. In the **Cluster Network Configuration** dialog box, enter the virtual IP address and subnet mask that your SQL Server 2012 cluster will use. Notice that the setup process has detected the existence of two network subnets - **LAN_DC1** and **LAN_DC2**. These are the names of the network adapters that I have defined in my Windows Server 2008 R2 Failover Cluster. Since you are performing the installation on a cluster node that belongs to one of the network subnets, only that option will be available. The other option to assign a virtual IP address will be made available to you when you run the Add Node option.

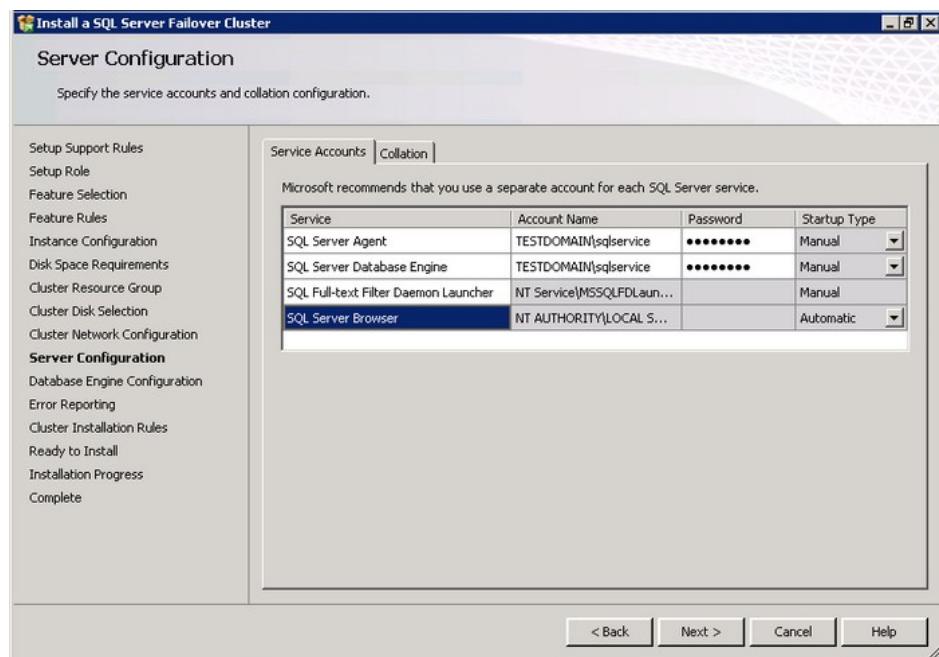
We will be using the following information for our SQL Server failover cluster instance.

Virtual Server Name	Networks	IP Address
SQLMULTISUBCLUS	172.16.0.0/24	172.16.0.113
	192.168.0.0/24	192.168.0.113

Deselect the checkbox under the DHCP column as you will be using static IP addresses. Click **Next**.



16. In the **Server Configuration** dialog box, enter the credentials that you will use for your SQL Server service accounts in the **Service Accounts** tab. In the **Collation** tab, select the appropriate collation to be used by SQL Server. Note that the startup type is set to manual for all cluster-aware services and cannot be changed during the installation process. The startup type behavior of the SQL Server services will be controlled by the cluster resource group. Click **Next**.

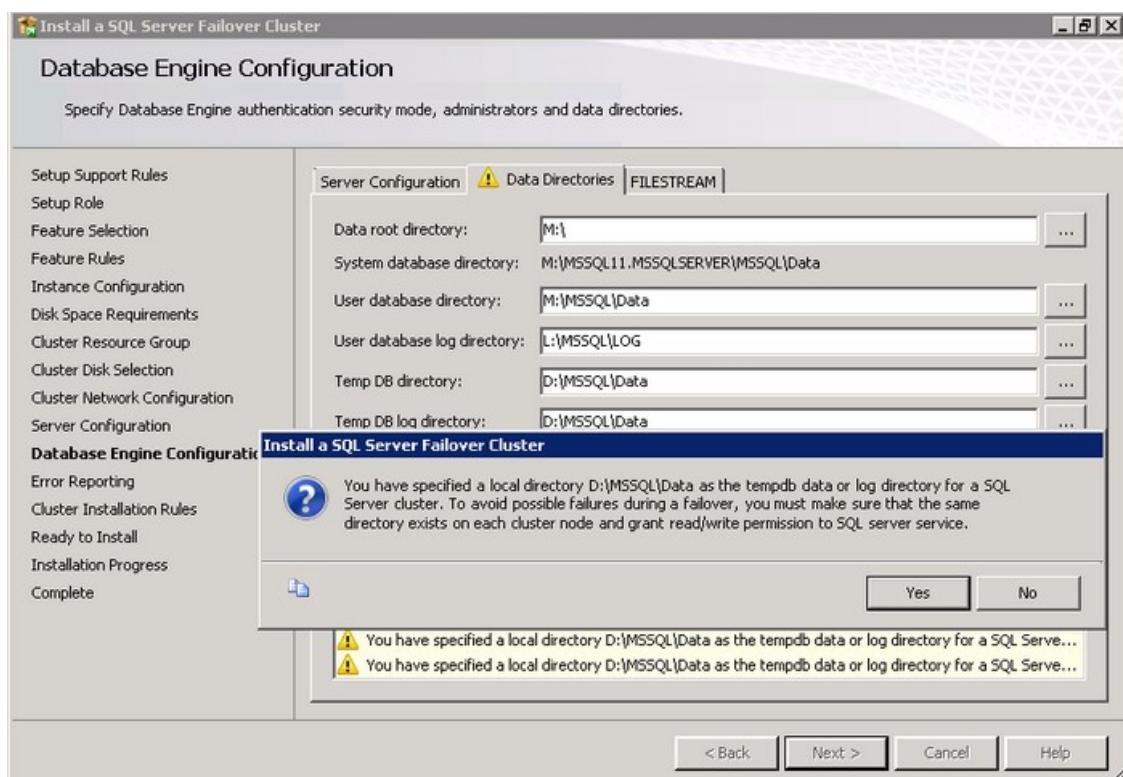


17. In the **Database Engine Configuration** dialog box, select the appropriate **Authentication Mode** in the **Server Authentication** tab. If you want to add the currently logged on user to be

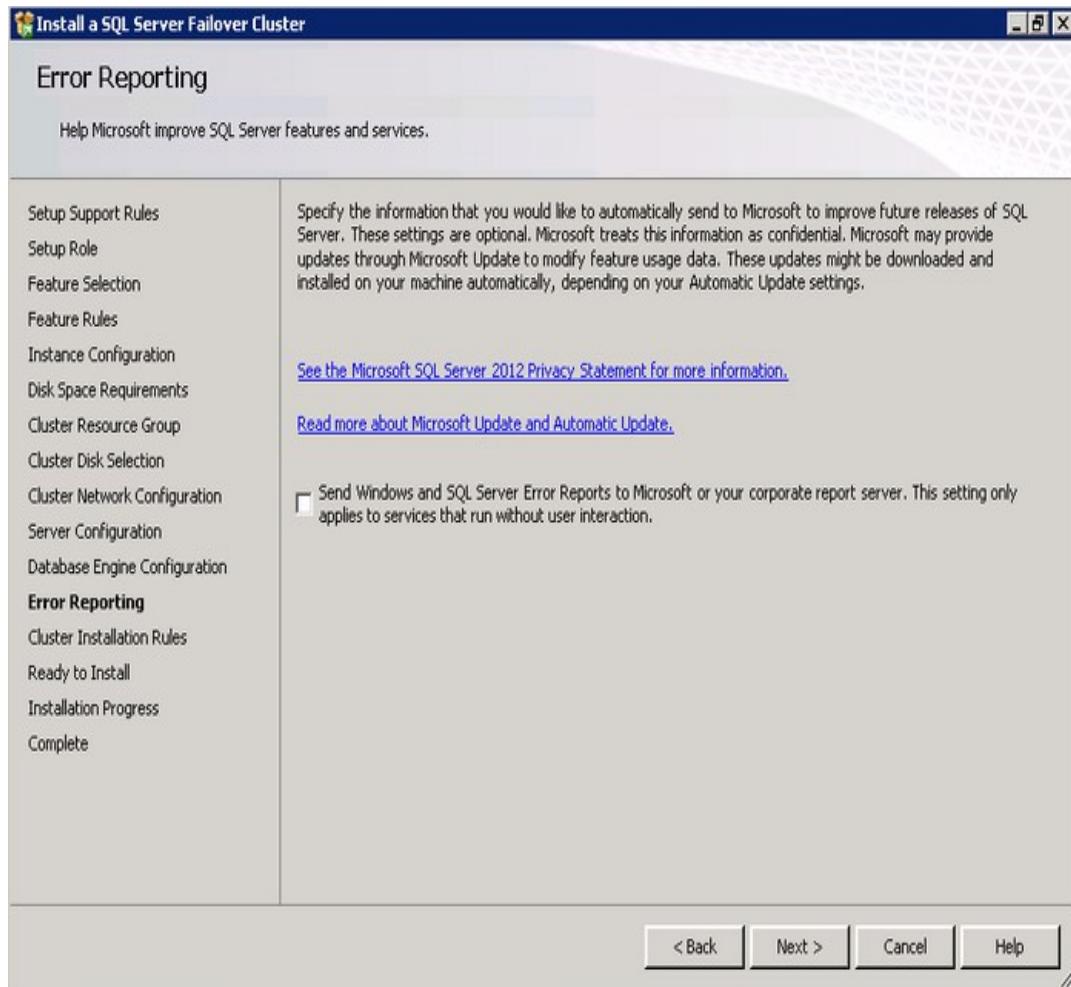
a part of the SQL Server administrators group, click the **Add Current User** button. Otherwise, you can add the appropriate domain accounts or security groups.

What I'd really want you to focus your attention on is the **Data Directories** tab. Notice that I used the local drive D:\ for my **tempdb** database files. This is a new feature in SQL Server 2012 where you can now store your tempdb database on a local disk in a Windows Failover Cluster. You can still choose to host the tempdb database on a clustered storage. But do you want to know why having the tempdb database in local disks is a good thing for a SQL Server Failover Clustered instance? This is because tempdb is a scratch database that gets recreated every time the SQL Server service is started. And since a cluster resource failover is simply a service stop-start process that is being controlled by the cluster resource group, tempdb gets recreated whenever you do a failover on all of the nodes of the cluster. In a multi-subnet cluster that spans across geographically dispersed data centers, replicating the storage that hosts the tempdb database doesn't make sense. It's just wasted bandwidth and IO resources that will get thrown away during failover.

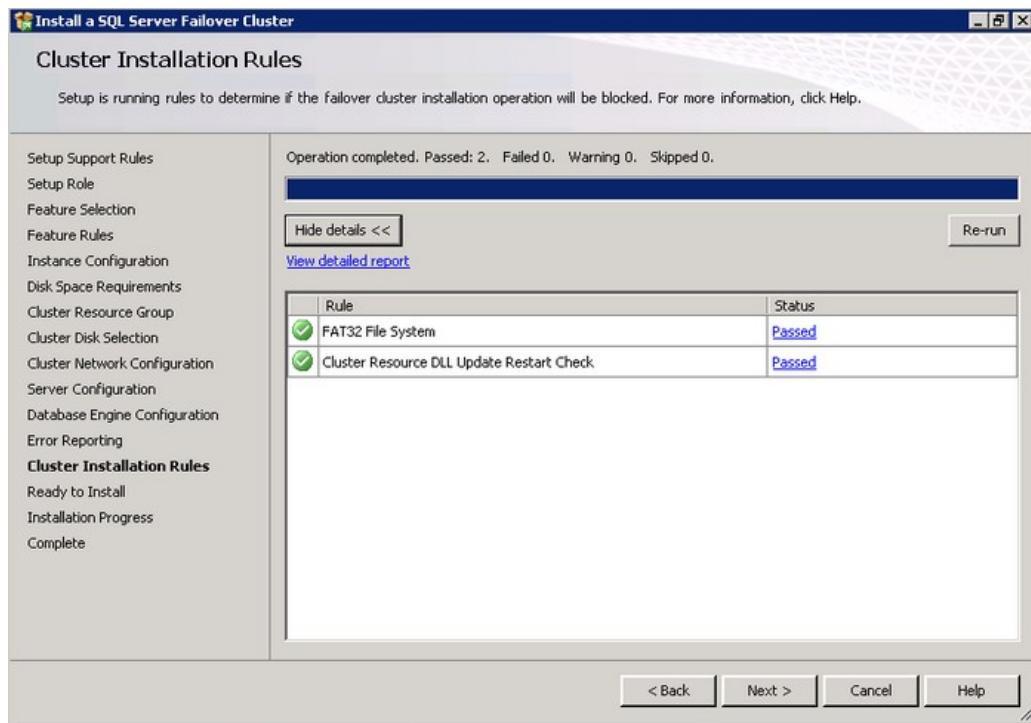
When you choose a local drive versus a clustered drive for the tempdb database, you will get prompted to make sure that all of the nodes in the cluster contain the same directory structure and that the SQL Server service account has read/write permissions on those folders. To make sure that you don't forget, take the time to go thru all of the nodes in your Windows Failover Cluster and create the same folder structure in the same local drive prior to proceeding with the installation. Click **Next**.



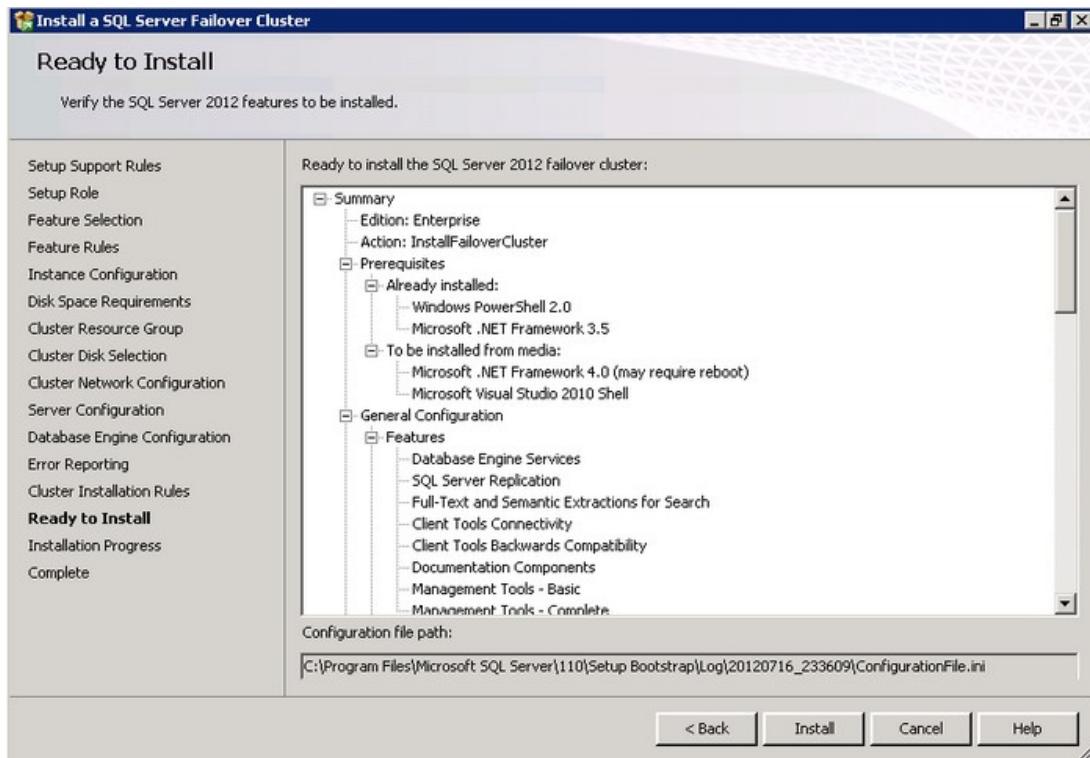
18. In the **Error and Usage Reporting** dialog box, click **Next**.



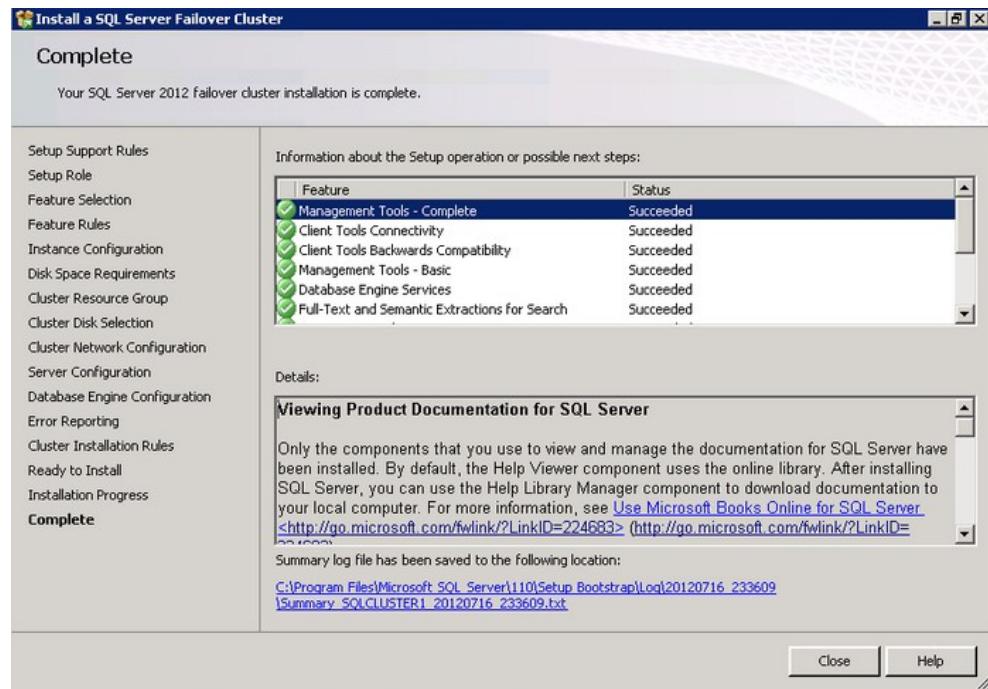
19. In the **Cluster Installation Rules** dialog box, verify that all checks are successful. Click **Next**.



20. In the **Ready to Install** dialog box, verify that all configurations are correct. Click **Next**.



21. In the **Complete** dialog box, click **Close**. This concludes the installation of a SQL Server 2012 Multi-Subnet Failover Cluster.



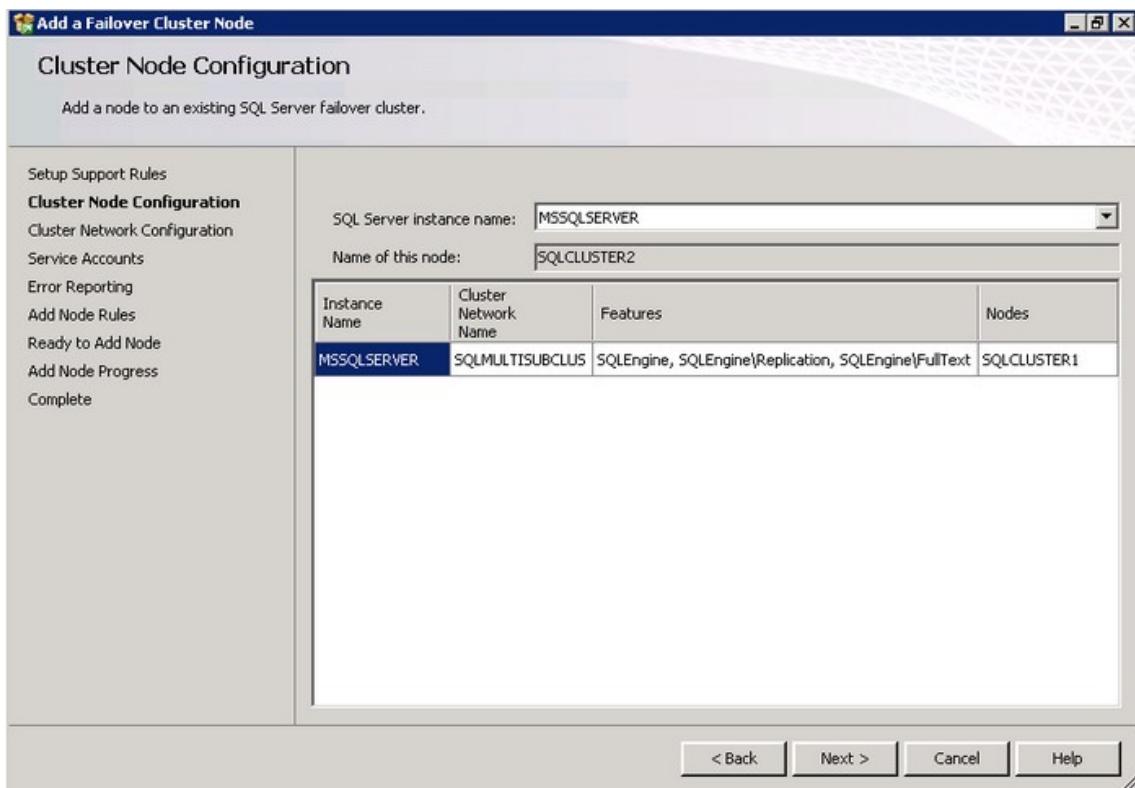
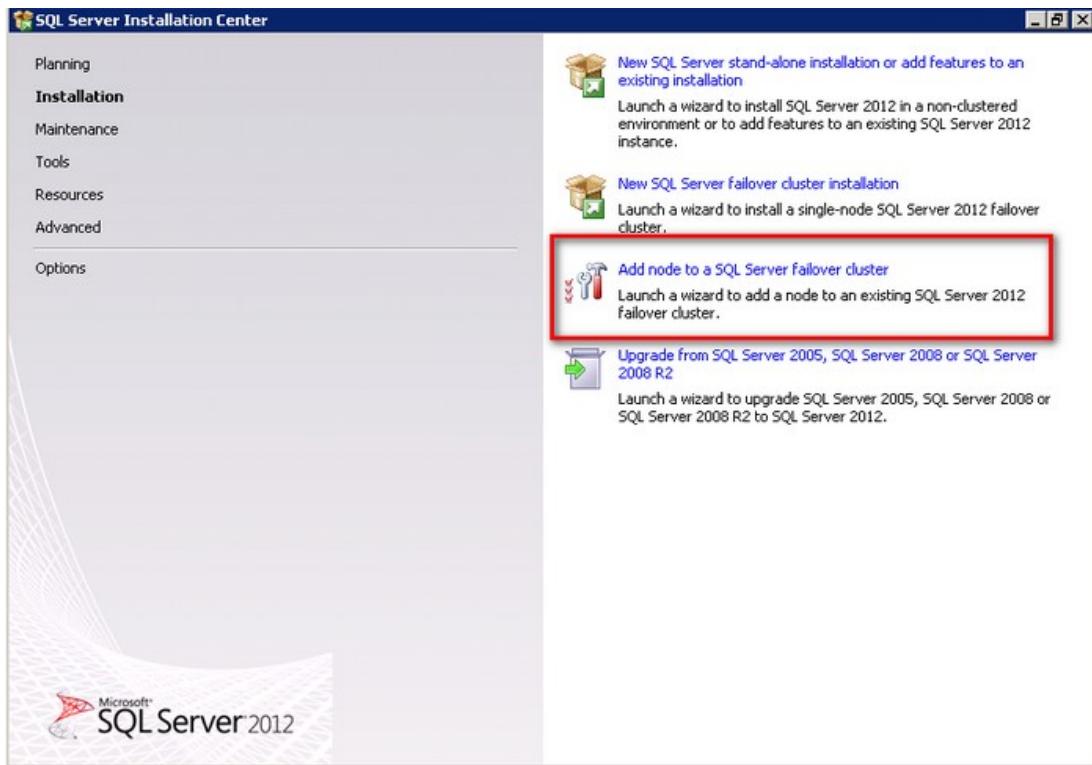
At the completion of a successful installation and configuration of the node we have to add the second node to the SQL Server 2012 multi-subnet cluster.

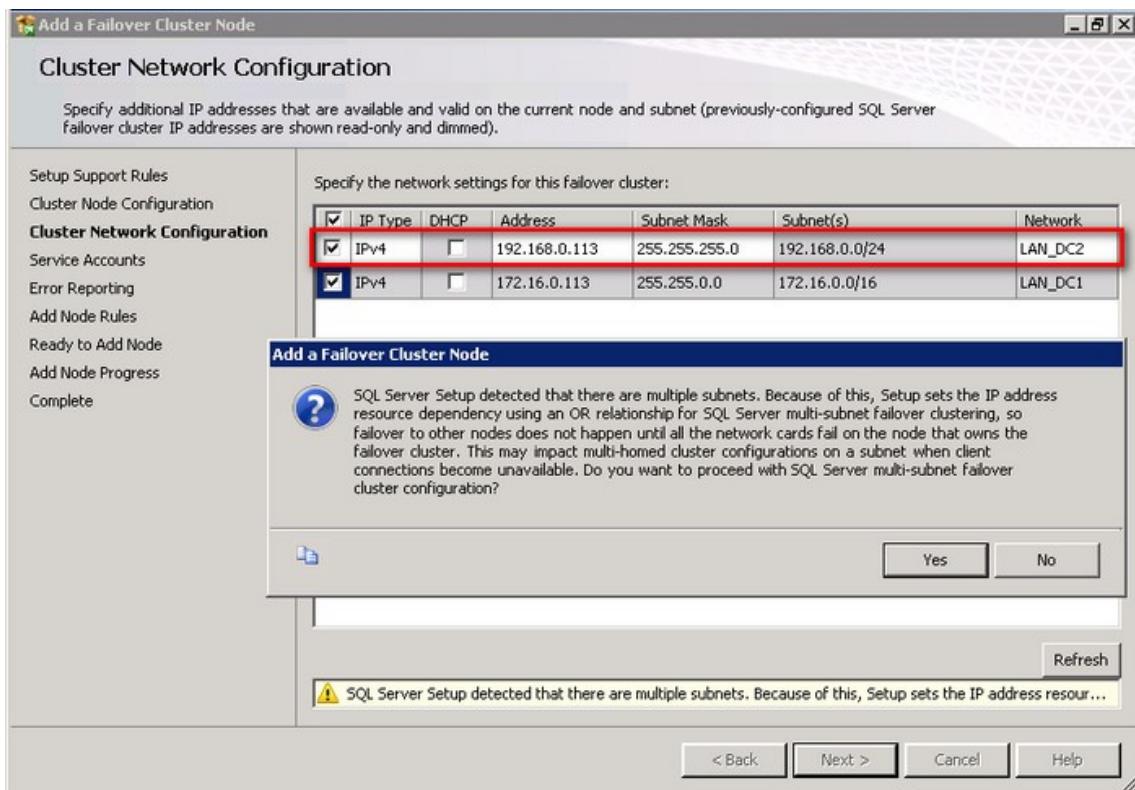
Adding a Node on a SQL Server 2012 Multi-Subnet Cluster

Adding a node in a SQL Server 2012 multi-subnet cluster is no different than performing the same task in a single-subnet cluster

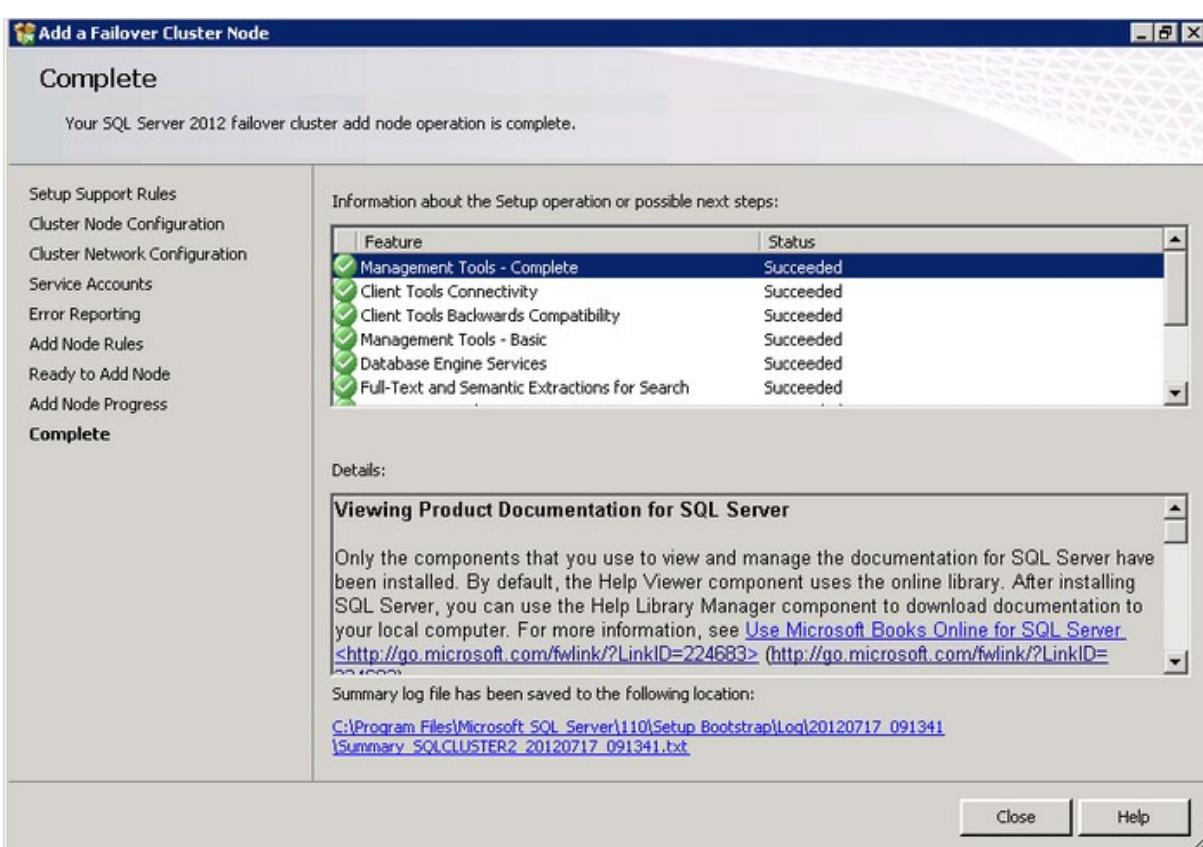
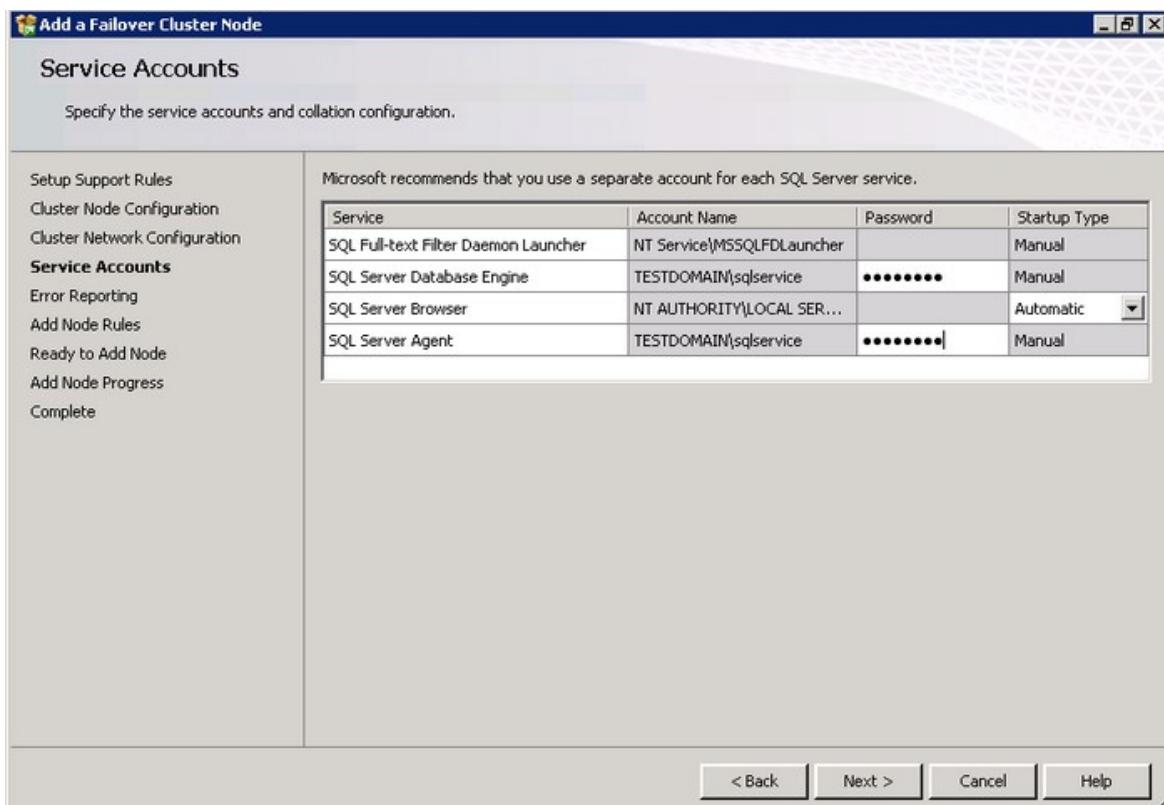
To add a node on a SQL Server 2012 multi-subnet cluster:

1. Run **setup.exe** from the installation media to launch SQL Server Installation Center.
2. Click on the **Installation** link on the left-hand side. Click the **Add node to a SQL Server failover cluster** link. This will run the SQL Server 2012 Setup wizard.
3. In the **Setup Support Rules** dialog box, validate that the checks return successful results and click **OK**.
4. In the **Cluster Node Configuration** dialog box, validate that the information for the existing SQL Server 2012 cluster is correct. Click **Next**.





5. In the **Cluster Network Configuration** dialog box, enter the virtual IP address and subnet mask that your SQL Server 2012 cluster will use in the subnet that the second node is in. From the [previous tip](#), this would be **192.168.0.13**. Similar to the previous tip, notice that the setup process has detected the existence of two network subnets - **LAN_DC1** and **LAN_DC2**. Since we've already configured the virtual IP address for **LAN_DC1** in the first subnet, the section has been disabled. You'll also notice the message box that gives you a brief explanation of how the OR logic dependency works similar to how we've defined it in another [previous tip](#). Click the **Yes** button in the message box. Click **Next**.
6. In the **Service Accounts** dialog box, verify that the information is the same as what you have used to configure the first node. Click **Next**.
7. In the **Error and Usage Reporting** dialog box, click **Next**.
8. In the **Add Node Rules** dialog box, verify that all checks are successful and click **Next**.
9. In the **Ready to Add Node** dialog box, verify that all configurations are correct and click **Install**.
10. In the **Complete** dialog box, click **Close**. This concludes adding a node to a SQL Server 2012 Multi-Subnet Cluster.



You can validate your cluster installation by expanding the **Services and Applications** node in **Failover Cluster Manager** and check the cluster name of your SQL Server instance. You can now see an option to move the service to another node, in this case, the node you've just added in your failover cluster.

Pros of SQL Server Clustering

- Reduces downtime to a bare minimum.
- Permits an automatic response to a failed server or software. No human intervention is required.
- It allows you to perform upgrades without forcing users off the system for extended periods of time.
- It allows you to reduce downtime due to routine server, network, or database maintenance.
- Clustering doesn't require any servers to be renamed. So when failover occurs, it is relatively transparent to end-users.
- Failing back is quick, and can be done whenever the primary server is fixed and put back on-line.
- In some cases, clustering can be used to increase the scalability of an application. For example, if a current cluster is getting too busy, another server could be added to the cluster to expand the resources and help boost the performance of the application.

Cons of Clustering

- More expensive than other failover alternatives, such as log shipping or stand-by servers.
- Requires more set up time than other alternatives.
- Requires more on-going maintenance than other alternatives.
- Requires more experienced DBAs and network administrators.

Best Practices on Clustering

- Detailed planning is critical to the success of every SQL Server cluster installation. Fully plan the install before performing the actual install.
- An expensive cluster is of little value if the supporting infrastructure is not also fault tolerant. For example, don't forget power redundancy, network redundancy, etc.
- Run only a single instance of SQL Server per node. Whether you have two or eight nodes in your cluster, leave one node as a failover node.
- Cluster nodes must not be domain controllers, and all nodes must belong in the same domain and should have access to two or more domain controllers.

- All cluster hardware must be on the Microsoft Windows Clustering Hardware Compatibility List, and certified to work together as part of a cluster.
- Since clustering is not designed to protect data (only SQL Server instances), the shared storage device used by the cluster must incorporate fault tolerant technology. Consider log shipping or mirroring to further protect your production databases.
- When initially installing Windows and SQL Server Clustering, be sure that all drivers and software are up-to-date, including the latest service packs or hot fixes.
- Each node of a cluster should have identical hardware, drivers, software, and configuration settings.
- Fiber channel shared arrays are preferred over SCSI, and Fiber channel has to be used if you include more than two nodes in your cluster.
- The Quorum drive must be on its own fault-tolerant, dedicated, logical drive.
- Once the cluster has been installed, test it thoroughly for every possible failure scenario.
- Do not run antivirus or antispyware on a SQL Server cluster.
- If you need to reconfigure any Windows or SQL Server clustering configuration options, such as IP addresses or virtual names, you will need to uninstall clustering and then reinstall it.
- Monitor active production clusters on a daily basis, looking for any potential problems. Periodically test failover on production servers to ensure all is working well.
- Once you have a stable SQL Server Cluster running, be very leery about making any changes to it, whatsoever.

Case study 1: Changing the Network Name of a SQL Server Failover Cluster

Both SQL Server 2005 and SQL Server 2008 failover clusters support changing the SQL Server failover cluster network name after installation. This was not possible with SQL Server 2000 failover clusters. A SQL Server failover cluster name can contain the network name and an instance name. For a default instance, there is a network name for the failover cluster instance, but no instance name. For example, if the SQL Server failover cluster was called VS1\instance1, SQL Server 2008 supports changes to the VS1 part of the name. SQL Server 2008 does not support changing the instance name (instance1) without a reinstallation of SQL Server.

1. Start Cluster Administrator.
2. Select the resource group that has the SQL Server resources.
3. Take the SQL Server service offline.
4. Right-click the SQL Server Network Name resource, and select Properties; alternatively, double-click the resource.

5. Select the Parameters tab of the resource's properties page
6. Enter the new name of the failover clustering resource and click Apply.
7. Select the General tab. Change the name of the resource to include the new name you configured in step 6. Click OK.
8. Bring the SQL Server resources online.
9. Ping the new name of the SQL Server failover clustering instance. If the name cannot be resolved, you will have to flush your DNS cache by issuing these three commands in succession: ipconfig /flushdnsipconfig /registerdnsand nbtstat –RR.
10. Start SQL Server Management Studio, and connect with the new instance name. If this succeeds, the change has been done successfully.
11. For a final verification, you can also run a SELECT @@SERVERNAME query, which should reflect the name change.

Note:

You can only rename the SQL Server virtual Name and not the instance name and this operation involves some down time.

Case Study 2: Changing the Service Accounts for a SQL Server Failover Cluster

The following steps are similar to what you would need to do on a stand-alone instance of SQL Server to change the service accounts. You should use SQL Server Configuration Manager to change the service accounts. Bear in mind the following points when you manage the service accounts for a failover cluster:

- Do not change passwords for any of the SQL Server service accounts when a failover cluster node is down or offline. If you must do this, you must reset the password again by using SQL Server Configuration Manager when all nodes are back online.
- If the service account for SQL Server does not have administrative rights on the Windows Server cluster, you cannot delete the administrative shares on any nodes of the cluster. The administrative shares must be available on a cluster for SQL Server to function.

Case Study 3: Changing the IP Address of a SQL Server Failover Cluster

1. Open the Failover Cluster Manager snap-in.
2. Expand the **Services and applications** node, in the left pane and click on the FCI.
3. On the right pane, under the **Server Name** category, right-click the SQL Server Instance, and select **Properties** option to open the **Properties** dialog box.
4. On the **General** tab, change the IP address resource.
5. Click **OK** to close the dialog box.
6. In the right-hand pane, right-click the SQL IP Address1(failover cluster instance name) and select **Take Offline**. You will see the SQL IP Address1(failover cluster instance name), SQL Network Name(failover cluster instance name), and SQL Server status change from Online to Offline Pending, and then to Offline.

7. In the right-hand pane, right-click SQL Server, and then select **Bring Online**. You will see the SQL IP Address1(failover cluster instance name), SQL Network Name(failover cluster instance name), and SQL Server status change from Offline to Online Pending, and then to Online.
8. Close the Failover Cluster Manager snap-in.

Case study- 4: How to add storage to Clustered Shared Volumes in Windows Server 2012

In Windows Server 2012 Cluster Shared Volumes (CSV) has been more tightly integrated into the Failover Clustering feature. The process for a cluster Physical Disk Resource (PDR) to be enabled for CSV has been simplified and streamlined. In this blog, I will show you the new experience of adding storage from the **Available Storage** pool of your cluster to Clustered Shared Volumes. The**Available Storage** pool contains disks that have been added to your cluster but not assigned to a specific use in your cluster.

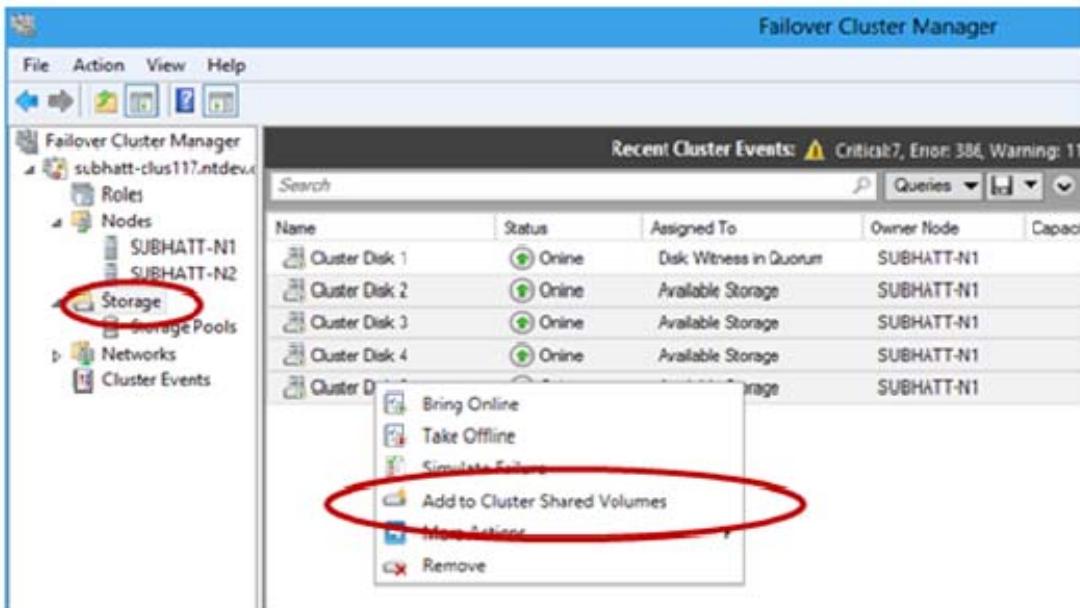
Failover Cluster Manager

To add storage to Clustered Shared Volumes follow these steps:

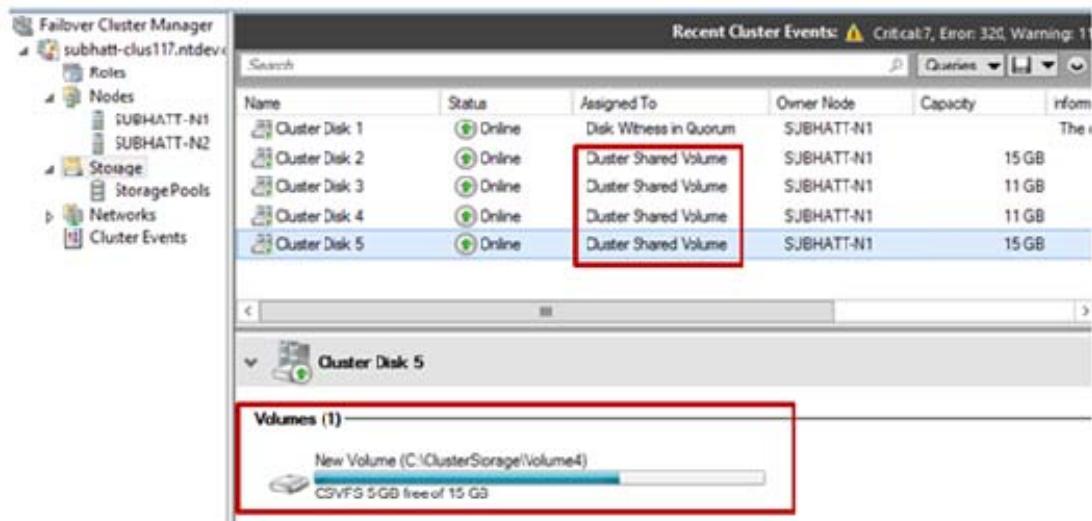
- 1) Launch the **Failover Cluster Manager** (CluAdmin.msc)
- 2) Select the **Storage** node
- 3) Select the Disks that you want to add to Clustered Shared Volumes.

Note: A great new Failover Cluster Manager feature in Windows Server 2012 is support for multi-select and the ability to enable CSV across a number of disks all at once!

- 4) Right click on your selection and choose the **Add to Cluster Shared Volumes** option.



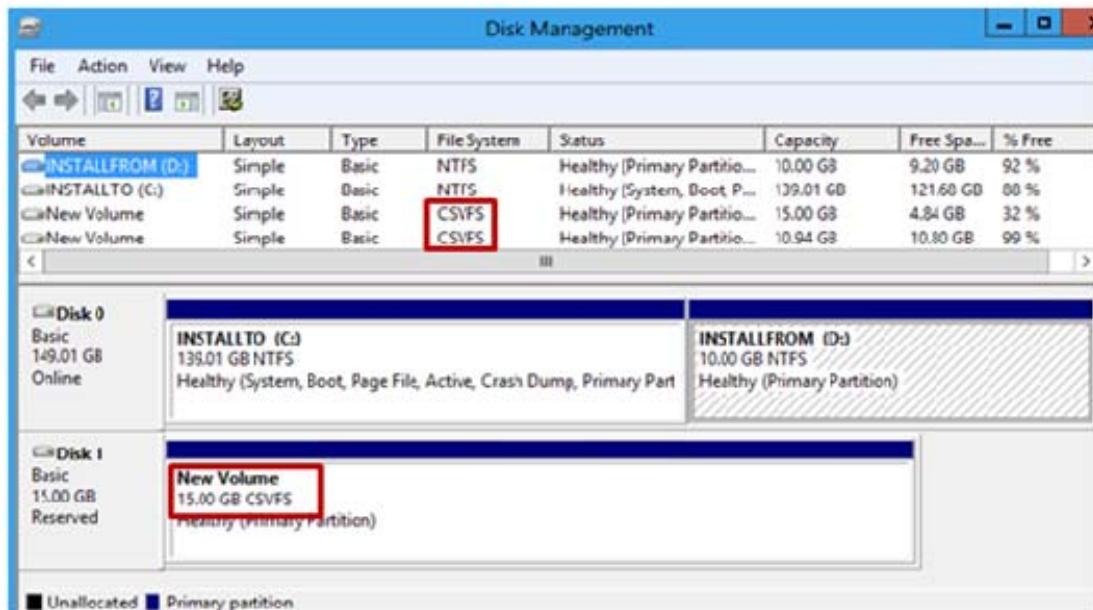
- 5) Your disks are now added to Clustered Shared Volumes! Yes, it is that easy in Windows Server 2012!



CSV provides a single consistent file name space. Files have the same name and path when viewed from any node in the cluster. CSV volumes are exposed as directories and subdirectories under the **"ClusterStorage"** root directory: **C:\ClusterStorage\VolumeX\<root>**

Name	Date modified	Type	Size
Volume1	4/2/2012 5:13 PM	File folder	15,725,564 ...
Volume2	4/2/2012 5:13 PM	File folder	11,468,800 ...
Volume3	4/2/2012 5:13 PM	File folder	11,468,800 ...
Volume4	4/3/2012 10:55 AM	File folder	15,725,564 ...

CSV enabled volumes now appear as "CSVFS". CSVFS is the NTFS file system under the covers and volumes are still formatted with the NTFS file system. However, this change enables applications to be aware that they are running on CSV and allows them to ensure compatibility.

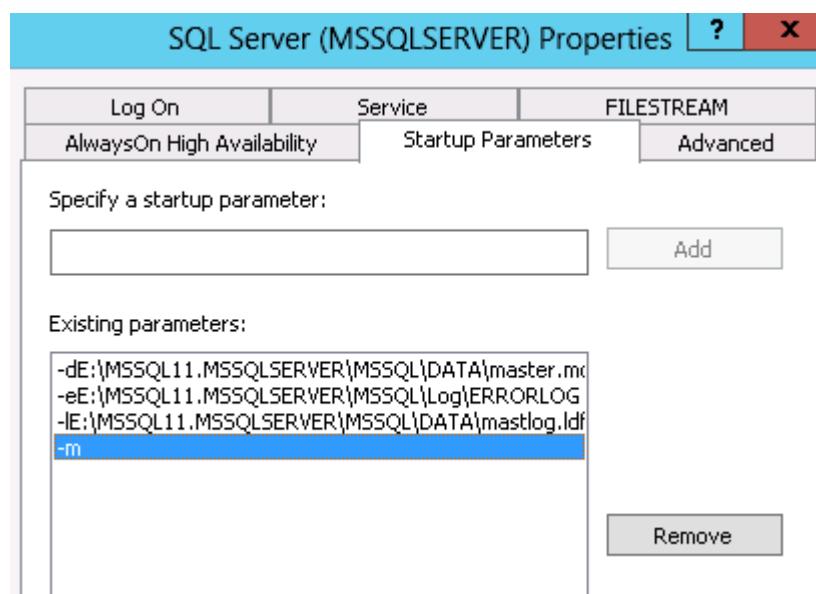


Case study 5: How to start SQL server clustered instance in single user mode

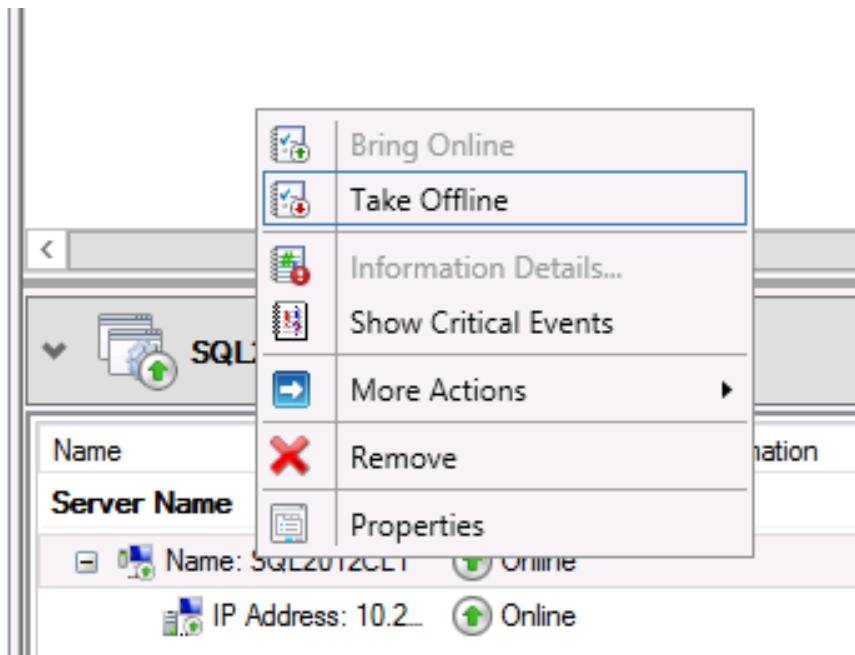
Remote Desktop into the node of your cluster that currently owns the SQL Server service. You can do this either by connecting the Cluster Network Name, or by checking who the Owner Node is in Failover Cluster Manager.

Roles (1)				
Name	Status	Type	Owner Node	Priority
SQL2012CL1	Running	Other	W2012-cl1	Medium

If you had a -M in your startup parameters, remove it now.



Take the resource name for the cluster offline—note this is not the whole clustered service or role—just the resource. You will still need the disks and IP address that the cluster needs to run. **Note—you also want to make sure the SQL Server service and SQL Agent service are stopped, if your dependencies are correct this should happen anyway, but confirms.**



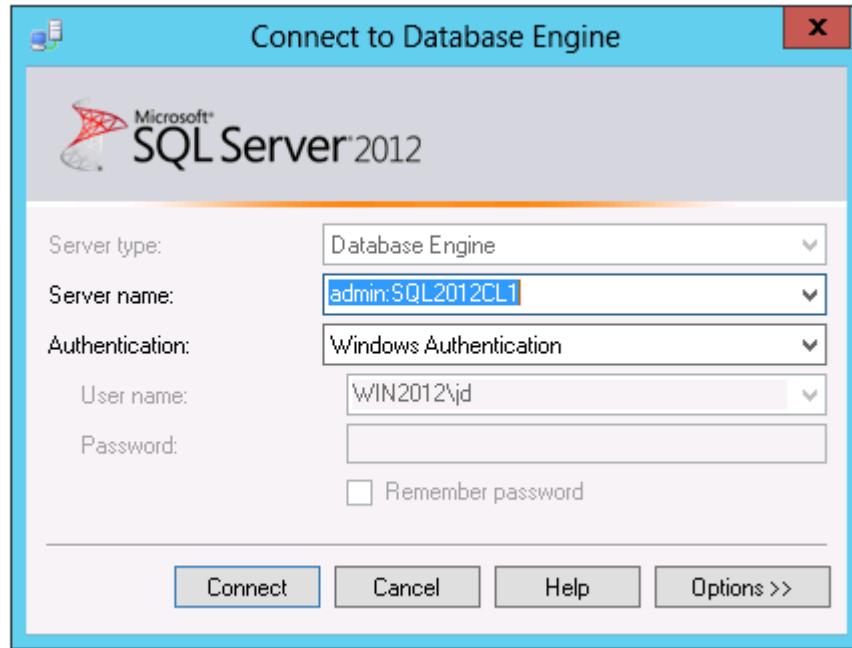
Launch a command prompt, and issue the following command: **net start MSSQLServer /m**

```
Administrator: Windows PowerShell
PS C:\Windows\system32> net start mssqlserver /m
The SQL Server (MSSQLSERVER) service is starting.
The SQL Server (MSSQLSERVER) service was started successfully.

PS C:\Windows\system32>
```

A screenshot of a Windows Command Prompt window titled 'Administrator: Windows PowerShell'. The command 'net start mssqlserver /m' is entered and executed. The output shows the service starting and then successfully starting. The prompt then returns to the command line.

Launch your DAC connection and do whatever work you need to do.



When you are completed stop the SQL Server service from your command window.

```
PS C:\Windows\system32> net stop mssqlserver
The SQL Server (MSSQLSERVER) service is stopping.
The SQL Server (MSSQLSERVER) service was stopped successfully.

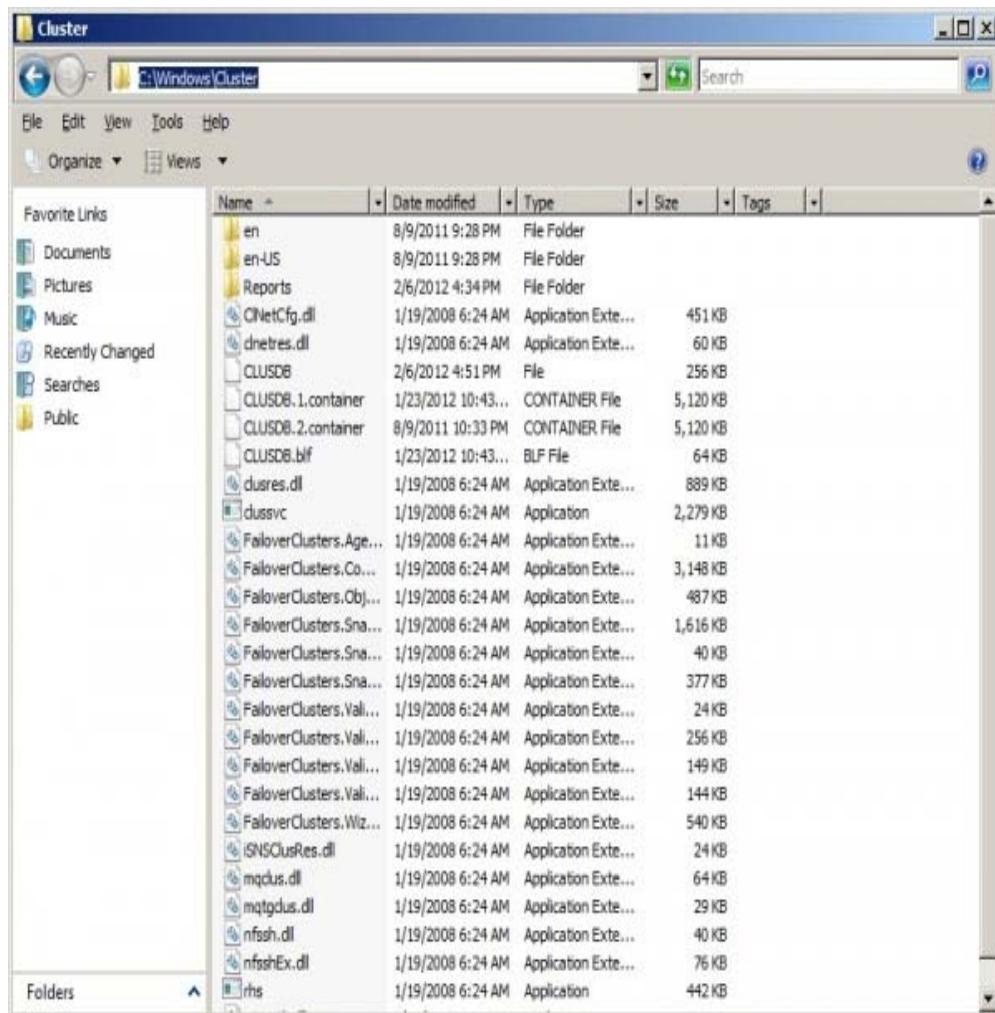
PS C:\Windows\system32>
```

Go back to failover cluster manager and bring your Cluster Resource back online. Then restart SQL Server and SQL Agent. Green is good—if you have any services that don't restart check the cluster and application logs in Event Viewer.

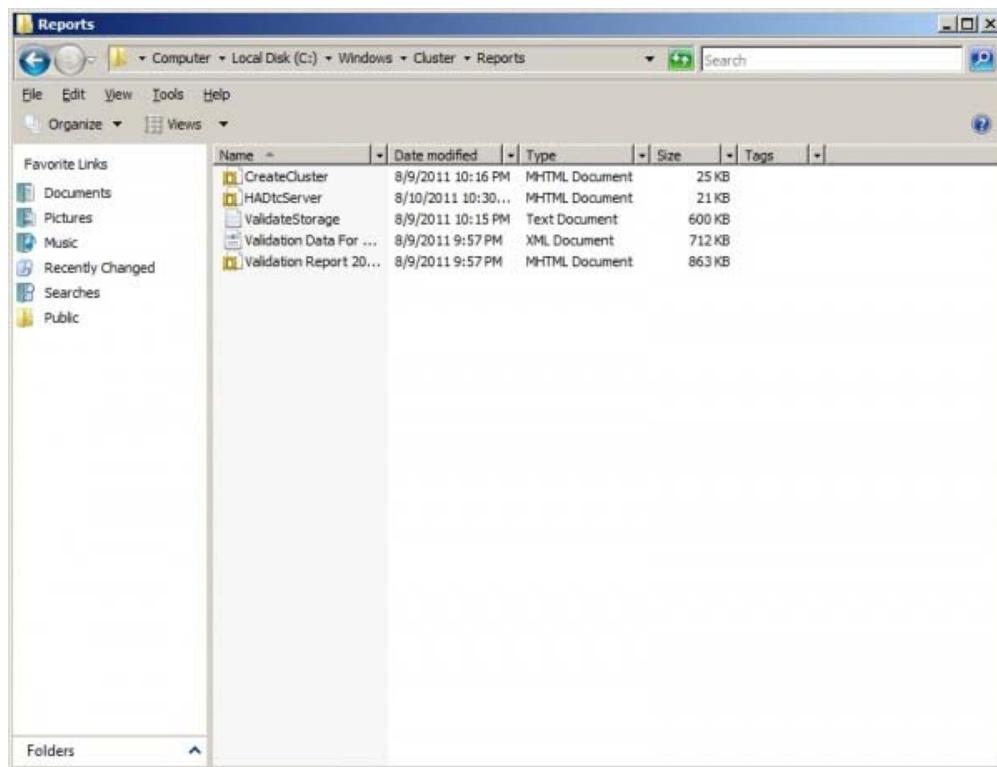


Case study 6: How to create cluster logs in windows Server 2008/R2?

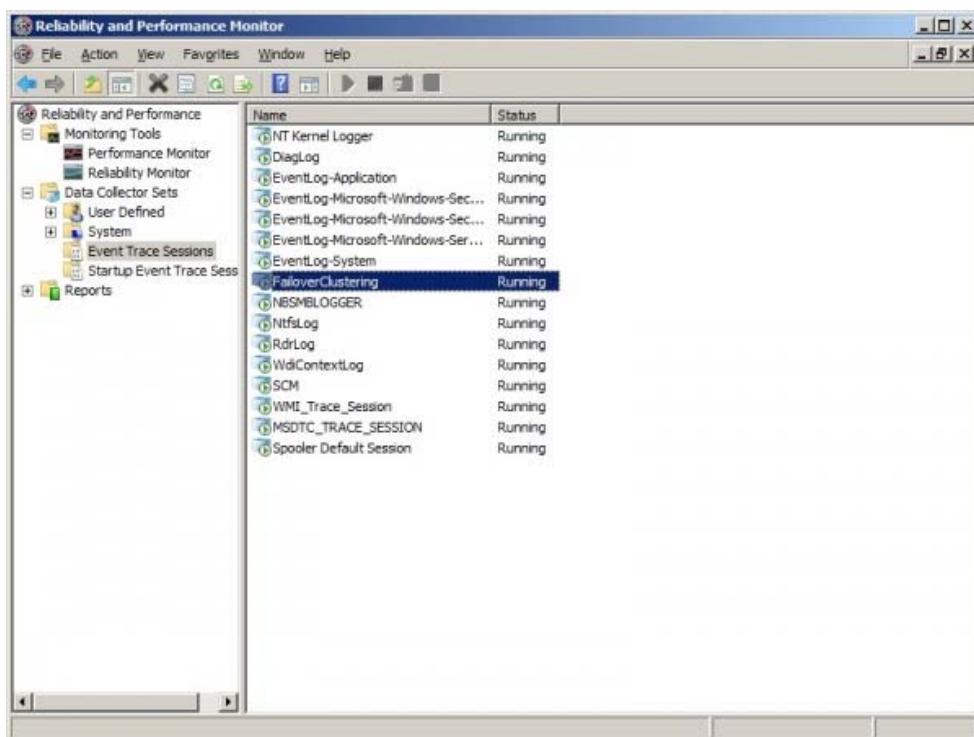
As most of us know on Windows Server 2003 Cluster, we used to have “**cluster.log**” file on each node participating in cluster, which contains debug information. FYI, One can locate these files in “**C:\windows\ cluster**” Folder. But how about cluster log files in Windows Server 2008/2008R2?? Uhuhh...It's not something which you can review directly by navigating to systemroot folder. Below is the screenshot of that folder in my cluster.



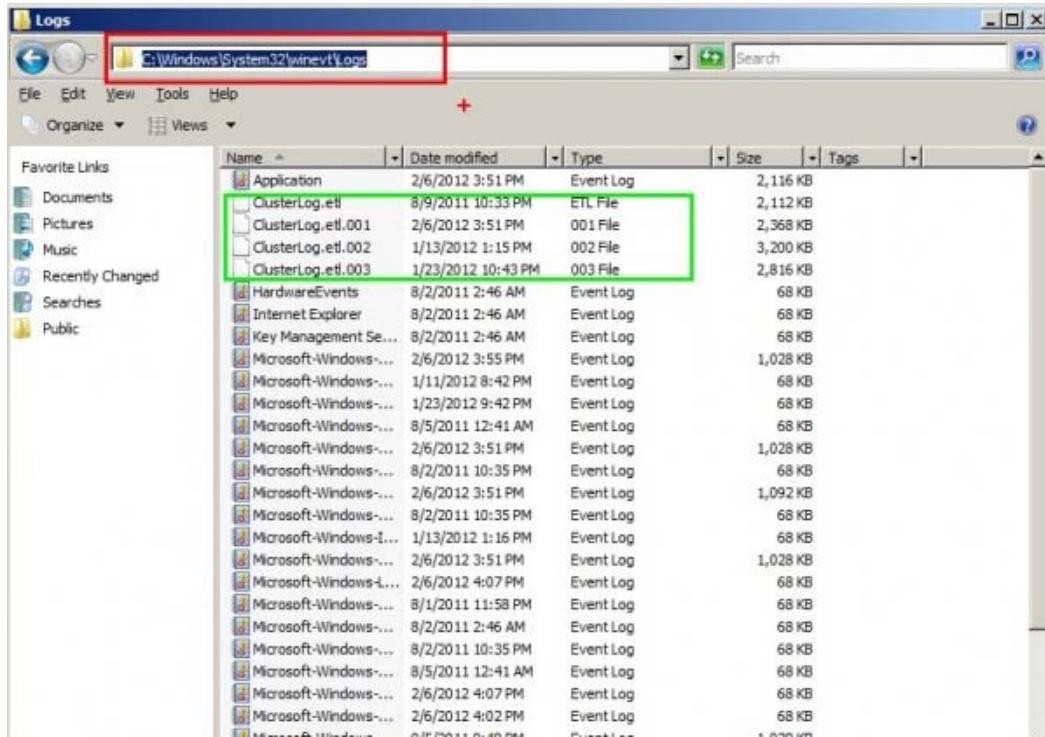
You can see a folder called “**REPORTS**” in the above screenshot where all the cluster Validation Reports will be stored by default attaching below Screenshot Just to prove, that cluster.log file can't be located in the “reports” folder as well



Starting Windows Server 2008, **cluster logs are managed by something called as “Windows Event Tracing”**. Just an FYI, If you are interested, You can pull all the current running traces by opening “perfmon” and navigating to Data Collector Sets. (Shown Below in the Screenshot)



So, as any other logs, cluster logs are stored in “C:\Windows\System32\winevt\Logs” folder with “.etl” extension as you can see below.



Well, so **How to read those .ETL files??**

For that, we have to use “cluster.exe” command with “/gen” switch. Basically this will generate a human readable text file in your “Reports” folder.

Syntax: Cluster log /gen

Output:

```
C:\>cluster log /gen
Generating the cluster log(s) ...
The cluster log has been successfully generated on node 'Node1'...
The cluster log could not be created on node 'Node2'...
System error 1722 has occurred <0x0000006ba>.
The RPC server is unavailable.

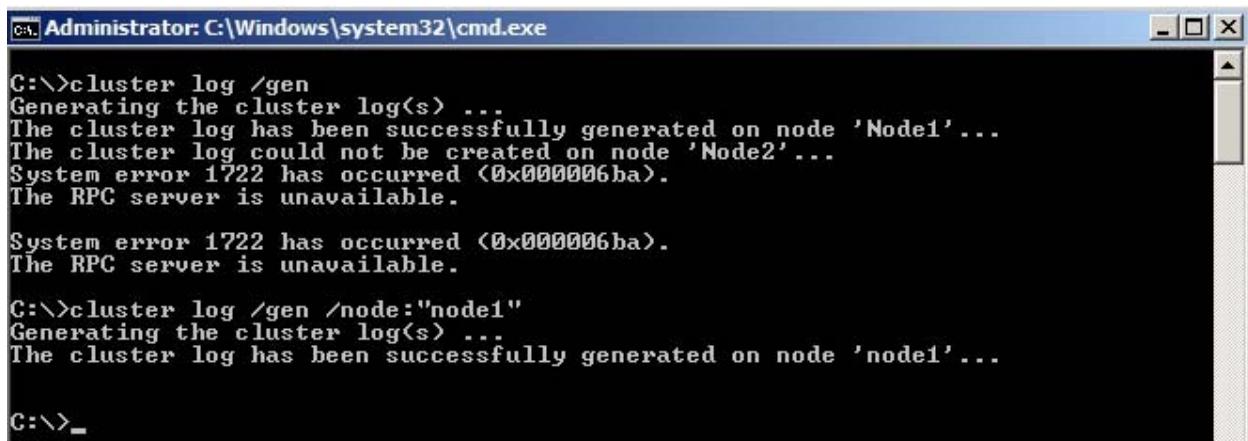
System error 1722 has occurred <0x0000006ba>.
The RPC server is unavailable.

C:\>_
```

As you can see in the above Screenshot, it will communicate with all the nodes in your cluster. In my scenario, Node2 is offline(Powered down). BTW, even though Node2 is down, it will create “Cluster.txt” file in your Reports Folder with related information.

So, **how to generate Logs related to a specific Node?**

You have to use “**/NODE**” switch with your cluster log syntax. Please see below Screenshot.



The screenshot shows a Windows Command Prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The command "cluster log /gen" is run, which generates logs on Node1 but fails on Node2 due to RPC server unavailability. Then, the command "cluster log /gen /node:'node1'" is run, successfully generating logs on Node1.

```
C:\>cluster log /gen
Generating the cluster log(s) ...
The cluster log has been successfully generated on node 'Node1'...
The cluster log could not be created on node 'Node2'...
System error 1722 has occurred (0x0000006ba).
The RPC server is unavailable.

System error 1722 has occurred (0x0000006ba).
The RPC server is unavailable.

C:\>cluster log /gen /node:"node1"
Generating the cluster log(s) ...
The cluster log has been successfully generated on node 'node1'...

C:\>_
```

Case Study 7: Force a WSFC Cluster to Start Without a Quorum

To force a cluster to start without a quorum

1. Open a Failover Cluster Manager and connect to the desired cluster node to force online.
2. In the Actions pane, click Force Cluster Start, and then click Yes – Force my cluster to start.
3. In the left pane, in the Failover Cluster Manager tree, click the cluster name.
4. In the summary pane, confirm that the current Quorum Configuration value is: Warning: Cluster is running in ForceQuorum state.

Chapter 20

AlwaysOn Availability Groups

Overview

The AlwaysOn Availability Groups feature is a high-availability and disaster-recovery solution that provides an enterprise-level alternative to database mirroring. Introduced in SQL Server 2012, AlwaysOn Availability Groups maximizes the availability of a set of user databases for an enterprise. An *availability group* supports a failover environment for a discrete set of user databases, known as *availability databases*, that fail over together. An availability group supports a set of read-write primary databases and one to four sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations.

An availability group fails over at the level of an availability replica. Failovers are not caused by database issues such as a database becoming suspect due to a loss of a data file, deletion of a database, or corruption of a transaction log.

Terminology:

availability group - A container for a set of databases, *availability databases*, that fail over together.

availability database - A database that belongs to an availability group. For each availability database, the availability group maintains a single read-write copy (the *primary database*) and one to four read-only copies (*secondary databases*).

primary database - The read-write copy of an availability database.

secondary database - A read-only copy of an availability database.

availability replica - An instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. Two types of availability replicas exist: a single *primary replica* and one to four *secondary replicas*.

primary replica - The availability replica that makes the primary databases available for read-write connections from clients and, also, sends transaction log records for each primary database to every secondary replica.

secondary replica - An availability replica that maintains a secondary copy of each availability database, and serves as a potential failover targets for the availability group. Optionally, a secondary replica can support read-only access to secondary databases can support creating backups on secondary databases.

availability group listener - A server name to which clients can connect in order to access a database in a primary or secondary replica of an AlwaysOn availability group. Availability group listeners direct incoming connections to the primary replica or to a read-only secondary replica.

Advantages:

AlwaysOn Availability Groups provides a rich set of options that improve database availability and that enable improved resource use. The key components are as follows:

- Supports up to five availability replicas. An *availability replica* is an instantiation of an availability group that is hosted by a specific instance of SQL Server and maintains a local copy of each availability database that belongs to the availability group. Each availability group supports one primary replica and up to four secondary replicas.
- Supports a flexible failover policy for greater control over availability-group failover.
- Supports alternative availability modes, as follows:
 - *Asynchronous-commit mode*. This availability mode is a disaster-recovery solution that works well when the availability replicas are distributed over considerable distances.
 - *Synchronous-commit mode*. This availability mode emphasizes high availability and data protection over performance, at the cost of increased transaction latency. A given availability group can support up to three synchronous-commit availability replicas, including the current primary replica.
- Supports several forms of availability-group failover: automatic failover, planned manual failover (generally referred as simply "manual failover"), and forced manual failover (generally referred as simply "forced failover").
- Supports an availability group listener for each availability group. An *availability group listener* is a server name to which clients can connect in order to access a database in a primary or secondary replica of an AlwaysOn availability group. Availability group listeners direct incoming connections to the primary replica or to a read-only secondary replica. The listener provides fast application failover after an availability group fails over

Disadvantages:

They provide support for one primary replica and up to four secondary replicas, where each replica is located on a separate SQL Server instance running on different Windows failover cluster nodes.

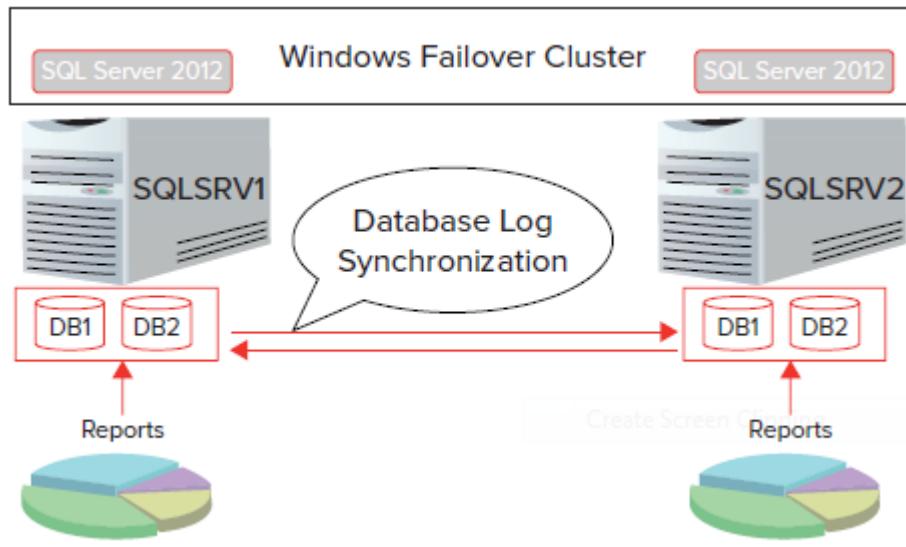
AlwaysOn Availability Groups can contain multiple databases, all of which can be automatically failed over as a unit. This means that AlwaysOn Availability Groups can protect multiple related databases and fail them over simultaneously. For example, if your application uses ASP.NET forms integration for authentication, you can fail over the default aspnetdb database in addition to your application's primary database. That way, the user authentication information can go along with your production data.

AlwaysOn is that you don't have to choose between asynchronous mode and synchronous mode, like you did with database mirroring. AlwaysOn can have both synchronous and asynchronous replicas at the same time. Synchronous connections are typically used in high-availability scenarios, where there's automatic failover. Asynchronous connections are typically used in disaster recovery scenarios, where there's geographical distance between the different servers.

Architecture:

An Availability Group is a set of up to five configured SQL servers into an Availability Group Failover for a discrete set of user databases, known as *availability databases*. Each set of availability database(s) is hosted by an *availability replica*. Every availability replica is assigned a role. Each

availability replica also has associated with it an availability mode. Figure shows an Availability Group architecture in detail and the following sections elaborate on these various components



Availability Group Replicas and Roles

As stated previously, an availability database is hosted by an availability replica. Two types of availability replicas exist:

1. A single *primary replica* makes the primary databases available for read-write connections from clients and also sends transaction log records for each primary database to every secondary replica.
2. One to four *secondary replicas* maintain a set of secondary databases. Every secondary replica applies transaction log records to its own set of secondary databases and serves as a potential failover target for the Availability Group.

Optionally, you can configure one or more secondary replicas to support read-only access to secondary databases, and you can configure any secondary replica to permit backups on secondary databases.

Every availability replica is assigned an initial role — either the primary role or the secondary role, which is inherited by the availability databases of that replica. The role of a given replica determines whether it hosts read-write databases or read-only databases. The primary replica is assigned the primary role and hosts read-write databases, which are known as primary databases. At least one secondary replica is assigned the secondary role. A secondary replica hosts read-only databases, known as secondary databases.

For the replicas to participate in Availability Groups they must be deployed as a Windows Failover Cluster (WFC); therefore, to implement an Availability Group, you must first enable Windows Failover Cluster in each server participating in the Availability Group. Inside of a WFC, an Availability Group is a cluster resource with its own network name and cluster IP address used for application clients to connect to the Availability Group. However, the major difference from a SQL Server cluster installation is that an Availability Group does not require a shared disk; that is, in this configuration, each server does not share its disks with the other server.

Availability Modes

Each availability replica contains a property that determines its availability mode in relationship to the transaction record data movement that it receives, which determines how current its data is in relationship to the primary replica that is moving the data to all the other replicas. Two modes of availability are supported:

1. Asynchronous-commit mode: The primary replica commits transactions without waiting for acknowledgment that an asynchronous-commit secondary replica has hardened the transaction log. Asynchronous-commit mode minimizes transaction latency on the secondary replica databases but enables them to lag behind the primary replica databases, increasing the probability in case of a failure or possible data loss.
2. Synchronous-commit mode: The primary replica waits for a synchronous-commit from the secondary replicas to acknowledge that it has finished hardening the log. Synchronous-commit mode ensures that when a given secondary replica database is synchronized with the primary replica database, committed transactions are fully protected. This protection comes at the cost of increased request latency and increases the user transaction times because the latency is included in the overall database response time.

Types of Failover Supported

During failover, the primary and secondary replicas are interchangeable where one of the secondary replicas becomes the primary, and the former primary becomes the secondary replica. The new primary brings its Availability Group databases online and makes its databases available for read/write operations. The former primary that is now the secondary replica then begins to receive transactional data from the new primary replicate after the failover. If the failover is caused by a failure to the former primary such that it became unavailable for a time, when the former primary returns online it automatically joins in its Availability Group and takes a secondary replica role.

There are three modes of failover: automatic, manual, and forced (with possible data loss). Support is based on the availability mode configured:

AVAILABILITY MODE	FAILOVER MODE SETTING	SECONDARY REPLICA FAILOVER SUPPORTED
Synchronous	Manual	Manual (without data loss)
Synchronous	Automatic	Automatic/Manual (without data loss)
Asynchronous	Manual	Manual (Possible data loss)

Manual failover (without data loss): A manual failover occurs after a database administrator issues a manual-failover command and causes a synchronized secondary replica to transition to be the primary replica role (with guaranteed data protection) and the primary replica to transition as the secondary replica role. The database administrator can specify which secondary replica to failover to. A manual failover requires that both the primary replica and the target secondary replica run under synchronous-commit mode, and the secondary replica must already be synchronized.

Automatic failover (without data loss): An automatic failover occurs in response to a failure that causes a synchronized secondary replica to transition to the primary role (with guaranteed data protection). When the former primary replica becomes available, it transitions to the secondary role.

Automatic failover requires that both the primary replica and the target secondary replica run under synchronous-commit mode with the failover mode set to Automatic. In addition, the secondary replica must already be synchronized.

Manual (possible data loss): Under asynchronous-commit mode, the only form of failover is manual forced failover (with possible data loss). A forced failover should only be used for disaster recovery as there is a good risk of data loss. Manual is the only form of failover possible when the target secondary replica is not synchronized with the primary replica, even for synchronous-commit mode.

Allowing Read-Only Access to Secondary Replicas

The main purpose of the Availability Group is to deliver high availability and disaster recovery, but in addition, a secondary replica can be configured for read-only database access. In such a scenario, read-only workloads are offloaded on the secondary replica databases, optimizing the primary replica to support read/write mission critical workloads. For example, if you want to run reports, rather than running them on the primary replica and overburdening it, you can select one of the secondary replicas to run the reports. You can also execute database backups from the secondary replica; full database, file, and filegroup backups are supported but differential backup is not supported. Transaction log backup is supported in the secondary replica. Consider the following capabilities of read-only secondary replicas:

1. The secondary replica's data is maintained to near real time with the primary replica. Real time depends on the network latency between the primary and secondary, the redo operation, locking on the secondary databases, and activities running on the secondary server.
2. Read-only on the secondary replica applies to all the Availability Group databases.
3. To greatly reduce locking in the secondary read-only replica, the secondary replica databases are configured by default in snapshot isolation to avoid read activities blocking the redo log updating the data. Snapshot isolation adds a 14-byte row identifier to maintain row version and maintains last committed row(s) in tempdb as the redo operation updates the replica databases.
4. For optimizing the read-only workload, any index required by the secondary replica or database statistics needs to be created in the primary replica database and transferred to all replicas by the transactional log data movement because the secondary is read-only. However, to support read-only workload, SQL Server 2012 does have the capability to create temporary database statistics that are stored in tempdb. These temporary statistics will be lost when the SQL Server instance is restarted or when the secondary replica is promoted to the primary during a failover.
5. Similar to data mirroring, Availability Groups support automatic page repair. Each availability replica tries to automatically recover from corrupted pages on a local database by resolving certain types of errors that prevent reading a data page. If a secondary replica cannot read a page, the replica requests a fresh copy of the page from the primary replica. If the primary replica cannot read a page, the replica broadcasts a request for a fresh copy to all the secondary replicas and gets the page from the first to respond. If this request succeeds, the unreadable page is replaced by the copy, which usually resolves the error.
6. A client application can connect to a read-only replica. You can connect to a secondary replica that supports read-only access in one of two ways: either directly by specifying the SQL Server instance name in the connection string, or by using the Availability Group Listener name and leveraging read-only routing to reconnect to the next available secondary read-only replica.

Pre-Requisites for configuring the AlwaysOn Availability groups:

Hardware:

AlwaysOn Availability Groups require at least two server systems and can work with as many as four systems. They can be either physical server systems or virtual machines (VMs). In addition, AlwaysOn Availability Groups require Windows failover clustering, which means you have to use the Enterprise Edition (or higher) of Windows Server 2008 R2 or Windows Server 2008. (The Standard Editions of Server 2008 R2 and Server 2008 don't support failover clustering.)

The AlwaysOn Availability Groups' requirement for clustering means additional complexity compared with database mirroring. Fortunately, Windows failover clustering became much easier to set up beginning with Server 2008. It's important to note that although AlwaysOn Availability Groups require a Windows failover cluster, they don't require installing SQL Server as a clustered application.

Windows/Server Instance:

To use availability groups, you need at least two SQL Server instances running on different Windows failover cluster nodes. The general steps to set up the SQL Server instances on Windows failover cluster nodes are as follows:

1. Ensure that the system is not a domain controller.
2. Ensure that each computer is a node in a Windows Server Failover Clustering (WSFC) cluster.
3. Use Server Manager to install the Failover Clustering feature on all nodes.
4. Use the Failover Cluster Management tool to create a new Windows failover cluster.
5. Install a new standalone instance of SQL Server on each cluster node.
6. Each server instance must be running the Enterprise Edition of SQL Server 2012.
7. All the server instances that host availability replicas for an availability group must use the same SQL Server collation
8. Enable the AlwaysOn Availability Groups feature on each server instance that will host an availability replica for any availability group. On a given computer, you can enable as many server instances for AlwaysOn Availability Groups as your SQL Server installation supports.

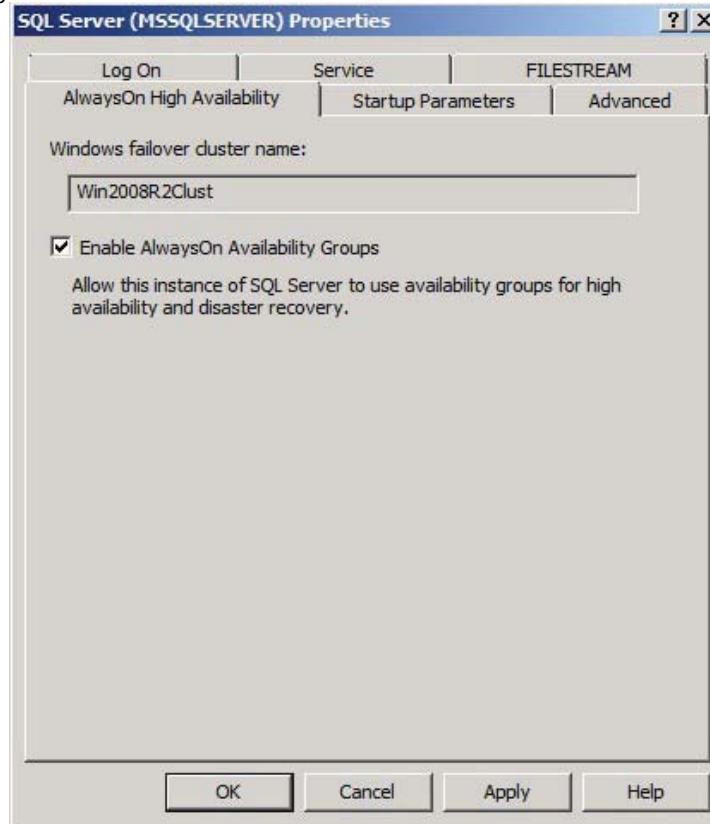
Database:

1. Availability groups must be created with user databases. Systems databases can't be used.
2. Databases must be read-write. Read-only databases aren't supported.
3. Databases must be multiuser databases.
4. Databases can't use the AUTO_CLOSE feature.
5. Databases must use the full recovery model, and there must be a full backup of them.
6. A given database can only be in a single availability group, and that database can't be configured to use database mirroring.

Availability Group Configuration

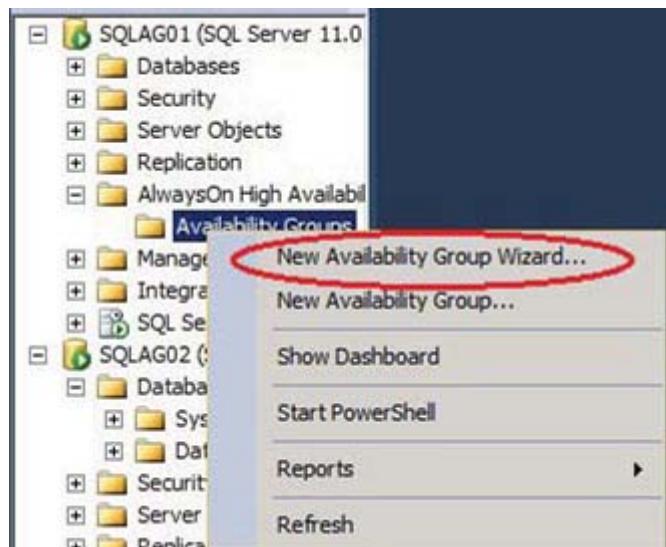
As stated previously, a prerequisite to deploy an Availability Group is a Windows Failover Cluster. The following steps begin with creating a Windows 2008 R2 Failover Cluster and continue through the whole process of configuring a new Availability Group:

1. Start with installing Windows 2008 R2 Failover Cluster on each server that will participate in the Availability Group.
2. Then install SQL Server 2012 as a standalone instance on each server that will participate in the Availability Group.
3. Next on each SQL Server that will be participating in the Availability Group, open SQL Server Configuration Manager, choose SQL Server Services, and then select the SQL Server properties. Under AlwaysOn High Availability, check the Enable AlwaysOn Availability Groups check box, as shown in Figure:

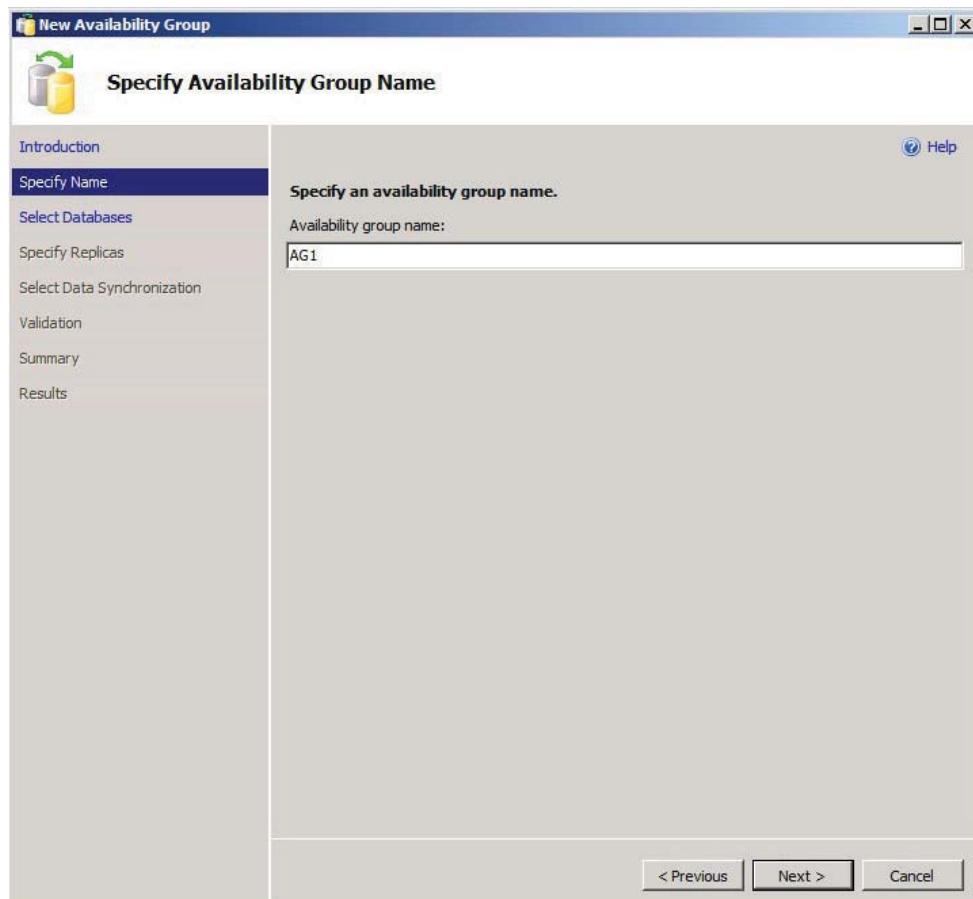


Now you are ready to create the Availability Group.

1. Choose one of the SQL Servers as the primary replica; this SQL Server must have the database(s) that will be included in the Availability Group. In this example, it is SLAG01 and the AdventureWorks database.
2. In the Microsoft SQL Server Management Studio, click on to the SQL Server, in this example SLAG01, then right-click and choose New Availability Group Wizard, as shown in Figure

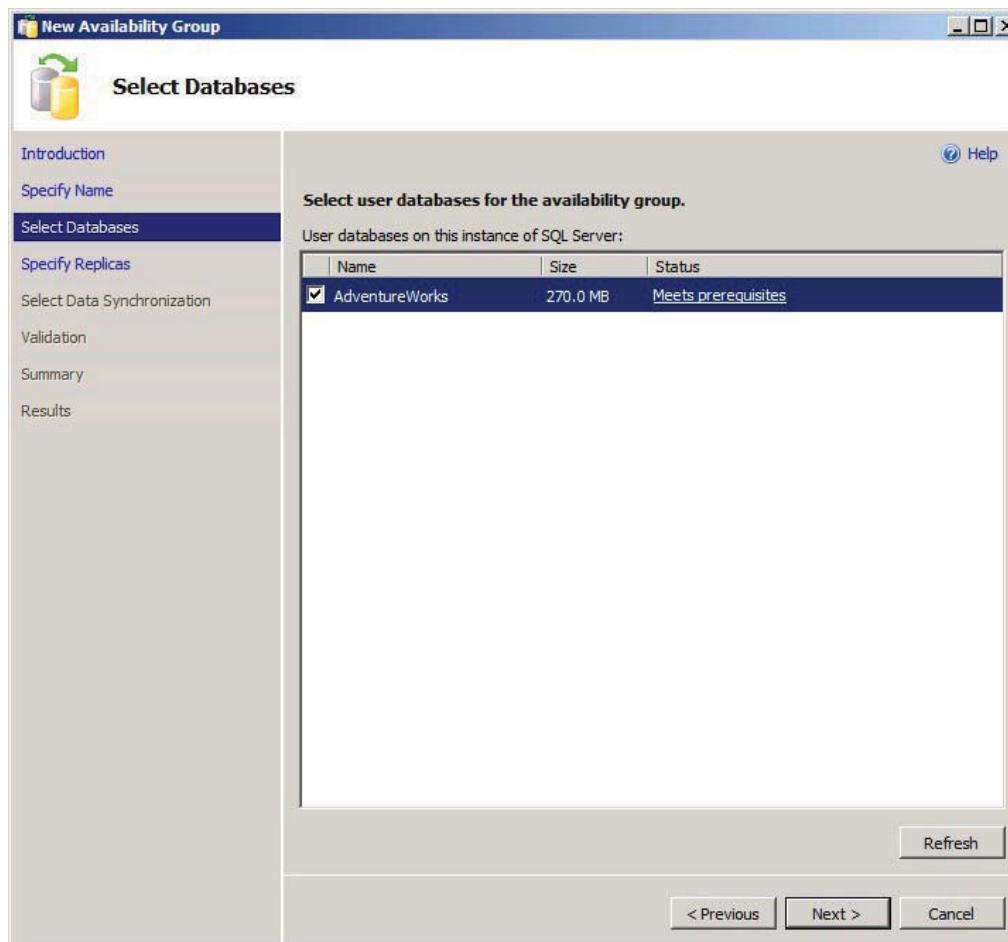


3. In the Introduction, click Next and in the Specify Name, choose a name, for example AG1, as shown in Figure

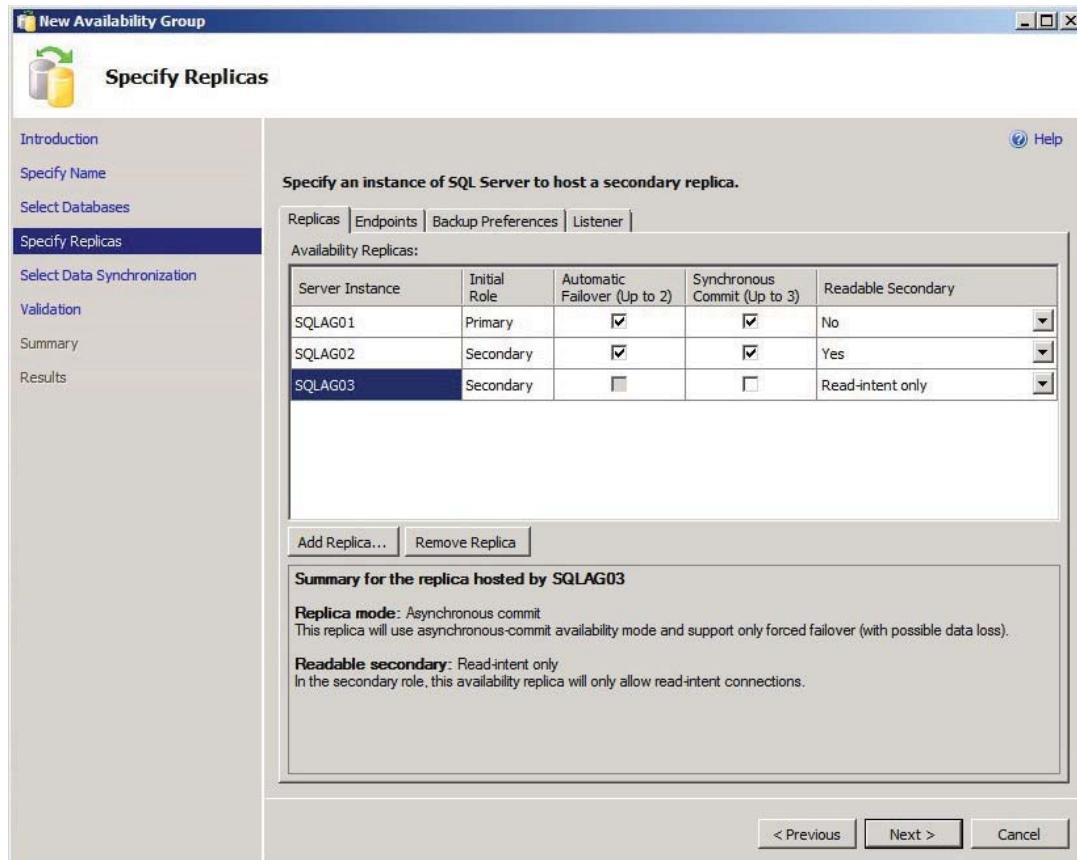


4. Click Next. Here on the Select Databases, click the check box for the databases, for example, AdventureWorks. If more than one database meets prerequisites, it will appear in the list for you to choose, as shown in Figure. For a database to be eligible to meet prerequisites, a database must:

- Be a user database. System databases cannot belong to an Availability Group.
- Be a read-write database. Read-only databases cannot be added to an Availability Group.
- Be a multiuser database.
- Not use AUTO_CLOSE.
- Use the full recovery model.
- Possess a full database backup.
- Reside on the SQL Server instance where you are creating the Availability Group and be accessible to the server instance.
- Not belong to another Availability Group.
- Not be configured for Database Mirroring.

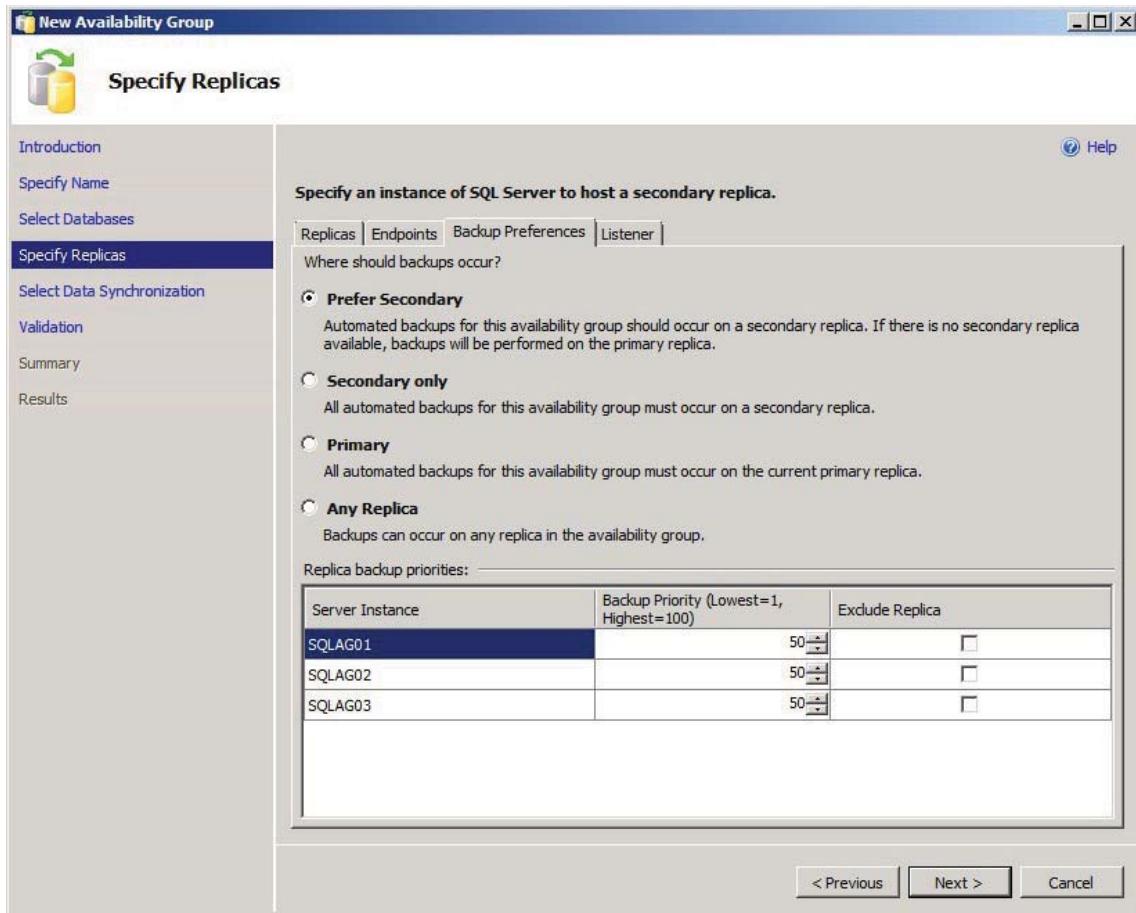


5. Click Next. On the Specify Replicas, under the Replicas tab, choose any secondary replica that you want to participate in the Availability Group, as shown in Figure. Table also provides a description for each option available in this step. Click Next again.

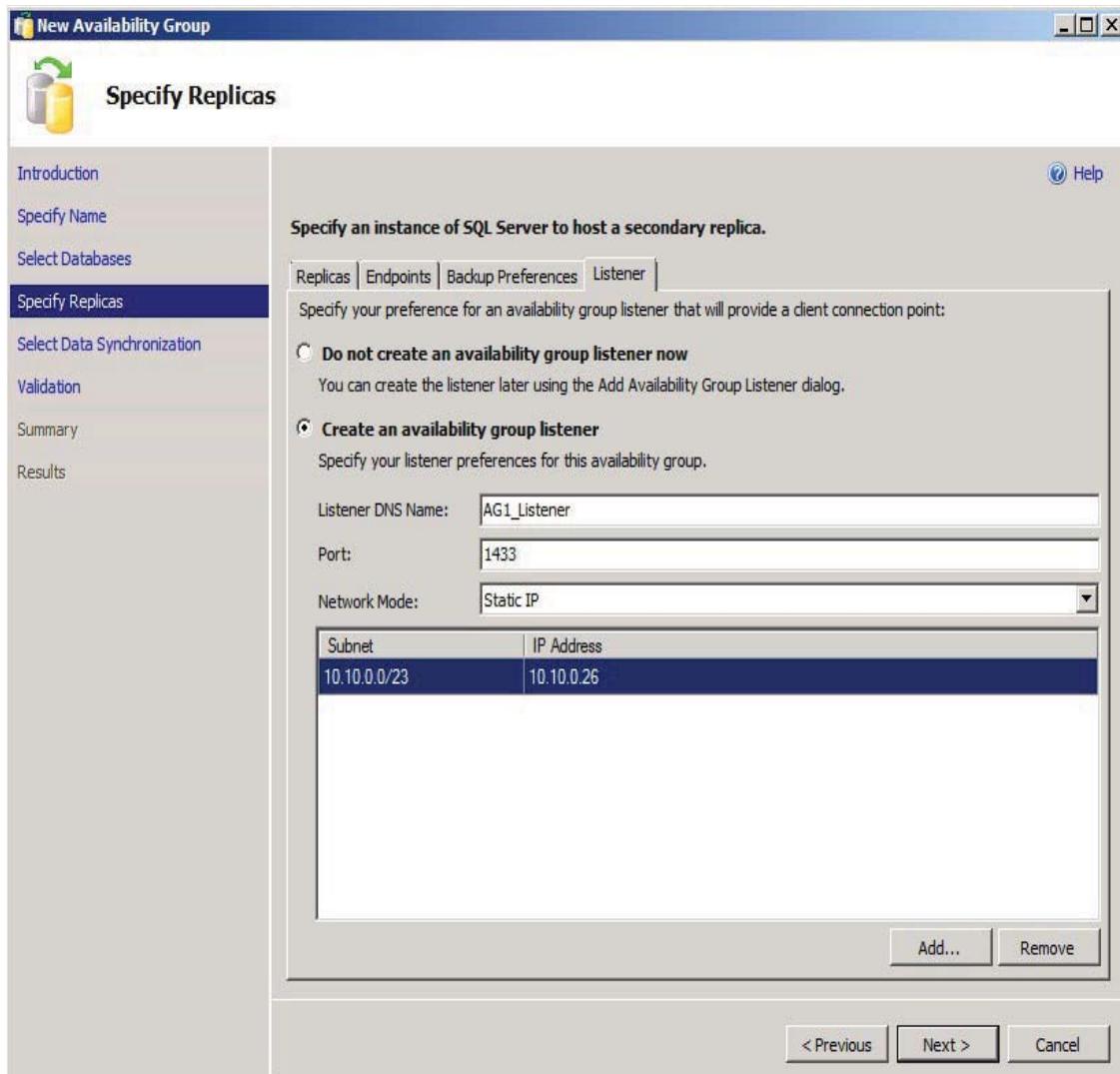


INITIAL ROLE	IDENTIFY THE INITIAL PRIMARY OR SECONDARY REPLICAS.
Automatic Failover (Up to 2)	If you want to configure automatic failover, choose up to two availability replica to be automatic failover partners where one of the partners chosen must be the initial primary replica. Both of these replicas use the synchronous-commit availability mode and only two replicas are supported in automatic failover.
Synchronous Commit (Up to 3)	If you selected Automatic Failover (Up to 2) for the two replica partners, Synchronous Commit (Up to 3) is automatically selected for them. Here you have the option to choose additional replicas to use synchronous-commit mode with only planned manual failover. Only three replicas are supported in synchronous-commit mode. Leave the checkbox blank if you want the replica to use asynchronous-commit availability mode. Then this replica will support only forced manual failover (with possible data loss).
Readable Secondary	No: No direct connections are allowed to the secondary databases of this replica. They are not available for read access. This is the default setting. Read-Intent only: Only direct read-only connections are allowed to secondary databases of this replica. The secondary database(s) are all available for read access. Yes: All connections are allowed to secondary databases of this replica, but only for read access. The secondary database(s) are all available for read access.

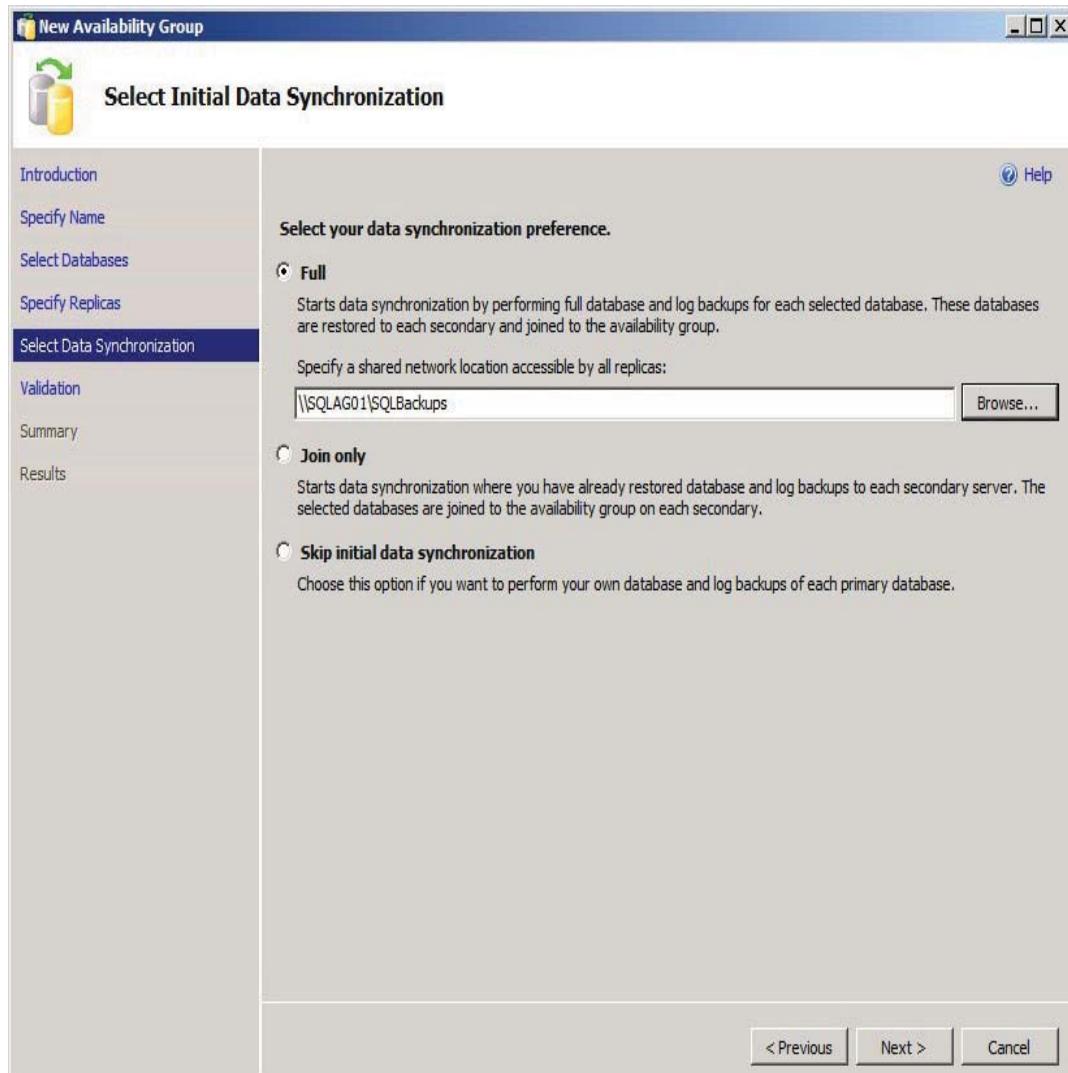
6. On the Specify Replicas, under the Backup Preferences tab, specify where backups should occur from the following options as shown in Figure:
- **Prefer Secondary:** Specifies that backups should occur on a secondary replica except when the primary replica is the only replica online. In that case, the backup should occur on the primary replica. This is the default option.
 - **Secondary Only:** Specifies that backups should never be performed on the primary replica. If the primary replica is the only replica online, then the backup should not occur.
 - **Primary:** Specifies that the backups should always occur on the primary replica. This option supports creating differential backups, which are not supported when backup is run on a secondary replica.
 - **Any Replica:** Specifies that you prefer for backup jobs to ignore the role of the availability replicas when choosing the replica to perform backups. Backup jobs might evaluate other factors such as backup priority of each availability replica in combination with its operational state and connected state.



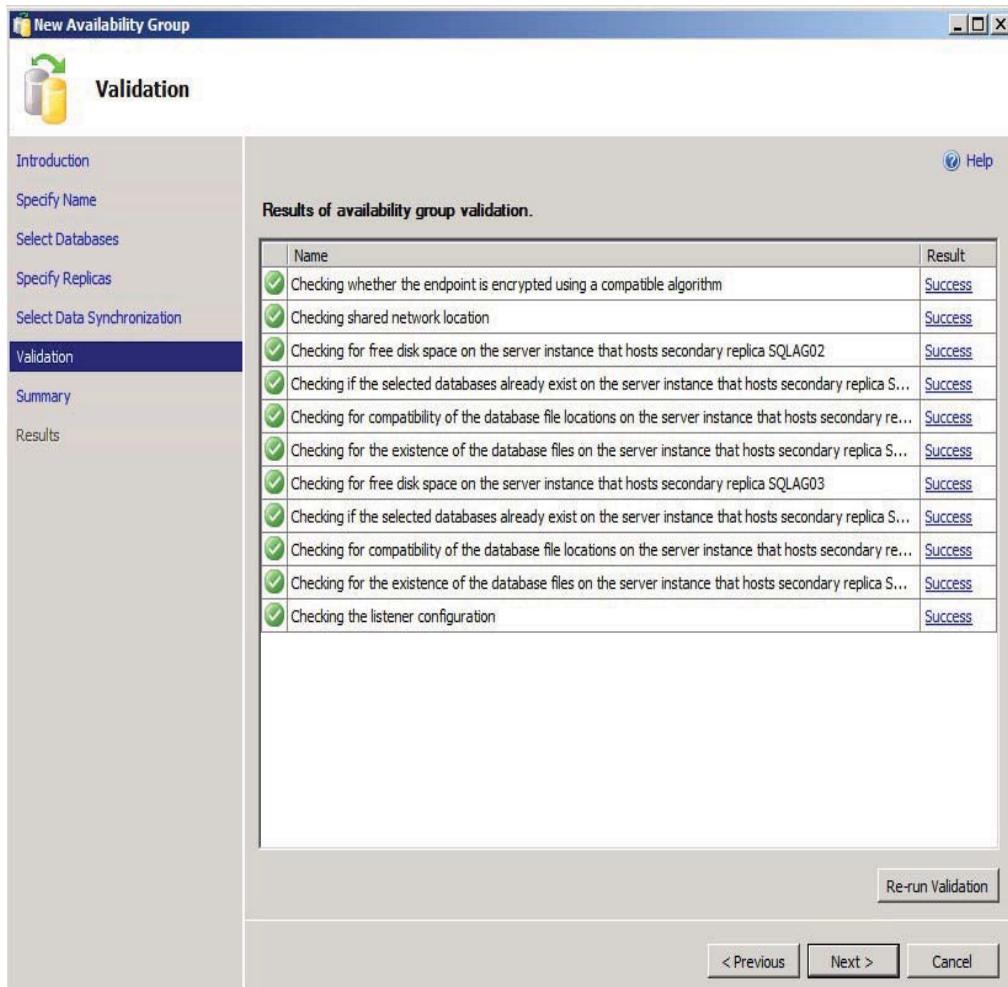
7. On Specify Replicas, under the Listener tab, choose Create An Availability Group Listener. Provide a Listener DNS Name, which is the network name that client applications use to connect to this Availability Group.
8. By default SQL Server listens on port 1433. If you changed the port number to a non-default value, provide that port number value.
9. Then choose either a DHCP or Static IP for the Availability Group Listener. In this example, a Static IP was given to the Availability Group Listener.
10. If you want, you can skip and create the Availability Group Listener later. For this example, the Listener is AG1_Listener with port number 1433, as shown in Figure Click Next.



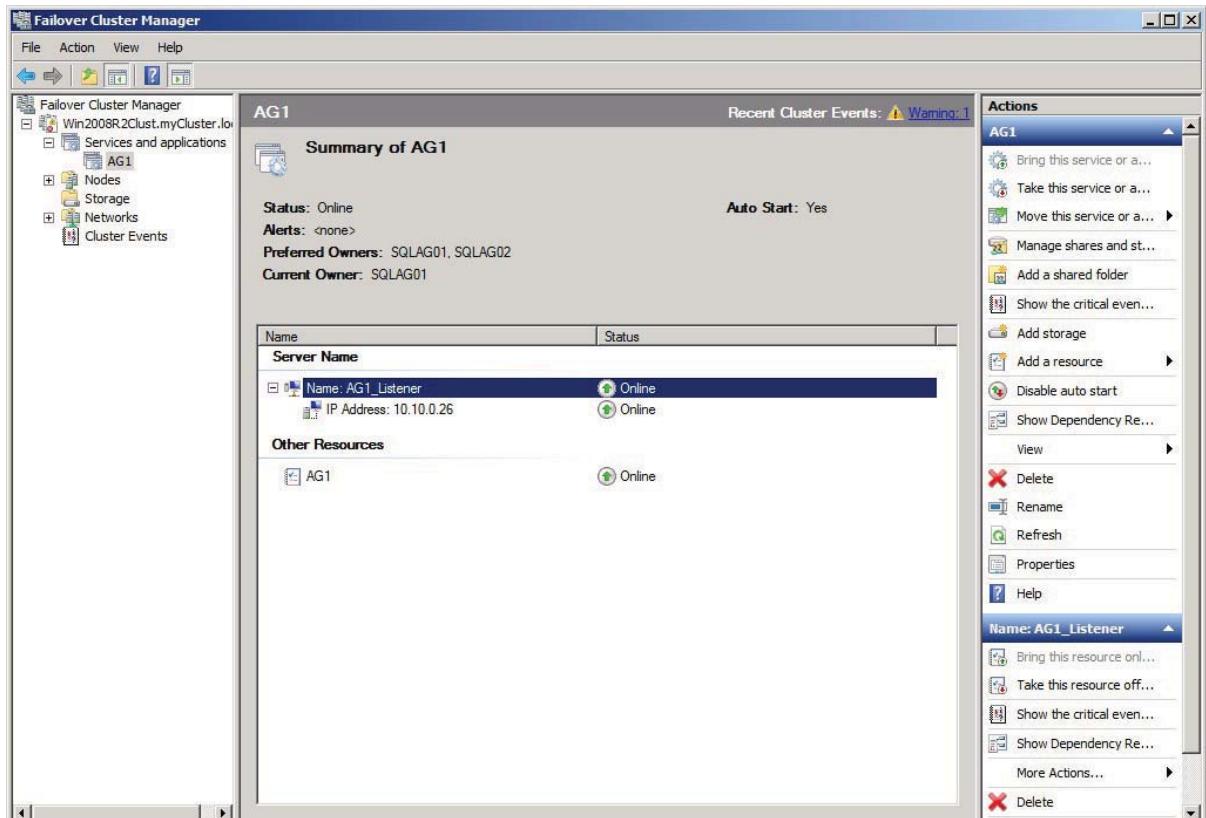
11. On the Select Initial Data Synchronization, choose Full. For Specify a Shared Network Location accessible by all replicas, choose a shared folder accessible by all participating replicas, for example \\SQLAG01\\SQLBackups, as shown in Figure. Also, you can choose Skip Initial Data Synchronization and manually restore to each secondary replica.



12. Click Next to run the validation, as shown in Figure. Then click next on the Summary. Click Finish to create the AG1 Availability Group



13. When you finish, you should be able to go to the Windows Failover Cluster Manager to see that the Availability Group has been created, as shown in Figure



ALWAYSON GROUP DASHBOARD

To monitor AlwaysOn Availability Groups, availability replicas, and availability databases in Microsoft SQL Server 2012, you can use the dashboard or data management views. From the dashboard, you can determine the health and performance of each Availability Group. To access the Availability Group Dashboard, follow these steps:

1. In SQL Server Management Studio, connect to the instance of SQL Server on which you want to run Availability Group Dashboard, which is either the primary or a secondary replica.
2. Expand the AlwaysOn High Availability, and the Availability Groups folders.
3. Right-click on your Availability Group name, in this example AG1, and then click Show Dashboard as shown in Figure

AG1:SQLAG01 X Dashboard: SQLAG01

AG1: hosted by SQLAG01 (Replica role: Primary)

Availability group state:	Healthy
Primary instance:	SQLAG01
Failover mode:	Automatic
Cluster state:	Win2008R2Clust (Normal Quorum)

Availability replica:

Name	Role	Failover Mode	Synchronization State	Issues
SQLAG01	Primary	Automatic	Synchronized	
SQLAG02	Second...	Automatic	Synchronized	
SQLAG03	Second...	Manual	Synchronizing	

Group by ▾

Name	Replica	Synchronization State	Failover Readin...	Issues
SQLAG01				
AdventureWorks	SQLAG01	Synchronized	No Data Loss	
SQLAG02				
AdventureWorks	SQLAG02	Synchronized	No Data Loss	
SQLAG03				
AdventureWorks	SQLAG03	Synchronizing	Data Loss	

Availability Group State	Displays the state of health for the Availability Group.
Role	The current role of the availability replica values is Primary or Secondary.
Primary Instance	Name of the server instance that is hosting the primary replica of the Availability Group.
Failover Mode	<p>Displays the failover mode for which the replica is configured. The possible failover mode values are:</p> <p>Automatic: indicates that one or more replicas is in automatic-failover mode</p> <p>Manual: indicates that no replica is in automatic-failover mode</p>

[Create Screen Clip](#)

Synchronization State	<p>Indicates whether a secondary replica is currently synchronized with primary replica; values are:</p> <p>Not Synchronized: database is not synchronized or has not yet been joined to the Availability Group</p> <p>Synchronized: the database is synchronized with the primary database on the current primary replica, if any, or on the last primary replica.</p> <p>NULL (Unknown): This value occurs when the local server instance cannot communicate with the WSFC failover cluster (that is the local node is not part of WSFC quorum).</p>
------------------------------	---

Configure an Existing Availability Group

Now you have Availability Group deployed and are relying on it for your high availability, disaster recovery, and reporting needs. At some point, a new requirement may arise if one of the following situations occurs:

The application is upgraded and an additional, new database is created that has dependencies to the other availability databases inside the Availability Group.

The company has increased reporting requirements and needs to deploy another read-only database to offload the work on the primary replica.

A disaster recovery site is identified and you are asked to deploy a replica there.

Backups are running on the primary replica and are impacting its performance and you have been asked to deploy a replica to perform backups.

You have consolidated several databases from the Availability Group and need to remove one of them.

You want to remove a replica as it is no longer needed, or it has been replaced with another replica running on faster hardware.

For situations like those in the preceding list, Availability Groups deliver the flexibility to start with at least a two-replica Availability Group, then deploy additional replicas and availability databases as your needs change. This section covers how to deploy additional replicas with a maximum of five or additional availability databases. You can remove replicas and availability databases when they are no longer needed as part of the Availability Group. When performing these operations in a production environment, consider doing so during your maintenance window and not during normal production time.

Add/remove replica

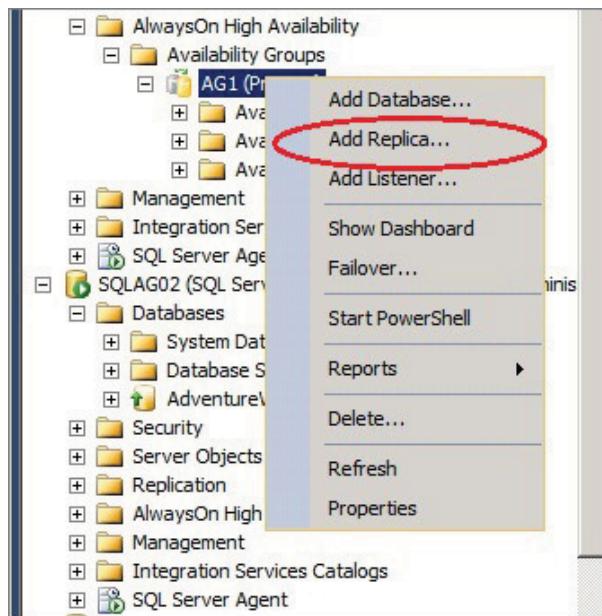
Anytime you want to add a replica, follow these steps:

1. Join the new replica to the Windows Cluster group from Windows Failover Cluster Manager
2. Enable AlwaysOn Availability Group for that SQL Server instance; refer to Figure

- 3.** In SQL Server Management Studio, on the Availability Groups folder, right-click the Availability Group for example AG1, and choose Add Replica as shown in Figure. Then follow the wizard to add a new replica.

To remove a replica, simply choose the replica, right-click, and Delete.

To remove an Availability Group, choose the Availability Group; then right-click and choose Delete.

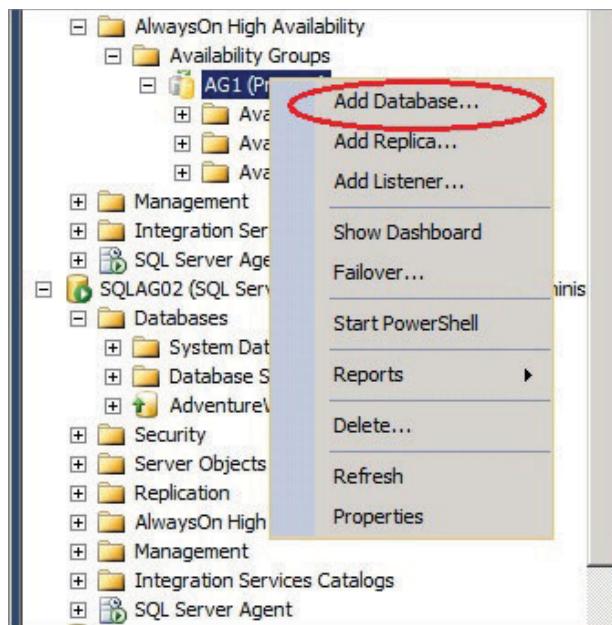


Add/remove a Database

Anytime you want to add an availability database, follow these steps:

- 1.** In SQL Server Management Studio, on the Availability Groups folder, right-click the Availability Group, for example AG1, and choose Add Database as shown in Figure
- 2.** Then, follow the wizard to add a new database.

To remove a database from the Availability Group, simply choose the database; then right-click and choose Delete.



Availability Group Failover Operation

There are two failover modes available: automatic and manual. Failover of the Availability Group depends on which of these two modes the replica is configured with.

In automatic failover mode (synchronous mode), when the primary replica goes down, the secondary replica takes over without any user intervention. The client applications that are connected using the Availability Group Listener, for example AG1_Listener, automatically reconnect to the new primary replica.

In manual failover mode (synchronous mode), in a failover, the secondary replica failover requires manual failover action. From within SQL Server Management Studio, perform the following operations to manually failover:

1. Connect to a SQL Server instance that hosts a secondary replica of the Availability Group that needs to be failed over, and expand the server tree.
2. Expand the AlwaysOn High Availability, and the Availability Groups folders.
3. Right-click the Availability Group to be failed over, and select Failover.

Moreover, failover can also be performed by Transact SQL. The following code snippet forces the AG1 Availability Group to fail over to the local secondary replica.

```
ALTER AVAILABILITY GROUP AG1 FAILOVER;
```

To force a failover with data loss, use this Transact SQL command.

```
ALTER AVAILABILITY GROUP AG1 FORCE_FAILOVER_ALLOW_DATA_LOSS;
```

After a forced failover, the secondary replica to which you failed over becomes the new primary replica, and all secondary databases are suspended. Before you resume any of the suspended databases however, you might need to reconfigure the Windows Failover Cluster's quorum and adjust the availability-mode configuration of the Availability Group. Consider these two scenarios:

1. If you failed over outside of the automatic failover set of the Availability Group, adjust the quorum votes of the Windows Failover Cluster nodes to reflect your new Availability Group configuration.
2. If you failed over outside of the synchronous-commit failover set, consider adjusting the availability mode and failover mode on the new primary replica and on remaining secondary replicas, to reflect your desired synchronous-commit and automatic failover configuration.

For each scenario, you must manually resume each suspended database individually. On resuming, a secondary database initiates data synchronization with the corresponding primary database. When the former primary replica becomes available, it switches to the secondary replica role, becoming a secondary replica, and immediately suspends its now-secondary databases. Under the asynchronous commit mode, the accumulated unsent log is a possibility on any of the new secondary databases. Resuming a new secondary database causes it to discard unsent log records and to roll back any changes that were never received by the now-primary replica.

If an availability replica that failed will not be returning to the availability replica or will return too late for you to delay transaction log truncation on the new primary database, consider removing the failed replica from the Availability Group.

Suspend an Availability Database

During a performance bottleneck, you may decide to suspend a secondary availability database. An availability database that is suspended means no transaction record data movement occurs and the transaction log on the primary replica keeps growing and cannot be truncated. If you suspend a secondary database on a secondary availability replica, only the local secondary database is suspended. If the suspend database is on the primary replica, transaction record data movement is suspended to all secondary databases on every secondary replica. When a secondary database is suspended, its database state is changed to SUSPENDED and it begins to fall behind the primary database. The primary database remains available, and if you have only one secondary replica, the primary database runs exposed.

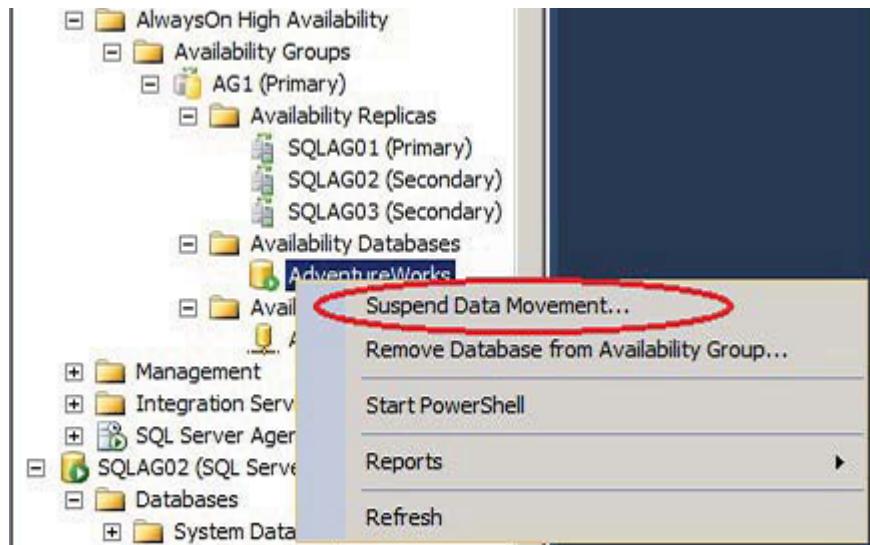
To suspend a database using SQL Server Management Studio, follow these steps:

1. In SQL Server Management Studio, connect to the SQL Server instance that hosts the availability replica on which you want to suspend a database.
2. Expand the AlwaysOn High Availability, and then the Availability Groups folders.
3. Expand the Availability Databases folder, right-click the database, and click Suspend Data Movement, as shown in Figure
4. In the Suspend Data Movement dialog box, click OK.

To suspend a database using Transact-SQL, follow these steps:

1. Connect to the SQL Server instance that hosts the replica whose database you want to suspend.
2. Suspend the secondary database by using the following ALTER DATABASE statement:

```
ALTER DATABASE database_name SET HADR SUSPEND;
```



Resume an Availability Database

After a database has been suspended, when the replica is resumed, the initial state is SYNCHRONIZING until it catches up. The primary database eventually resumes all its secondary databases that were suspended as the result of suspending the primary database. To resume a database using SQL Server Management Studio, follow these steps:

1. In SQL Server Management Studio, connect to the server instance that hosts the availability replica on which you want to resume a database.
2. Expand the AlwaysOn High Availability, and then the Availability Groups folders.
3. Expand the Availability Databases folder, right-click the database, and click Resume Data Movement.
4. In the Resume Data Movement dialog box, click OK.

To resume a database using Transact-SQL, follow these steps:

1. Connect to the server instance that hosts the database you want to resume.
2. Resume the secondary database by using the following ALTER DATABASE statement like so:

```
ALTER DATABASE database_name SET HADR RESUME;
```

Availability Group Listener Network Name

When the Availability Group fails over and the client applications have the network name in the connection strings, the network name directs connections to the new primary replica. You must create a network name that is unique in the domain for each Availability Group. As you create a network name, the IP address is assigned to the network name. Only the TCP protocol is supported for using a network name and, optionally, an IP to connect to an Availability Group. In the connection strings that your client applications use to connect to the databases in the Availability Group, specify the Availability Group network name rather than the server name; then applications can connect directly to the current primary replica. Following are examples of client application connection strings:

```
Server=tcp:MyNetworkName;Database=AdventureWorks;IntegratedSecurity=SSPI
```

```
Server=tcp:MyNetworkName,1433;Database=AdventureWorks;IntegratedSecurity=SSPI
```

ACTIVE SECONDARY FOR SECONDARY READ-ONLY

This section covers the secondary replica capabilities offered by Availability Groups in relation to running reports and read-only queries on the secondary replicas. Availability Groups enable read-only secondary replicas. The primary purpose of secondary replicas is for high-availability and disaster recovery, but secondary replicas can be configured to offload read-only queries for running reports from the primary replica. Then, the primary replica can better deliver the mission critical data activity while read-only reporting is executed on a secondary replica. When configured in read-only, the secondary replica is in near real time, just seconds behind data changes from the primary replica because the latency of transaction log synchronization impacts data freshness and the redo thread to update the data in each secondary replica before it is available to a query.

Read-Only Access Behavior

When you configure read-only access for a secondary replica, all databases in the Availability Group allow read-only. The behavior that determines the type of access allowed when a replica is in a secondary role is shown in Figure 25-6 under Readable Secondary column. Read-only access does not mean that the secondary databases are set to read-only. It means that user connections to the secondary databases have read-only access to the data. Even though you cannot write any data to the secondary databases, you can write to individual databases that do not belong to the Availability Group, including system databases such as tempdb. An instance of SQL Server can concurrently host multiple availability replicas along with databases that do not belong to an Availability Group.

As shown in Table, the initial primary replica, SLAG01, enables read-write connections, and the secondary replica, SLAG02, enables all direct connections, including connections that do not include the Application Intent property. When a failover occurs the replicas switch roles; SLAG02, which now runs under the primary role, enables read-write connections, whereas SLAG01, which now runs under the secondary role, disallows all direct connections.

REPLICA NAME	INITIAL ROLE	READABLE SECONDARY
SLAG01	Primary	No
SLAG02	Secondary	Yes

BACKUP ON THE SECONDARY REPLICA

One of the features of AlwaysOn in SQL Server 2012 is that you can run backups from any availability primary or secondary replicas; therefore, you can offload backup that is high IO and CPU operation from the primary replica to a secondary replica. See the “Allowing Read-Only Access to Secondary Replicas” section for more information. By offloading the backup operations to a secondary replica, you can use the primary replica to run uninterrupted, mission-critical workloads that run the business. In addition, backups can be performed on either synchronous or asynchronous replicas, and transaction log backups across the replicas can be combined to form a single transaction log chain.

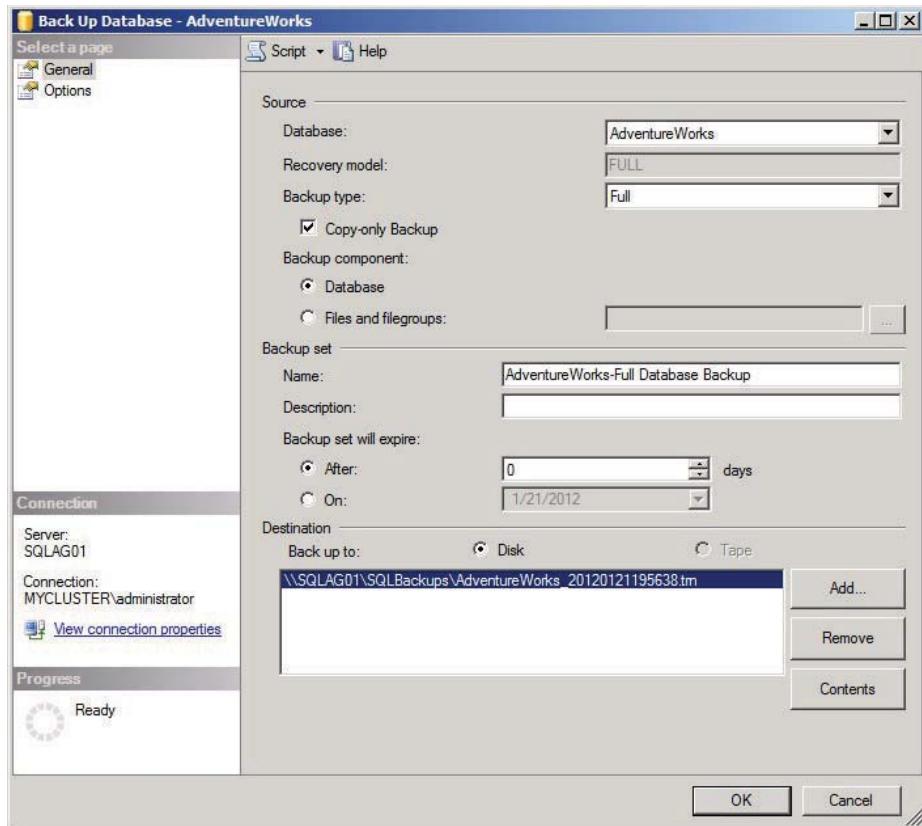
A copy-only full backup, as shown in Figure, is supported on the secondary replicas for full database, files, and filegroups backups and has the following characteristics:

A *copy-only full backup* is a SQL Server backup that is independent of the sequence of conventional SQL Server backups. It doesn't affect the overall backup nor restore procedures for the database.

Copy-only full backups are supported in all recovery models.

A copy-only full backup cannot serve as a differential base or differential backup and does not affect the differential base. Differential backups are not supported on the secondary replicas.

The secondary replica backup can be performed only when that secondary replica can communicate with the primary replica, and it is in synchronized or in the process of synchronizing.



MONITORING AND TROUBLESHOOTING

Like any solution, you need to monitor an Availability Group as part of regular operations to identify any issues with the Availability Group that may prevent the data from being synchronized with the secondary replicas or if failover to a secondary replica fails. To monitor Availability Groups, availability replicas, and availability databases using SQL Server Management Studio, follow these steps:

1. In SQL Server Management Studio, connect to the instance of SQL Server on which you want to monitor an Availability Group and click the server name.
2. Expand AlwaysOn High Availability, and the Availability Groups folders.
3. The Object Explorer Details pane displays every Availability Group for which the connected server instance hosts a replica. For each Availability Group, the Server Instance (Primary) column displays the name of the server instance currently hosting the primary replica. To display more information about a given Availability Group, select it in Object Explorer. The Object Explorer Details pane then displays the Availability Replicas and Availability Databases for the Availability Group.

To perform detail monitoring and troubleshooting, use the DMVs provided in Table to identify, verify, and then troubleshoot an Availability Group.

<code>sys.availability_groups</code>	Returns a row for each Availability Group for which the local instance of SQL Server hosts an availability replica. <small>Create Screen Clip</small>
<code>sys.dm_hadr_availability_group_states</code>	Returns a row for each Availability Group that possesses an availability replica on the local instance of SQL Server.
<code>sys.availability_replicas</code>	Returns a row for every availability replica in each availability group for which the local instance of SQL Server hosts an availability replica.
<code>sys.dm_hadr_availability_replica_states</code>	Returns a row showing the state of each local availability replica and a row for each remote availability replica in the same availability group.
<code>sys.dm_hadr_database_replica_states</code>	Returns a row for each database that is participating in any Availability Group for which the local instance of SQL Server is hosting an availability replica.
<code>sys.dm_hadr_database_replica_cluster_states</code>	Returns a row containing information to provide detail into the health of the availability databases in each Availability Group on the Windows Server Failover Clustering (WSFC) cluster.
<code>sys.availability_group_listener_ip_addresses</code>	Returns a row for every IP address that is currently online for an Availability Group listener.
<code>sys.availability_group_listeners</code>	For a given Availability Group, returns either zero rows indicating that no network name is associated with the Availability Group, or returns a row for each availability-group listener configuration in the WSFC cluster. <small>Create Screen Clip</small>
<code>sys.dm_tcp_listener_states</code>	Returns a row containing dynamic-state information for each TCP listener.

Chapter 21

High Availability: Interoperability and Coexistence

Database Mirroring and Log Shipping:

A given database can be mirrored or log shipped; it can also be simultaneously mirrored and log shipped. To choose what approach to use, consider the following:

- How many destination servers do you require? If you require only a single destination database, database mirroring is the recommended solution.
- If you require more than one destination database, you need to use log shipping, either alone or with database mirroring. Combining these approaches gives you the benefits of database mirroring along with the support for multiple destinations provided by log shipping.
- If you need to delay restoring log on the destination database (typically, to protect against logical errors), use log shipping, alone or with database mirroring.

The principal database in a mirroring session can also act as the primary database in a log shipping configuration, or vice versa, as the log shipping backup share is intact. The database mirroring session runs in any operating mode, whether synchronous (with transaction safety set to FULL) or asynchronous (with transaction safety set to OFF).

Typically, when combining log shipping and database mirroring, the mirroring session is established before log shipping, although this is not required. Then the current principal database is configured as the log shipping primary (*the principal/primary database*), along with one or more remote secondary databases. Also, the mirror database must be configured as a log shipping primary (*the mirror/primary database*). The log shipping secondary databases should be on different server instances than either the principal/primary server or mirror/primary server.

During a log shipping session, backup jobs on the primary database create log backups in a backup folder. From there, the backups are copied by the copy jobs of the secondary servers. For the backup jobs and copy jobs to succeed, they must have access to the log shipping backup folder. To maximize availability of the primary server, we recommend that you establish the backup folder in a shared backup location on a separate host computer. Ensure that all the log shipping servers, including the mirror/primary server, can access the shared backup location (*known as a backup share*).

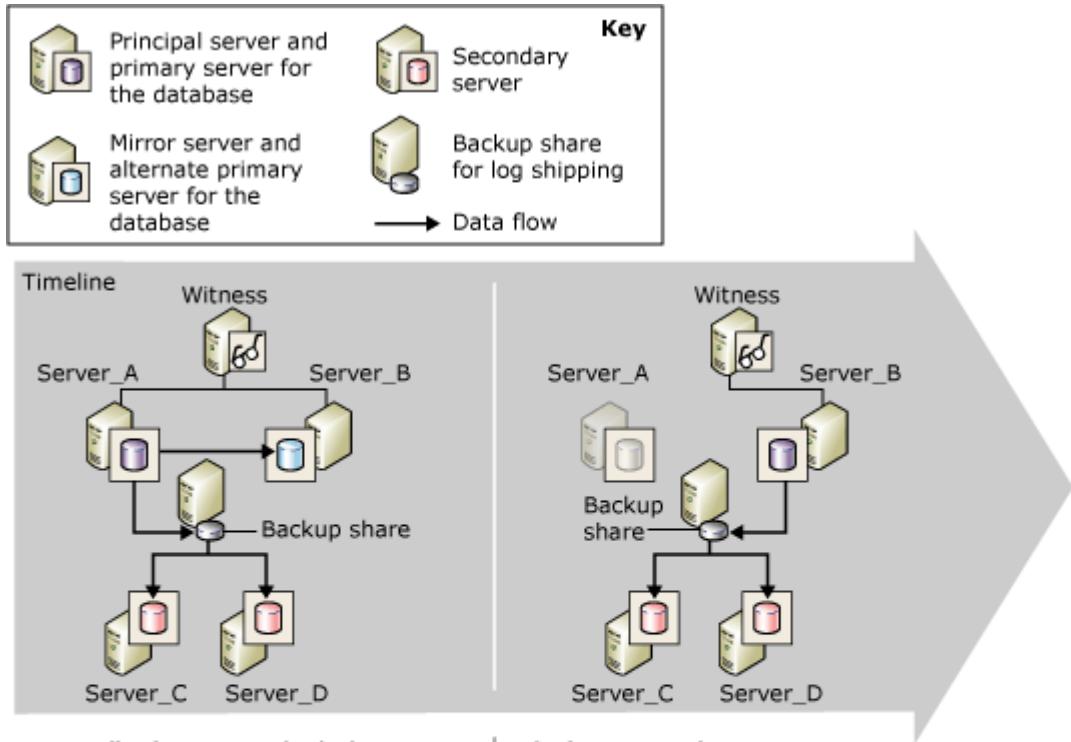
To allow log shipping to continue after database mirroring fails over, you must also configure the mirror server as a primary server, using the same configuration you use for the primary on the principal database. The mirror database is in the restoring state, which prevents the backup jobs from backing up the log on the mirror database. This ensures that the mirror/primary database does not interfere with the principal/primary database whose log backups are currently being copied by secondary servers. To prevent spurious alerts, after the backup job executes on the mirror/primary database, the backup job logs a message to the **log_shipping_monitor_history_detail** table, and the agent job returns a status of success.

The mirror/primary database is inactive in the log shipping session. However, if mirroring fails over, the former mirror database comes online as the principal database. At that point, that database also becomes active as the log shipping primary database. The log shipping backup jobs that were previously unable to ship log on that database, begin shipping log. Conversely, a failover causes the former principal/primary database to become the new mirror/primary database and enter the restoring state, and backup jobs on that database cease to backup log.

To run in high-safety mode with automatic failover the mirroring session is configured with an additional server instance known as the *witness*. If the principal database is lost for any reason after the database is synchronized and if the mirror server and witness can still communicate with each other, automatic failover occurs. An automatic failover causes mirror server to assume the principal role and bring its database online as the principal database. If the log shipping backup location is accessible to the new principal/primary server, its backup jobs begin to ship log backups to that location. The database mirroring synchronous mode guarantees that the log chain is unaffected by a mirroring failover and that only valid log is restored. The secondary servers continue to copy log backups without knowing that a different server instance has become the primary server.

Failing over from the Principal to the Mirror database

The following figure shows how log shipping and database mirroring work together when mirroring is running in high-safety mode with automatic failover. Initially, **Server_A** is both the principal server for mirroring and the primary server for log shipping. **Server_B** is the mirror server and is also configured as a primary server, which is currently inactive. **Server_C** and **Server_D** are log shipping secondary servers. To maximize availability of the log shipping session, the backup location is on a share directory on a separate host computer.



1. Initially, Server_A is both the principal server for mirroring and the primary server for log shipping. Server_B is the mirror server and is also configured as an alternate primary server, which is currently inactive.

Server_C and Server_D are log shipping secondary servers that copy shipped log backups from a backup share on a remote host computer.

2. Server_A is lost.

Server_B becomes both the principal server for mirroring and the active primary server for log shipping and begins to ship log backups to the backup share.

Server_C and Server_D continue to copy the backups from that share.

When Server_A returns it will become the mirror server.

After a mirroring failover, the primary server name defined on the secondary server is unchanged. .

Setting Up Mirroring and Log Shipping Together

To set up database mirroring and log shipping together, the following steps are required:

1. Restore backups of the principal/primary database with NORECOVERY onto another server instance to be later used as database mirroring mirror database for the principal/primary database.
2. Set up database mirroring.
3. Restore backups of the principal/primary database to other server instances to be later used as log shipping secondary databases for the primary database.
4. Set up log shipping on the principal database as the primary database for one or more secondary databases.

You should set up a single share as the backup directory (a backup share). This ensures that after role switching between the principal and mirror servers, backup jobs continue to write to the same directory as before. A best practice is to ensure that this share is located on a different physical server from the servers hosting the databases involved in mirroring and log shipping.

5. Manually failover from the principal to the mirror.
6. Set up log shipping on the new principal (previously mirror) as the primary database.
7. Perform another manual failover to fail back to the original principal.

Database Mirroring and Database Snapshots

You can take advantage of a mirror database that you are maintaining for availability purposes to offload reporting. To use a mirror database for reporting, you can create a database snapshot on the mirror database and direct client connection requests to the most recent snapshot. A database snapshot is a static, read-only, transaction-consistent snapshot of its source database as it existed at the moment of the snapshot's creation. To create a database snapshot on a mirror database, the database must be in the synchronized mirroring state.

Unlike the mirror database itself, a database snapshot is accessible to clients. As long as the mirror server is communicating with the principal server, you can direct reporting clients to connect to a snapshot. Note that because a database snapshot is static, new data is not available. To make relatively recent data available to your users, you must create a new database snapshot periodically and have applications direct incoming client connections to the newest snapshot.

A new database snapshot is almost empty, but it grows over time as more and more database pages are updated for the first time. Because every snapshot on a database grows incrementally in this way, each database snapshot consumes as much resources as a normal database. Depending on the configurations of the mirror server and principal server, having an excessive number of database snapshots on a mirror database might decrease performance on the principal database. Therefore, we recommend that you keep only a few relatively recent snapshots on your mirror databases. Typically, after you create a replacement snapshot, you should redirect incoming queries to the new snapshot and drop the earlier snapshot after any current queries complete.

If role switching occurs, the database and its snapshots are restarted, temporarily disconnecting users. Afterwards, the database snapshots remain on the server instance where they were created, which has become the new principal database. Users can continue to use the snapshots after the failover. However, this places an additional load on the new principal server. If performance is a concern in your environment, we recommend that you create a snapshot on the new mirror database when it becomes available, redirect clients to the new snapshot, and drop all of the database snapshots from the former mirror database.

Create the first database snapshot on the mirror of **AdventureWorks2008R2**.

```
CREATE DATABASE AdventureWorks2008R2_0600
ON (NAME = 'datafile', FILENAME = 'F:\AdventureWorks2008R2_0600.SNP')
AS SNAPSHOT OF AdventureWorks2008R2;
```

Database Mirroring and Failover Clustering

Failover clusters provide high-availability support for an entire Microsoft SQL Server instance, in contrast to database mirroring, which provides high-availability support for a single database. Database mirroring works between failover clusters and, also, between a failover cluster and a nonclustered host.

Typically, when mirroring is used with clustering, the principal server and mirror server both reside on clusters, with the principal server running on the failover clustered instance of one cluster and the mirror server running on the failover clustered instance of a different cluster. You can establish a mirroring session in which one partner resides on the failover clustered instance of a cluster and the other partner resides on a separate, unclustered computer, however.

If a cluster failover makes a principal server temporarily unavailable, client connections are disconnected from the database. After the cluster failover completes, clients can reconnect to the principal server on the same cluster, or on a different cluster or an unclustered computer, depending on the operating mode.

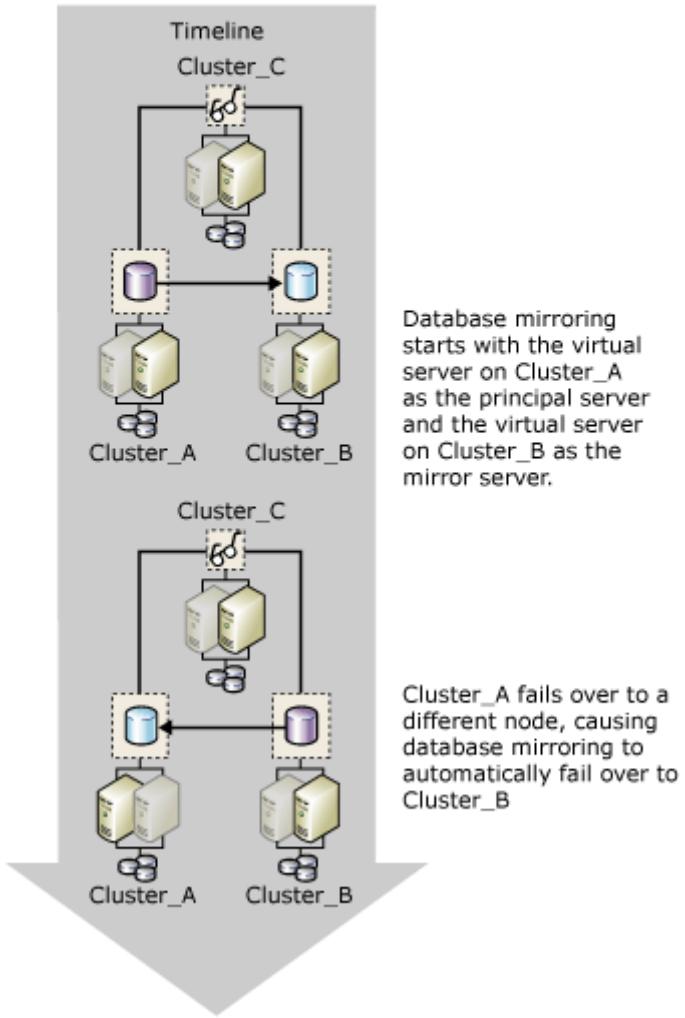
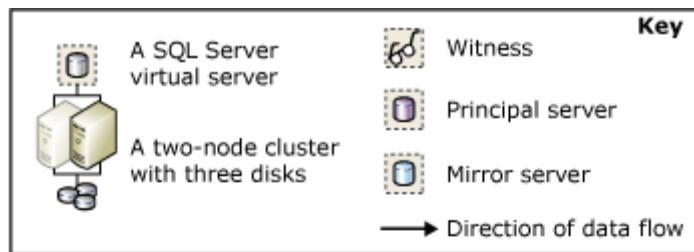
When deciding how to configure database mirroring in a clustered environment, the operating mode you use for mirroring is significant.

High-Safety mode Session with Automatic Failover

If you intend to mirror a database in high-safety mode with automatic failover, a two-cluster configuration is recommended for the partners. This configuration provides maximum availability. The witness can reside either on a third cluster or on an unclustered computer.

If the node running the current principal server fails, automatic failover of the database begins within a few seconds, while the cluster is still failing over to another node. The database mirroring session fails over to the mirror server on the other cluster or unclustered computer, and the former mirror server becomes the principal server. The new principal server rolls forward its copy of the database as quickly as possible and brings it online as the principal database. After the cluster failover completes, which typically takes several minutes, the failover clustered instance that was formerly the principal server becomes the mirror server.

The following illustration shows an automatic failover between clusters in a mirroring session running in high-safety mode with a witness (which supports automatic failover).



The three server instances in the mirroring session reside on three distinct clusters: **Cluster_A**, **Cluster_B**, and **Cluster_C**. On each cluster, a default instance of SQL Server is running as a SQL Server failover clustered instance. When the mirroring session starts, the failover clustered instance on **Cluster_A** is the principal server, the failover clustered instance on **Cluster_B** is the mirror server, and the failover clustered instance on **Cluster_C** is the witness in the mirroring session. Eventually, the active node on **Cluster_A** fails, which causes the principal server to become unavailable.

Before the cluster has time to fail over, the loss of the principal server is detected by the mirror server, with the help of the witness. The mirror server rolls forward its database and brings it online as the new principal database as quickly as possible. When **Cluster_A** finishes failing over, the former principal

server is now the mirror server, and it synchronizes its database with the current principal database on **Cluster_B**.

High-Safety Mode Session Without Automatic Failover

If you are mirroring a database in high-safety mode without automatic failover, another node in the cluster will act as the principal server if the node running the current principal server fails. Note that while the cluster is unavailable, the database is unavailable.

High-Performance Mode Session

If you intend to mirror a database in high-performance mode, consider placing the principal server on the failover clustered instance of a cluster and placing the mirror server on an unclustered server in a remote location. If the cluster fails over to a different node, the failover clustered instance will continue as the principal server in the mirroring session. If the entire cluster has problems, you can force service onto the mirror server.

Replication and Log Shipping

Log shipping involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients. This copy is known as the primary database. Updates made by clients to the primary database are propagated by means of log shipping to the other copy of the database, known as the secondary database. Log shipping involves applying the transaction log from every insertion, update, or deletion made on the primary database onto the secondary database.

Log shipping can be used in conjunction with replication, with the following behavior:

- Replication does not continue after a log shipping failover. If a failover occurs, replication agents do not connect to the secondary, so transactions are not replicated to Subscribers. If a failback to the primary occurs, replication resumes. All transactions that log shipping copies from the secondary back to the primary are replicated to Subscribers.
- If the primary is permanently lost, the secondary can be renamed so that replication can continue. The remainder of this topic describes the requirements and procedures for handling this case. The example given is the publication database, which is the most common database to log ship, but a similar process can also be applied to subscription and distribution databases.

Pre-Requisites:

- If a primary contains more than one publication database, log ship all of the publication databases to the same secondary.
- The installation path for the secondary server instance must be the same as the primary. User database locations on the secondary server must be the same as on the primary.

- Log shipping does not guarantee against data loss. A failure on the primary database can result in the loss of data that has not yet been backed up or for backups that are lost during the failure.

Log Shipping with Transactional Replication

For transactional replication, the behavior of log shipping depends on the sync with backup option. This option can be set on the publication database and distribution database; in log shipping for the Publisher, only the setting on the publication database is relevant.

Setting this option on the publication database ensures that transactions are not delivered to the distribution database until they are backed up at the publication database. The last publication database backup can then be restored at the secondary server without any possibility of the distribution database having transactions that the restored publication database does not have. This option guarantees that if the Publisher fails over to a secondary server, consistency is maintained between the Publisher, Distributor, and Subscribers. Latency and throughput are affected because transactions cannot be delivered to the distribution database until they have been backed up at the Publisher; if your application can tolerate this latency, we recommend that you set this option on the publication database. If the sync with backup option is not set, Subscribers might receive changes that are no longer included in the recovered database at the secondary server.

To configure transactional replication and log shipping with the sync with backup option

1. If the sync with backup option is not set on the publication database, execute `sp_replicationdboption '<publicationdatabasename>', 'sync with backup', 'true'`.
2. Configure log shipping for the publication database.
3. If the Publisher fails, restore the last log of the database to the secondary server, using the `KEEP_REPLICATION` option of `RESTORE LOG`. This retains all replication settings for the database.
4. Restore the msdb database and master databases from the primary to the secondary. If the primary was also a Distributor, restore the distribution database from the primary to the secondary.

These databases must be consistent with the publication database at the primary in terms of replication configuration and settings.

5. At the secondary server, rename the computer and then rename the Microsoft SQL Server instance to match the primary server name

Log Shipping with Merge Replication

1. Configure log shipping for the publication database.
2. If the Publisher fails, restore the last log of the database to the secondary server, using the `KEEP_REPLICATION` option of `RESTORE LOG`. This retains all replication settings for the database.

3. Restore the msdb database and master databases from the primary to the secondary. If the primary was also a Distributor, restore the distribution database from the primary to the secondary.

These databases must be consistent with the publication database at the primary in terms of replication configuration and settings.

4. At the secondary server, rename the computer and then rename the SQL Server instance to match the primary server name.
5. Synchronize the publication database with one or more subscription databases. This allows you to upload those changes made previously in the publication database, but not represented in the restored backup. The data that can be uploaded depends on the way in which a publication is filtered:
 - If the publication is not filtered, you should be able to bring the publication database up-to-date by synchronizing with the most up-to-date Subscriber.
 - If the publication is filtered, you might not be able to bring the publication database up-to-date. Consider a table that is partitioned such that each subscription receives customer data only for a single region: North, East, South, and West. If there is at least one Subscriber for each partition of data, synchronizing with a Subscriber for each partition should bring the publication database up-to-date. However, if data in the West partition, for example, was not replicated to any Subscribers, this data at the Publisher cannot be brought up-to-date. In this case, we recommend reinitializing all subscriptions so that the data at the Publisher and Subscribers converges.

Replication and Database Mirroring

Database mirroring can be used in conjunction with replication to provide availability for the publication database. Database mirroring involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients. This copy is known as the principal database. Updates made by clients to the principal database are applied on the other copy of the database, known as the mirror database. Mirroring involves applying the transaction log from every insertion, update, or deletion made on the principal database onto the mirror database.

Replication failover to a mirror is supported for publication databases only; it is not supported for the distribution database or subscription databases.

Pre-requisites:

- The principal and mirror must share a Distributor. We recommend that this be a remote Distributor, which provides greater fault tolerance if the Publisher has an unplanned failover.
- The Publisher and Distributor must be Microsoft SQL Server 2005 or a later version. Subscribers can be any version, but merge replication pull subscriptions from a version prior to SQL Server 2005 do not support failover; the agent in this case runs at the Subscriber and previous versions

of the agent are not mirror aware. Replication to such Subscribers resumes if the database fails back from the mirror to the principal.

- Replication supports mirroring the publication database for merge replication and for transactional replication with read-only Subscribers or queued updating Subscribers. Immediate updating Subscribers, Oracle Publishers, Publishers in a peer-to-peer topology, and republishing are not supported.
- Metadata and objects that exist outside the database are not copied to the mirror, including logins, jobs, linked servers, and so on. If you require the metadata and objects at the mirror, you must copy them manually.

Configuring Replication with Database Mirroring

1. Configure the Publisher.
 - a. We recommend using a remote Distributor. For more information about configuring distribution, see Configuring Distribution.
 - b. You can enable a database for snapshot and transactional publications and/or merge publications. For mirrored databases that will contain more than one type of publication, you must enable the database for both types at the same node using `sp_replicationdboption`. For example, you could execute the following stored procedure calls at the principal:

```
exec sp_replicationdboption @dbname='<PublicationDatabase>', @optname='publish',  
@value=true
```

```
exec sp_replicationdboption @dbname='<PublicationDatabase>',  
@optname='mergepublish', @value=true
```

2. Configure database mirroring.
3. Configure the mirror to use the same Distributor as the principal.

Specify the mirror name as the Publisher, and specify the same Distributor and snapshot folder that the principal uses. For example, if you are configuring replication with stored procedures, execute `sp_adddistpublisher` at the Distributor; and then execute `sp_adddistributor` at the mirror. For `sp_adddistpublisher`:

Set the value of the `@publisher` parameter to the network name of the mirror.

Set the value of the `@working_directory` parameter to the snapshot folder used by the principal.

4. Configure replication agents for failover.

Specify the mirror name for the `-PublisherFailoverPartner` agent parameter. Agent This parameter is required for the following agents to identify the mirror after failover:

- a. Snapshot Agent (for all publications)

- b. Log Reader Agent (for all transactional publications)
- c. Queue Reader Agent (for transactional publications that support queued updating subscriptions)
- d. Merge Agent (for merge subscriptions)
- e. SQL Server replication listener (replisapi.dll: for merge subscriptions synchronized using Web synchronization)
- f. SQL Merge ActiveX Control (for merge subscriptions synchronized with the control)

The Distribution Agent and Distribution ActiveX Control do not have this parameter because they do not connect to the Publisher.

5. Add the principal and mirror to Replication Monitor.

Failover Clustering and AlwaysOn Availability Groups

AlwaysOn Availability Groups, the high availability and disaster recovery solution introduced in SQL Server 2012, requires Windows Server Failover Clustering (WSFC). Also, though AlwaysOn Availability Groups is not dependent upon SQL Server Failover Clustering, you can use a failover clustering instance (FCI) to host an availability replica for an availability group. It is important to know the role of each clustering technology, and to know what considerations are necessary as you design your AlwaysOn Availability Groups environment.

Deploying AlwaysOn Availability Groups requires a Windows Server Failover Clustering (WSFC) cluster. To be enabled for AlwaysOn Availability Groups, an instance of SQL Server must reside on a WSFC node, and the WSFC cluster and node must be online. Furthermore, each availability replica of a given availability group must reside on a different node of the same WSFC cluster. The only exception is that while being migrated to another WSFC cluster, an availability group can temporarily straddle two clusters.

AlwaysOn Availability Groups relies on the Windows Failover Clustering (WSFC) cluster to monitor and manage the current roles of the availability replicas that belong to a given availability group and to determine how a failover event affects the availability replicas. A WSFC resource group is created for every availability group that you create. The WSFC cluster monitors this resource group to evaluate the health of the primary replica.

The quorum for AlwaysOn Availability Groups is based on all nodes in the WSFC cluster regardless of whether a given cluster node hosts any availability replicas. In contrast to database mirroring, there is no witness role in AlwaysOn Availability Groups.

The overall health of a WSFC cluster is determined by the votes of quorum of nodes in the cluster. If the WSFC cluster goes offline because of an unplanned disaster, or due to a persistent hardware or communications failure, manual administrative intervention is required. A Windows Server or WSFC cluster administrator will need to force a quorum and then bring the surviving cluster nodes back online in a non-fault-tolerant configuration.

SQL Server Failover Cluster Instances (FCIs) and Availability Groups

You can set up a second layer of failover at the server-instance level by implementing SQL Server failover clustering together with the WSFC cluster. An availability replica can be hosted by either a standalone instance of SQL Server or an FCI instance. Only one FCI partner can host a replica for a given availability group. When an availability replica is running on an FCI, the possible owners list for the availability group will contain only the active FCI node.

AlwaysOn Availability Groups does not depend on any form of shared storage. However, if you use a SQL Server failover cluster instance (FCI) to host one or more availability replicas, each of those FCIs will require shared storage as per standard SQL Server failover cluster instance installation.

Comparison of Failover Cluster Instances and Availability Groups

	Nodes within an FCI	Replicas within an availability group
Uses WSFC cluster	Yes	Yes
Protection level	Instance	Database
Storage type	Shared	Non-shared
Storage solutions	Direct attached, SAN, mount points, SMB	Depends on node type
Readable secondaries	No	Yes
Applicable failover policy settings	<ul style="list-style-type: none">• WSFC quorum• FCI-specific• Availability group settings³	<ul style="list-style-type: none">• WSFC quorum• Availability group settings
Failed-over resources	Server, instance, and database	Database only

SQL Server Failover Cluster Instances (FCIs) do not support automatic failover by availability groups, so any availability replica that is hosted by an FCI can only be configured for manual failover.

You might need to configure a Windows Server Failover Clustering (WSFC) cluster to include shared disks that are not available on all nodes. For example, consider a WSFC cluster across two data centers with three nodes. Two of the nodes host a SQL Server failover clustering instance (FCI) in the primary data center and have access to the same shared disks. The third node hosts a stand-alone instance of SQL

Server in a different data center and does not have access to the shared disks from the primary data center. This WSFC cluster configuration supports the deployment of an availability group if the FCI hosts the primary replica and the stand-alone instance hosts the secondary replica.

When choosing an FCI to host an availability replica for a given availability group, ensure that an FCI failover could not potentially cause a single WSFC node to attempt to host two availability replicas for the same availability group.

Disadvantages:

Do not use the Failover Cluster Manager to manipulate availability groups, for example:

- Do not change any availability group properties, such as the possible owners.
- Do not use the Failover Cluster Manager to fail over availability groups. You must use Transact-SQL or SQL Server Management Studio.

Chapter 22

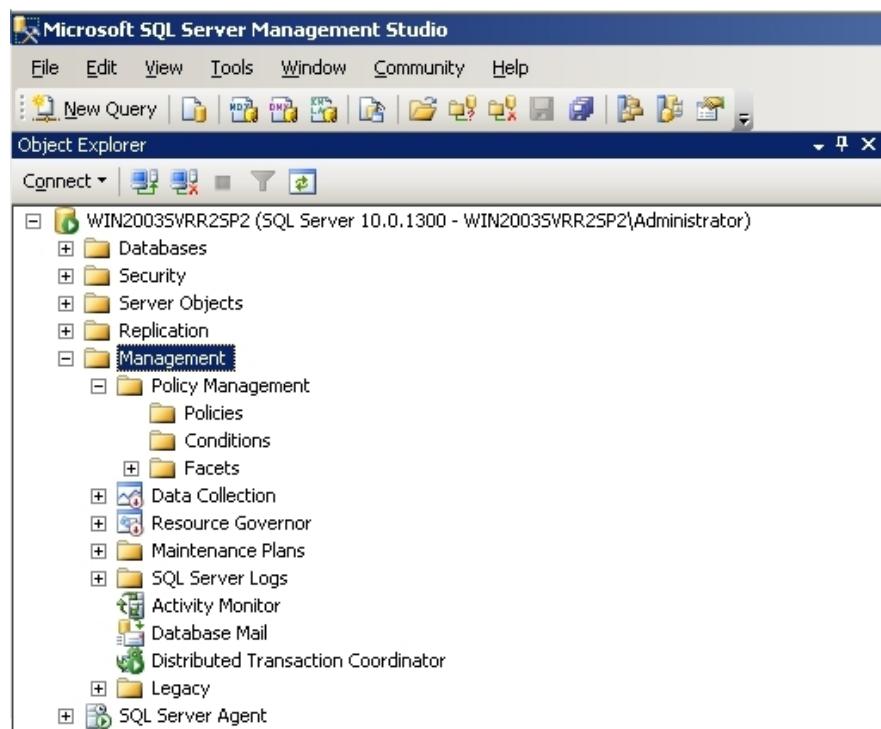
Optimizing SQL server

Policy based management:

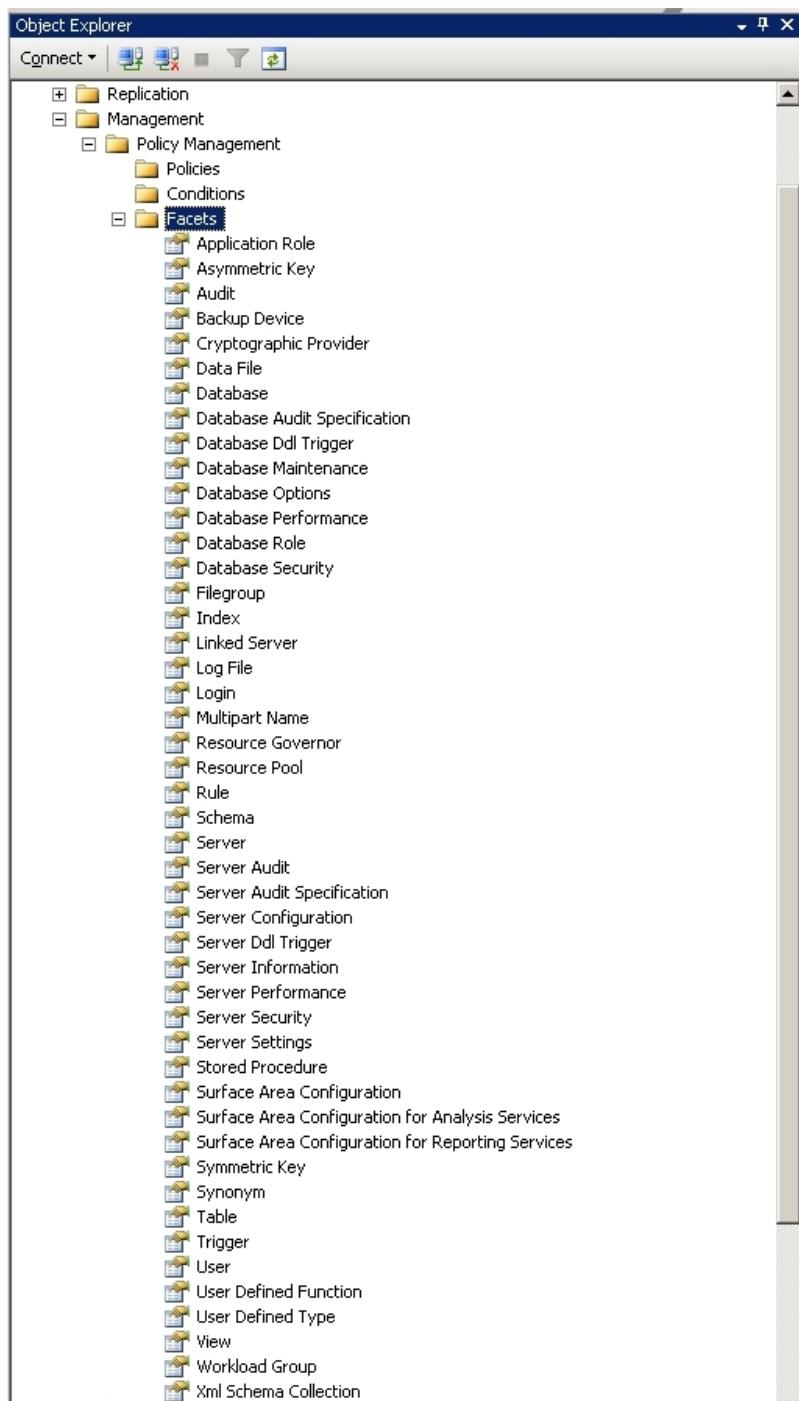
Policy-Based Management is indeed a new feature in SQL Server 2008. It allows you to define and enforce policies for configuring and managing SQL Server across the enterprise. Originally this feature was called the Declarative Management Framework but has since been renamed. There are a number of terms that we need to define in order to begin to understand Policy-Based Management:

- Target - an entity that is managed by Policy-Based management; e.g. a database, a table, an index, etc.
- Facet - a predefined set of properties that can be managed
- Condition - a property expression that evaluates to True or False; i.e. the state of a Facet
- Policy - a condition to be checked and/or enforced

Policy-Based Management is configured in SQL Server Management Studio (SSMS). Navigate to the Object Explorer and expand the Management node and the Policy Management node; you will see the Policies, Conditions, and Facets nodes:



Expand the Facet node to see the list of facets:



As you can see there is a rather comprehensive collection of facets predefined in SQL Server 2008, allowing you to manage just about every aspect of SQL Server. Double click on a facet to see the actual list of properties in the facet; e.g. double click the Database facet:

Description:	Exposes properties of the Database object.
Applicable target types:	Database
Properties:	
ActiveConnections	The number of active connections to the database.
AnsiNullDefault	Specifies whether the ANSI_NULL_DEFAULT database option is enabled.
AnsiNullsEnabled	Specifies whether the ANSI_NULLS_ENABLED database option is enabled.
AnsiPaddingEnabled	Specifies whether the ANSI_PADDING_ENABLED database option is enabled.
AnsiWarningsEnabled	Specifies whether the ANSI_WARNINGS_ENABLED database option is enabled.
ArithAbortEnabled	Specifies whether the ARITHMETICABORT database option is enabled.
AutoClose	Specifies whether the AUTOCLOSE database option is active.
AutoCreateStatistics	Specifies whether the AUTOCREATESTATISTICS database option is enabled.
AutoCreateStatisticsEnabled	Specifies whether statistics are automatically created for the database.
AutoShrink	Specifies whether the AUTOSHRINK database option is active.

These facet properties are used to specify a condition; e.g. AutoShrink = False means that you do not want to automatically shrink database files. A policy specifies an expression that evaluates to True or False. The expression can be made up of one or more conditions logically joined by And / Or.

In this tip we are going to gain an understanding of Policy-Based Management by walking through the following demonstration:

- Create a Condition
- Create a Policy
- Evaluate a Policy

The demo steps below were only tested on the February, 2008 Community Technology Preview (CTP) of SQL Server 2008.

Create a Condition

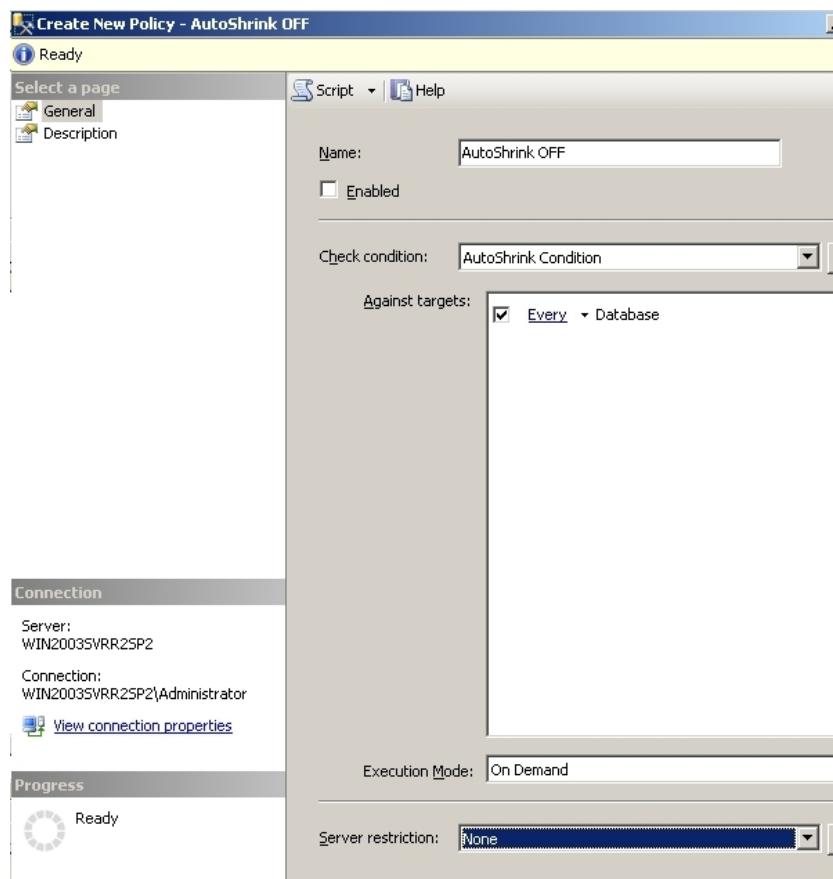
The starting point in Policy-Based Management is to create a Condition. Right click on Conditions in the SSMS Object Explorer (under the Management | Policy Management node) then select New Condition from the menu. Fill in the dialog as follows:

Name:	AutoShrink Condition																						
Facet:	Database																						
Expression:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th></th> <th>AndOr</th> <th>Field</th> <th>Operator</th> <th>Value</th> <th></th> </tr> </thead> <tbody> <tr> <td>▶</td> <td>...</td> <td>@AutoShrink</td> <td>=</td> <td>False</td> <td>...</td> </tr> <tr> <td>*</td> <td colspan="5">Click here to add a clause</td> </tr> </tbody> </table>						AndOr	Field	Operator	Value		▶	...	@AutoShrink	=	False	...	*	Click here to add a clause				
	AndOr	Field	Operator	Value																			
▶	...	@AutoShrink	=	False	...																		
*	Click here to add a clause																						

You select a single Facet for a Condition, then enter an Expression. The Expression evaluates to either True or False. This is the essence of Policy-Based Management which will test whether the Condition is True.

Create a Policy

Right click Policies in the SSMS Object Explorer (under the Management | Policy Management node) then select New Policy from the menu. Fill in the dialog as follows:



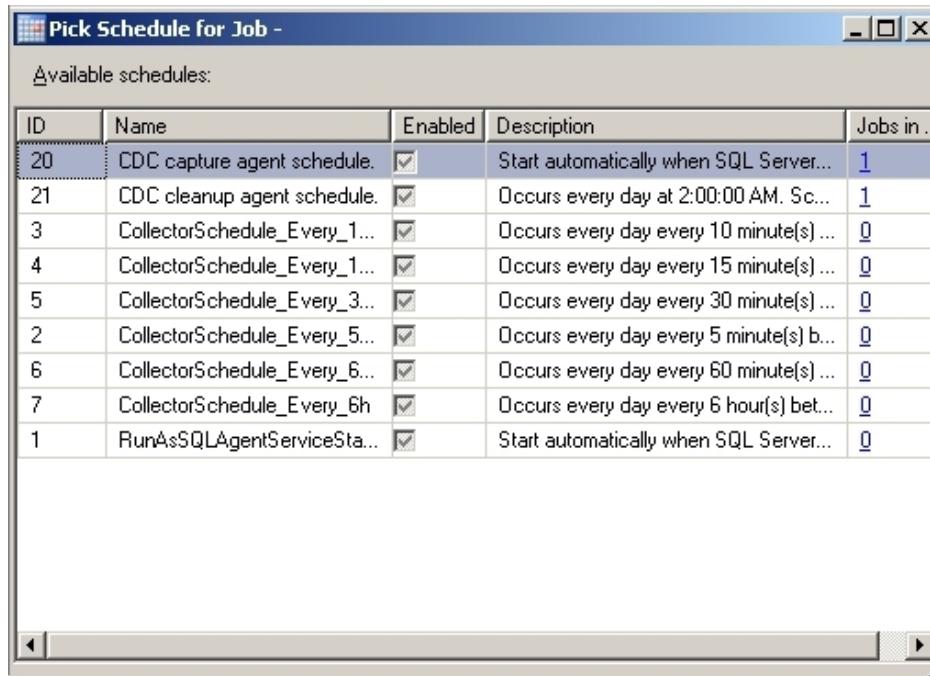
The Check Condition drop down will include the list of conditions that you have defined. You can check Every Database in the Against targets list, or you can click the glyph (between Every and Database) and define a condition. Execution Mode can have one of the following values:

- On Demand (this is the default)
- On Schedule
- On Change - Log Only
- On Change - Prevent

The On Demand option only evaluates the policy when a user right clicks on the policy in the SSMS Object Explorer and selects Evaluate from the menu.

The On Schedule option takes advantage of SQL Agent to execute a job on a particular schedule to check the policy. After selecting On Schedule from the Execution Mode drop down list, you can click either the Pick or New button.

To pick an existing schedule, make a selection from the available options:



To create a new schedule, fill in the familiar schedule dialog:

New Job Schedule

Name: Standard Policy-Based Management Schedule

Schedule type: Recurring Enabled

One-time occurrence

Date: 4/30/2008 Time: 8:47:21 PM

Frequency

Occurs: Weekly

Recurs every: 1 week(s) on

Monday Wednesday Friday Saturday Sunday

Tuesday Thursday

Daily frequency

Occurs once at: 12:00:00 AM

Occurs every: 1 hour(s)

Starting at: 12:00:00 AM

Ending at: 11:59:59 PM

Duration

Start date: 4/30/2008

End date: 4/30/2008

No end date

Summary

Description: Occurs every week on Sunday at 12:00:00 AM. Schedule will be used starting on 4/30/2008.

When policy evaluation is scheduled, any violations are logged to the Windows Event Log.

The On Change - Log Only option evaluates the policy whenever the property in the facet is changed and any violation is logged to the Windows Event Log. The On Change - Prevent option evaluates the policy whenever the property in the facet is changed and actually prevents the change; this option uses DDL triggers to enforce the policy. Not all changes can be detected and rolled back by DDL triggers; the Execution Mode drop down list will include the On Change - Prevent option only when it is available.

One final note on the policy setup concerns the Enabled check box. When the Execution Mode is On Demand, the Enabled check box must be unchecked; for all other options you must check the Enabled check box in order for the policy to be evaluated.

Evaluate a Policy

To evaluate a policy on demand, right click on the policy in the SSMS Object Explorer and select Evaluate from the menu. The following is a partial screen shot of the output from evaluating a policy on demand:

Target details:			
	Server	Target	Details
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Database/demo	View...
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Database/ReportServer	View...
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Database/ReportServerTempDB	View...
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Database/sql2008_fs	View...
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Database/sql2008demo	View...

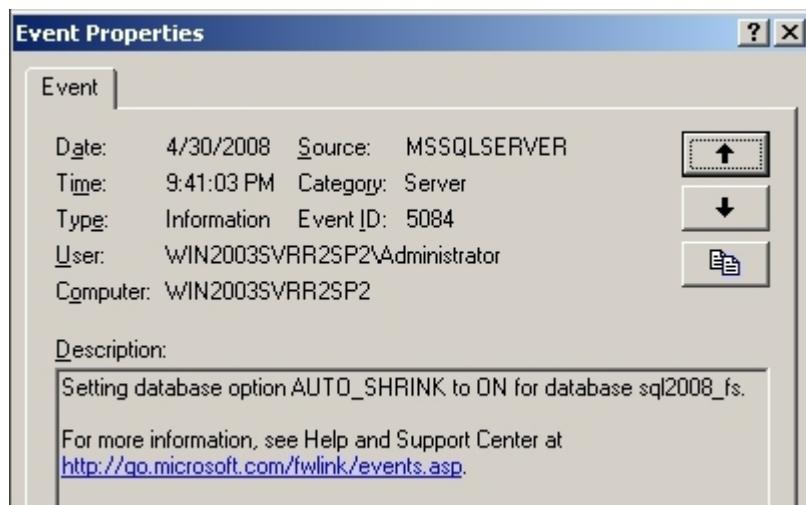
The green check icon signifies that the policy evaluated to True for the databases shown. Not shown above is a Configure button that allows the user to automatically fix a target where the policy evaluates to False.

Right click on a database in the SSMS Object Explorer and select Properties from the menu. Click the Options page and change the AutoShrink property to True. Evaluate the policy again and you will see the following output:

	Server	Target	Details	Message
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Data...	View...	
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Data...	View...	
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Data...	View...	
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Data...	View...	
	WIN2003VRR2SP2	/Server/WIN2003VRR2SP2/Data...	View...	

Note the red icon with the X indicating that policy evaluation failed for a particular database. Not shown above is the Configure button which you can click to automatically change the AutoShrink property to comply with the policy.

Edit the policy and change the Execution Mode to On Change - Log Only. Select a database and change the AutoShrink property to True. Open the Windows Event Viewer, click on Application and you will see an event that was written when the policy evaluation detected the violation:



Resource Governor:

Handling workloads has been quite difficult until SQL Server 2005. For example consider a scenario where one SQL instance is serving two applications i.e. an OLTP application and a reporting/data warehousing application. Since reporting applications are normally resource intensive, it may consume all the SQL Server available resources and may hamper the OLTP application which ideally should have more preference over the reporting application.

To handle this scenario, in earlier version of SQL Server there was one option to create multiple instances for these application (segregating by running one application on each instance) and setting CPU affinity mask for these instances appropriately. But the problems with this approach are, first it works for CPU only and second the dedicated CPUs cannot be shared by other SQL Server instances. For example, if there are two SQL Server instances and instance one has been assigned CPU 1 and 2 and instance two has been assigned CPU 3 and 4 on a four processor machine, even if instance one is idle and instance two is in need of additional resources, it can only use CPU 3 and 4. So what does SQL 2008 offer to solve this issue?

Resource Governor is a new technology in SQL Server 2008 that enables you to manage SQL Server workloads and resources by specifying limits on resource consumption by incoming requests. In an environment where multiple distinct workloads are present on the same server, Resource Governor enables us to differentiate these workloads and allocate shared resources as they are requested, based on the limits that you specify. These resources are CPU and memory. In other words, Resource Governor enables you to assign a relative importance to workloads. In other words, one workload can be allowed to proceed faster than another or is guaranteed to complete if there is resource contention. It allows a DBA or ITPros to define resource limits and priorities for different workloads.

Resource Governor Components

There are three new components of Resource Governor which are important to understand : resource pools, workload groups and classification (or classifier user-defined functions).

- **Pool:** A *resource pool*, or *pool*, is a collection of system resources such as memory or CPU; it represents a portion of the physical resources of the server. Depending on its settings, a pool may have a fixed size (its minimum and maximum resource usage settings are equal to each other) or have a part which is shared between multiple pools (its minimum is less than its effective maximum). "Shared" in this case simply means that resources go to the pool that requests the resources first. In the default configuration all resources are shared, thus maintaining backward compatibility with SQL Server 2005 policies. Two resource pools (*internal* and *default*) are created when SQL Server 2008 is installed. Resource Governor also supports 18 user-defined resource pools. You specify MIN and MAX values for resources (CPU or Memory) which represents the minimum guaranteed resource availability of the pool and the maximum size of the pool, respectively. The sum of MIN values across all pools cannot exceed 100 percent of the server resources. MAX value can be set anywhere in the range between MIN and 100 percent inclusive. The *internal* pool represents the resources consumed by the SQL Server itself. This pool always contains only the internal group, and the pool is not alterable in any way. Resource consumption by the internal pool is not restricted. Any workloads in the pool are considered critical for server function, and Resource Governor allows the internal pool to pressure other pools even if it means the violation of limits set for the other pools. The *default* pool is the first predefined user pool. Prior to any configuration the default pool only contains the default group. The default pool cannot be created or dropped but it can be altered. The default pool can contain user-defined groups in addition to the default group.
- **Group:** A *workload group*, or *group*, is a user-specified category of requests that are similar according to the classification rules that are applied to each session request. A group defines the policies for its members. A resource pool is assigned to a Workload Group, which in turn is assigned to the Resource Governor. Two workload groups (*internal* and *default*) are created and mapped to their corresponding resource pools when SQL Server 2008 is installed, apart from that the Resource Governor also supports user-defined workload groups. The *internal* workload group is populated with requests that are for internal SQL Server use only. You cannot change the criteria used for routing these requests and you cannot classify requests into the *internal* workload group whereas requests are mapped to *default* workload group, if there is a classification failure, an attempt to map to a non-existent workload group and there is no criteria to classify. If the Resource Governor is disabled, all new connections are automatically classified into the *default* group and System-initiated requests are classified into the *internal* workload group.
- **Classification:** *Classification* is a set of user-written rules that enable Resource Governor to classify session requests into the workload groups as described previously; for example classifying on the basis of user, application etc. It is implemented through a scalar Transact-SQL user-defined function (UDF) which is designated as a "classifier UDF" for the Resource Governor in the master database. Only one user-defined function can be designated as a classifier at a time.

Putting it all together

The incoming connection request for a session is classified by a classifier UDF and is routed to an appropriate workload group. This workload group in turn uses the resource pool associated with it and finally the resource pool provides and limits on the resources required by the session. Let's see this with an example, I will consider the same problem discussed in the problem section where we have one SQL Server instance serving an OLTP application and a reporting application, though it can be used in variety of different circumstances where you have to manage workloads:

- First I will create two resource pools to be used by OLTP and Reporting application, then I will create two workload groups which will categorize the request coming from these applications.

Working with Resource Governor in SQL Server 2008 - Part 1

```
--Resource pool to be used by OLTP Application
CREATE RESOURCE POOL OLTPPool
WITH
(
    MIN_CPU_PERCENT=50, MAX_CPU_PERCENT=100,
    MIN_MEMORY_PERCENT=50, MAX_MEMORY_PERCENT=100
)
GO

--Resource pool to be used by Report Application
CREATE RESOURCE POOL ReportPool
WITH
(
    MIN_CPU_PERCENT=50, MAX_CPU_PERCENT=100,
    MIN_MEMORY_PERCENT=50, MAX_MEMORY_PERCENT=100
)
GO

--Workload Group to be used by OLTP Application
CREATE WORKLOAD GROUP OLTPGroup
    USING OLTPPool ;
GO

--Workload Group to be used by Report Application
CREATE WORKLOAD GROUP ReportGroup
    USING ReportPool ;
GO
```

- Next I will create the classifier UDF to route incoming request to different workload groups and finally I will enable Resource Governor with ALTER RESOURCE GOVERNOR RECONFIGURE statement. Assumption here is, the OLTP application uses "OLTPUser" login whereas reporting application uses "ReportUser" login.

Working with Resource Governor in SQL Server 2008 - Part 2

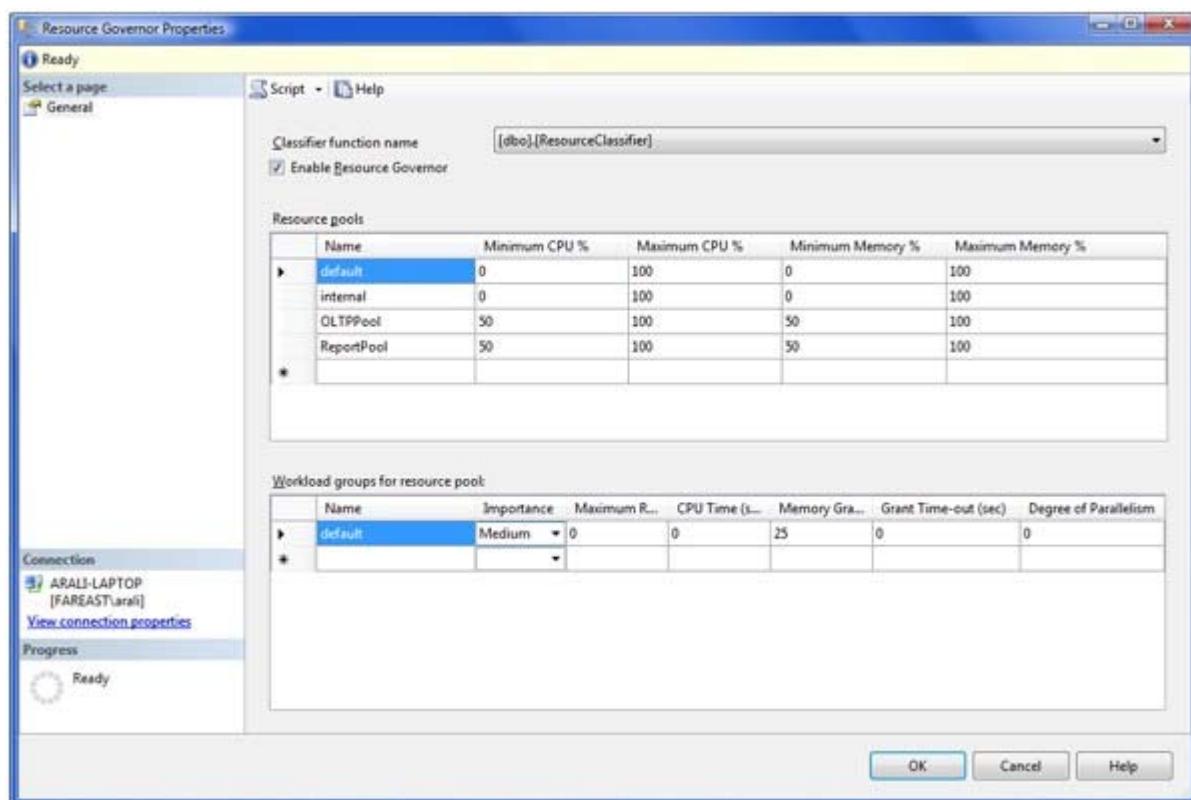
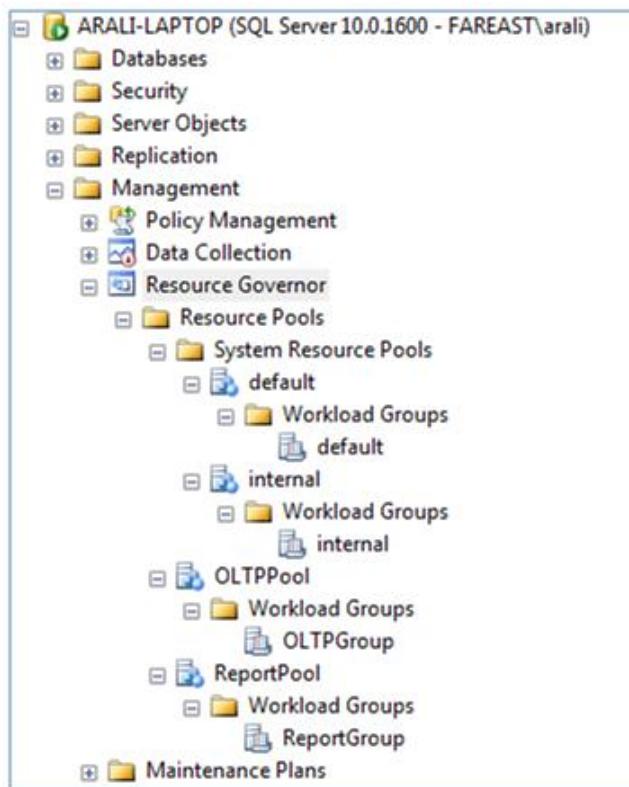
```
USE master;
GO
CREATE FUNCTION dbo.ResourceClassifier()
RETURNS SYSNAME
WITH SCHEMABINDING
AS
BEGIN
    --Declare the variable to hold the value returned in sysname.
    DECLARE @WorkloadGroup AS SYSNAME
```

```

--If the user login is 'OLTPUser', map the connection to the
--OLTPGroup workload group.
IF (SUSER_NAME() = 'OLTPUser')
    SET @WorkloadGroup = 'OLTPGroup'
--If the user login is 'ReportUser', map the connection to
--the ReportGroup workload group.
ELSE IF (SUSER_NAME() = 'ReportUser')
    SET @WorkloadGroup = 'ReportGroup'
ELSE
    SET @WorkloadGroup = 'default'
RETURN @WorkloadGroup
END
GO
--Register the classifier user-defined function and update the
--the in-memory configuration.
ALTER RESOURCE GOVERNOR
WITH (CLASSIFIER_FUNCTION=dbo.ResourceClassifier);
GO
--Enabling Resource Governor(By default when you install
--SQL Server, Resource Governor is disabled)
--It loads the stored configuration metadata into memory
ALTER RESOURCE GOVERNOR RECONFIGURE
GO
--Disabling Resource Governor
ALTER RESOURCE GOVERNOR DISABLE
GO
--It resets statistics on all workload groups and resource pools.
ALTER RESOURCE GOVERNOR RESET STATISTICS
GO

```

- Resource Governor can also be managed using SQL Server Management Studio (SSMS), you can CREATE, ALTER, DROP resource pools, workload groups, change classifier UDF and ENABLE/DISABLE Resource Governor as shown below.



Resource Governor's Catalog Views and Dynamic Management Views

There are three new Catalog Views and three new Dynamic Management Views introduced for Resource Governor.

- sys.resource_governor_configuration - used to display the Resource Governor configuration as stored in metadata.
- sys.resource_governor_resource_pools - used to display resource pool configuration as stored in metadata.
- sys.resource_governor_workload_groups - used to display workload group configuration as stored in metadata.
- sys.dm_resource_governor_configuration - used to get the current in-memory configuration state of Resource Governor
- sys.dm_resource_governor_resource_pools - used to get the current resource pool state, the current configuration of resource pools, and resource pool statistics.
- sys.dm_resource_governor_workload_groups - used to get the workload group statistics and the current in-memory configuration of the workload group.

In addition to new views that are specific to Resource Governor, existing system views have been modified to include information about Resource Governor as well.

Note

- Only one resource pool can be assigned to a workload, though a single resource pool can serve multiple workload groups. If there are multiple workload groups in a given resource pool, you can set relative importance of each workload group to either LOW, MEDIUM or HIGH.
- The resource governor limits can easily be reconfigured in real time with minimal impact on the workloads that are executing; for that purpose you use ALTER RESOURCE GOVERNOR statement with the RECONFIGURE parameter.
- To monitor utilization of Resource governor you can monitor different performance counters under *SQLServer:Resource Pool Stats* and *SQLServer:Workload Group Stats* performance counter categories.

Limitations

There are also some limitations to the Resource Governor. They are as follows:

- Resource management is limited to the SQL Server Database Engine. Resource Governor cannot be used for Analysis Services, Integration Services, and Reporting Services.
- Only a single instance can be managed through this. An organization may have more than a single instance, but must manage each separately.
- Limited to only two resources i.e. CPU bandwidth and memory management.
- You are allowed to create only 18 resource pools apart from *Default* and *Internal*/pool. Creating more resource pool than this throws an error, "The resource pool cannot be created. The maximum number of resource pools cannot exceed current limit of 20 including predefined resource pools." Though in one sense it is good to have fewer resource pools and assign multiple workloads to it if required.

- Resource Governor is available only on the Enterprise, Developer, and Evaluation editions of SQL Server.

Change Data Capture:

Change Data Capture (CDC) is a new feature in SQL Server 2008 which records insert, update and delete activity in SQL Server tables.

CDC is intended to capture insert, update and delete activity on a SQL table and place the information into a separate relational table. It uses an asynchronous capture mechanism that reads the transaction logs and populates the CDC table with the row's data which change. The CDC table mirrors the column structure of the tracked table, together with metadata regarding the change.

To use the CDC feature, first we have to enable it database level. You can use below query to retrieve the CDC enabled databases.

Steps to Enable the CDC on database level

```

7 USE master
8 GO
9 SELECT [name], database_id, is_cdc_enabled
10 FROM sys.databases WHERE is_cdc_enabled >> 0
11 GO

```

The screenshot shows a SQL query window with the results tab selected. The results show two databases: DBM_Test and dbcdc, both with is_cdc_enabled set to 1.

	name	database_id	is_cdc_enabled
1	DBM_Test	14	1
2	dbcdc	52	1

You can use below script to create the sample database and table

```
create database SQLDBPool
go
```

Sample DB and Table Creation Script

```
use sqldbpool
create table Employee
(
empID int constraint PK_Employee primary key Identity(1,1)
,empName varchar(20)
,salary int
)

insert into Employee values('Jugal','50000000'),('Abhinav',1000),('Sunil',2000)
```

To enable CDC on database SQLDBPool execute the below query.

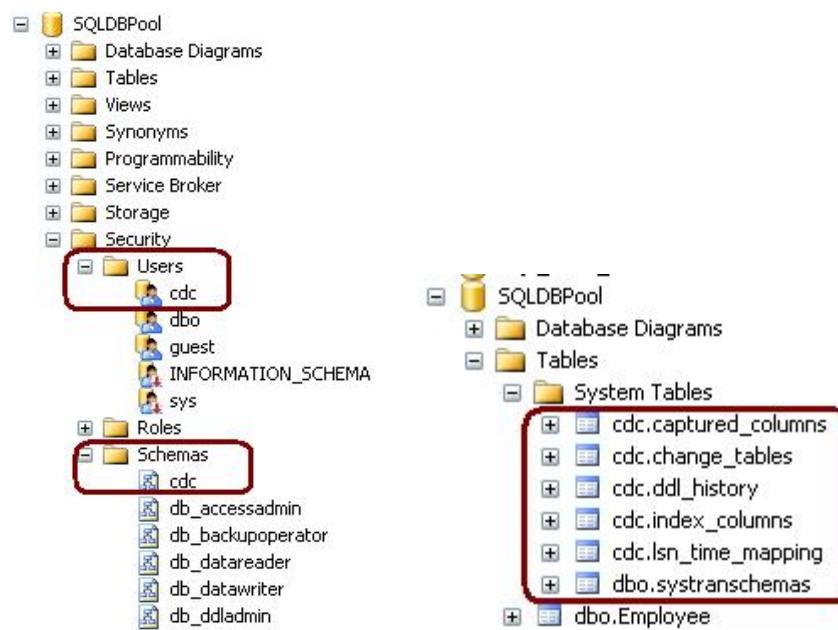
```
USE SQLDBPool
```

```
GO
```

```
EXEC sys.sp_cdc_enable_db
```

Once you have enabled the CDC for the database, you can see the CDC schema, CDC User and CDC tables in the database. Please see the below images for more information.

CDC Schema, CDC User and CDC system tables



cdc.captured_columns – Returns list of captured column

cdc.change_tables – Returns list of all the CDC enabled tables

cdc.ddl_history – Records history of all the DDL changes since capture data enabled

cdc.index_columns – Contains indexes associated with change table

cdc.lsn_time_mapping – Maps LSN number and time

Enable CDC on Table

As CDC feature can be applied at the table-level to any CDC enabled database. You can run below query to enable the CDC on the table.

Please note:

- You must have database owner permission (db_Owner fixed role)
- SQL Agent Service must be running

Using sys.sp_cdc_enable_table procedure we can enable the CDC at the table level. You can specify all the below different options as required.

@source_schema is the schema name of the table that you want to enable for CDC
@source_name is the table name that you want to enable for CDC
@role_name is a database role which will be used to determine whether a user can access the CDC data; the role will be created if it doesn't exist.
@supports_net_changes determines whether you can summarize multiple changes into a single change record; set to 1 to allow, 0 otherwise.
@capture_instance is a name that you assign to this particular CDC instance; you can have up two instances for a given table.
@index_name is the name of a unique index to use to identify rows in the source table; you can specify NULL if the source table has a primary key.
@captured_column_list is a comma-separated list of column names that you want to enable for CDC; you can specify NULL to enable all columns.
@filegroup_name allows you to specify the FILEGROUP to be used to store the CDC change tables.
@partition_switch allows you to specify whether the ALTER TABLE SWITCH PARTITION command is allowed

The screenshot shows a SQL Server Management Studio window. The left pane displays a script with numbered lines 37 through 44. Lines 37, 38, and 43 contain standard SQL commands: USE, GO, and another GO. Lines 39, 40, 41, and 42 call the sys.sp_cdc_enable_table stored procedure with parameters: @source_schema = N'dbo', @source_name = N'Employee', @role_name = NULL, and GO respectively. The right pane shows the 'Messages' tab with two lines of output: 'Job 'cdc.SQLDBPool_capture' started successfully.' and 'Job 'cdc.SQLDBPool_cleanup' started successfully.'

```
37 USE SQLDBPool
38 GO
39 EXEC sys.sp_cdc_enable_table
40 @source_schema = N'dbo',
41 @source_name    = N'Employee',
42 @role_name      = NULL
43 GO
44
```

Messages

```
Job 'cdc.SQLDBPool_capture' started successfully.
Job 'cdc.SQLDBPool_cleanup' started successfully.
```

cdc.SQLDBPool_capture – Capture the changes by doing log scan
cdc. SQLDBPool _cleanup –Clean Up the database changes tables.

Once the above query executes successfully, it will create 1 more system table cdc.dbo.Employee_CT for the tracking purpose.

See the result of the SELECT query on both the tables.

The screenshot shows a SQL query window with two queries:

```
45 select * from Employee
46 select * from cdc.dbo.Employee_CT
47
```

The second query, `select * from cdc.dbo.Employee_CT`, is highlighted with a red box. A red arrow points from this box to the results grid below. The results grid displays data from the `Employee` table and the corresponding columns from the `cdc.dbo.Employee_CT` table.

	empID	empName	salary	__\$start_lsn	__\$end_lsn	__\$seqval	__\$operation	__\$update_mask	empID	empName	salary
1	1	Jugal	50000000								
2	2	Jugal	50000000								
3	3	Abhinav	1000								
4	4	Sunil	2000								

Below 5 additional columns are available into `cdc.dbo.Employee_CT` table.

The screenshot shows the Object Explorer with the path `cdc.dbo.Employee_CT > Columns`. A red box highlights the following five columns:

- __\$start_lsn (binary(10), not null)
- __\$end_lsn (binary(10), null)
- __\$seqval (binary(10), not null)
- __\$operation (int, not null)
- __\$update_mask (varbinary(128), null)

Below these, the regular columns `empID`, `empName`, and `salary` are listed.

`__$operation` and `__$update_mask` are very important columns. `__$operation` table contains the value against the DML operations.

1 = Delete Statement

2 = Insert Statement

3 = Value before Update Statement

4 = Value after Update Statement

`__$update_mask` A bit mask with a bit corresponding to each captured column identified for the capture instance. This value has all defined bits set to 1 when `__$operation` = 1 or 2. When `__$operation` = 3 or 4, only those bits corresponding to columns that changed are set to 1.

Example

Execute the below query on the SQLDBPool database.

```

49 insert into Employee values('DJ','10000')
50 delete Employee where empName = 'DJ'
51 update Employee set salary = 10 where Empname = 'Sunil'
52
53 select * from Employee
54 select * from cdc.dbo_Employee_CT
55

```

Results

empID	empName	salary
1	Jugal	50000000
2	Jugal	50000000
3	Abhinav	1000
4	Sunil	10

cdc.dbo_Employee_CT

_start_lsn	_end_lsn	_seqval	_operation	_update_mask	empID	empName	salary
0x00000027000000B00004	NULL	0x00000027000000B00003	2	0x07	5	DJ	10000
0x00000027000000CF0005	NULL	0x00000027000000CF0002	1	0x07	5	DJ	10000
0x00000027000000E20004	NULL	0x00000027000000E20002	3	0x04	4	Sunil	2000
0x00000027000000E20004	NULL	0x00000027000000E20002	4	0x04	4	Sunil	10

The diagram illustrates the CDC log entries. It shows four log entries corresponding to the operations in the code. The first two entries (rows 1 and 2) represent an insert of 'DJ' with salary 10000. The third entry (row 3) represents a delete of 'DJ'. The fourth entry (row 4) represents an update of 'DJ' to Sunil with salary 10. Red arrows point from the log entries to their respective operations: an arrow from row 1 to 'Insert', an arrow from row 2 to 'Delete', an arrow from row 3 to 'Before Update', and an arrow from row 4 to 'After Update'.

You can get more information on the CDC configuration by executing sys.sp_cdc_help_change_data_capture stored procedure.

```

56 USE SQLDBPool
57 GO
58 EXEC sys.sp_cdc_help_change_data_capture
59

```

Results

source_schema	source_table	capture_instance	object_id	source_object_id	start_lsn	end_lsn	supports_net_changes	has_drop_pending
dbo	Employee	dbo_Employee	341576255	2105058535	0x0000002100000397003E	NULL	1	NULL
role_name	index_name	filegroup_name	create_date	index_column_list	captured_column_list			
NULL	PK_Employee	NULL	2011-06-01 11:32:59.380	[emplID]	[emplID], [empName], [salai			

You can disable the CDC either on the table level or the database level. Use below code to disable the CDC on table or database level.

Table Level:

```

exec sys.sp_cdc_disable_table
@source_schema = 'dbo',
@source_name = 'Employee',
@capture_instance = 'dbo_Employee'

```

Database Level:

```

use SQLDBPool;
go
sys.sp_cdc_disable_db

```

CleanUp Job

As we checked in the above example that CDC is capturing all the changes at the table level which create the disk space issue. To resolve disk space issue we have clean up job which run every 3 days interval by default. We can schedule it to run as per our requirement.

Compression Techniques in SQL Server:

In computer science, compression is the process of encoding information in fewer bits than an un-encoded representation would use. The compression techniques available in SQL Server can be broadly categorized into two types depending on the way they are architected – **Data Compression** and **Backup Compression**.

Data compression occurs at runtime, the data is stored in a compressed form to reduce the disk-space occupied by a database. On the other hand, backup compression occurs only at the time of a backup and uses a proprietary compression technique. Backup compression can be used on a database that has already undergone data compression, but the savings might not be significant.

Data Compression comes in two different forms:

- **Row-level Data Compression:** Row-level data compression is essentially turning fixed length data types into variable length data types, freeing up empty space. It also has the ability to ignore zero and null values, saving additional space. In turn, more rows can fit into a single data page.
- **Page-level Data Compression:** Page-level data compression starts with row-level data compression, then adds two additional compression features: prefix and dictionary compression. We will take a look at what this means a little later in this chapter. As you can imagine, page-level compression offers increased data compression over row-level compression alone.

Backup Compression comes in a single form:

- **Backup Compression:** Backup compression does not use row-level or page-level data compression. Instead, backup compression occurs only at the time of a backup, and it uses its own proprietary compression technique. Backup compression can be used when using, or not using, data compression, although using backup compression on a database that is already compressed using data compression may not offer additional benefits.

Though compression can provide significant space saving, it can also cause severe performance issues if misused. As the name suggests, Unicode Data Compression comes under Data Compression, and to be more specific it's a part of Row-level Compression.

Data compression can be performed using either SQL Server Management Studio (SSMS) or by using Transact-SQL. For this demo, we will take a look at how you can compress a table that uses a clustered index, using SSMS.

Let's say that we want to compress the Sales.Customer table (which has a clustered index) in the AdventureWorks database. The first step is to right-click on the table in SSMS, select "Storage," and then select "Manage Compression."

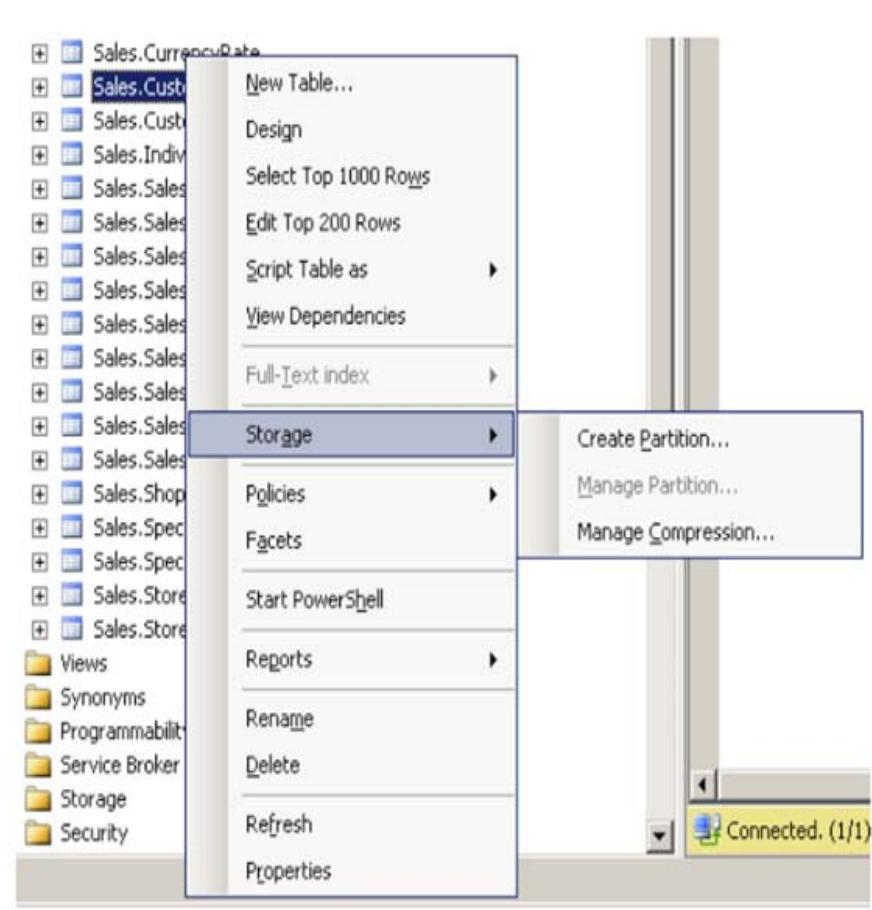


Figure 1: SSMS can be used to manage compression.

This brings up the Data Compression Wizard, displayed below.



Figure 2: The Data Compression Wizard, or Transact-SQL commands, can be used to manage data compression.

After clicking "Next," the wizard displays the following screen, which allows you not only to select the compression type, but it also allows you to calculate how much space you will save once compression has been turned on.

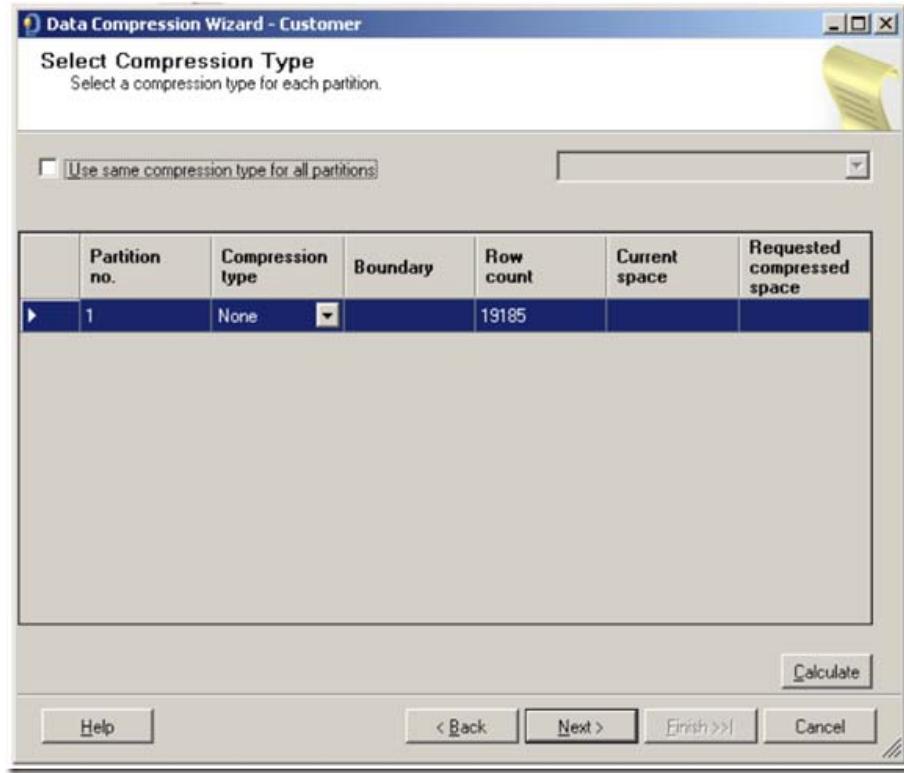


Figure 3: Use this screen to select the compression type, and to calculate how much space will be saved.

First, let's see how much space we will save if we use row-level compression on this table. To find out, click on the drop-down box below "Compression Type," select "Row," and then click "Calculate."

	Partition no.	Compression type	Boundary	Row count	Current space	Requested compressed space
▶	1	Row	▼	19185	1.953 MB	1.922 MB

Figure 4: For this table, row-level compression doesn't offer much compression.

After clicking "Calculate," the wizard runs and calculates how much space is currently being used, and how much space would be used after row-level compression. As we can see, very little space will be saved, about 1.6%.

Now, let's see how much compression savings page-level compression offers us for this particular table. Again, I go to the drop-down menu under "Compression Type," select "Page," then press "Calculate."

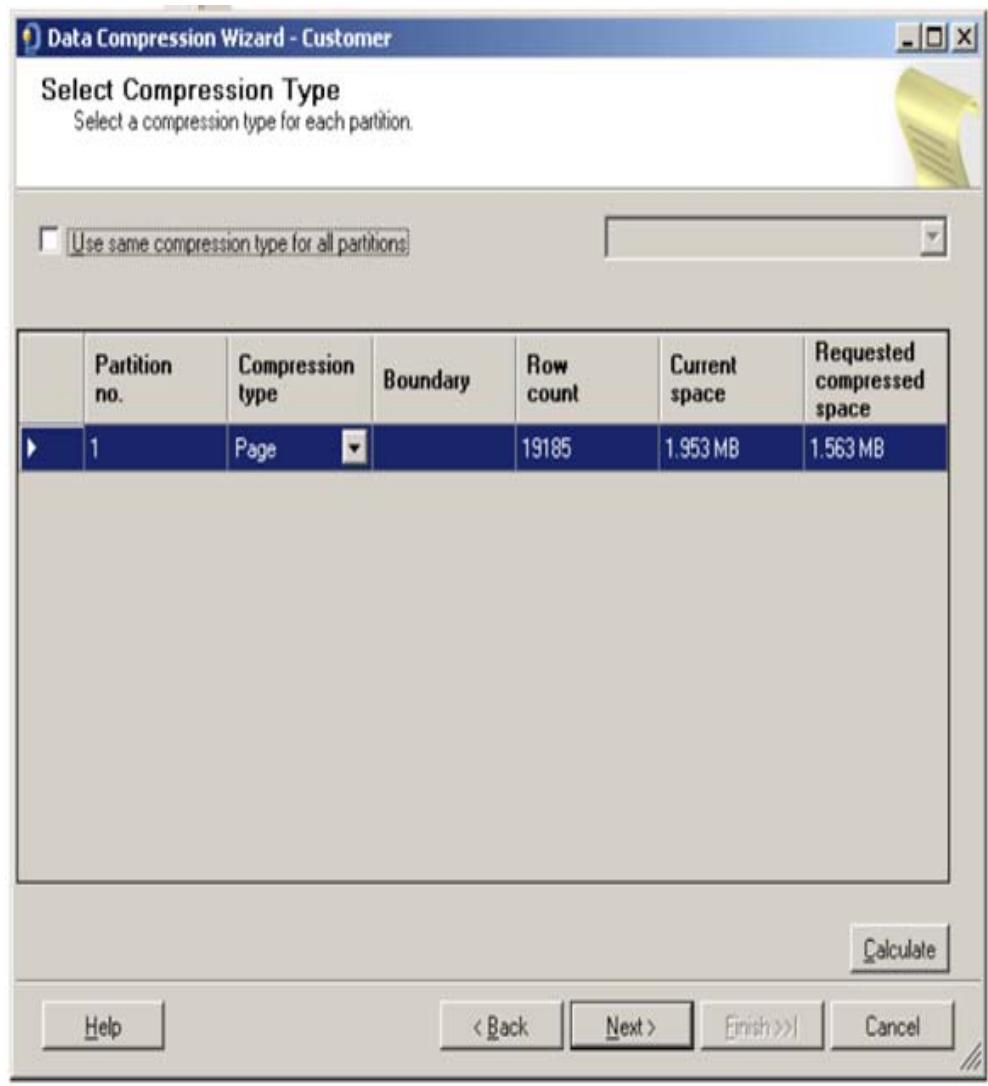


Figure 5: Page-level compression is higher than row-level compression.

After pressing "Calculate," we see that compression has improved greatly, now saving about 20% space. At this point, if you should decide to turn on page-level compression for this table, click on the "Next" button.

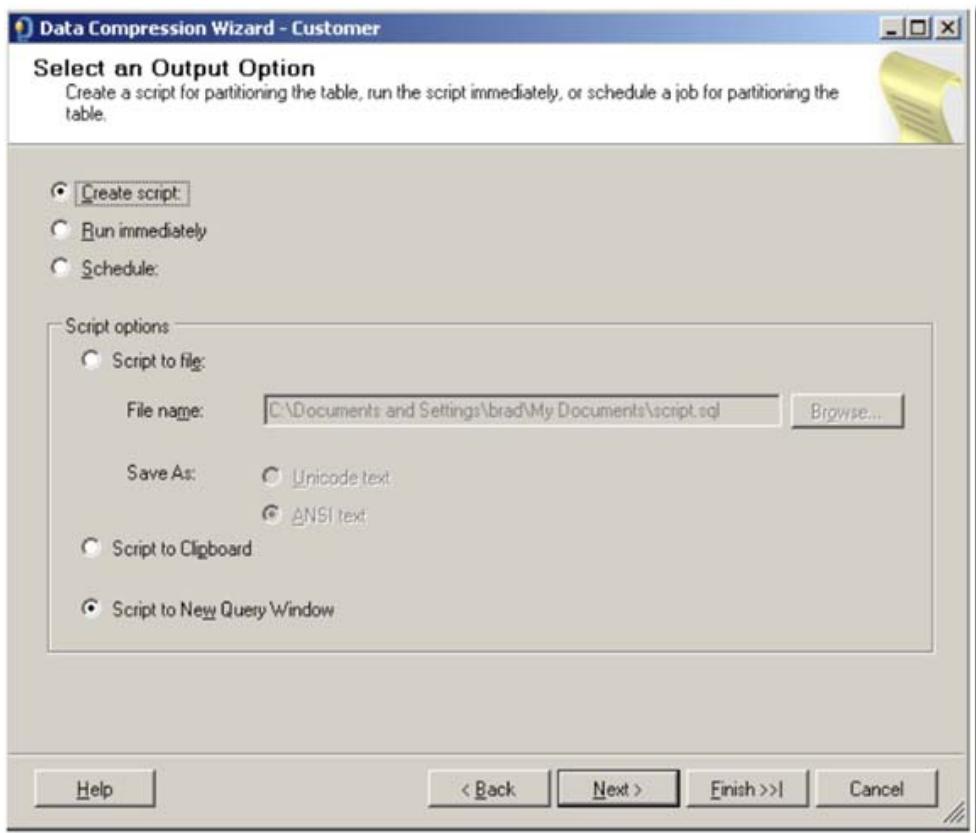


Figure 6: The wizard allows you several options in which to turn on compression.

At the above screen, you can choose to perform the compression now (not a good idea during busy production times because the initial compression process can be very CPU and disk I/O intensive), schedule it to run later, or just to script the Transact-SQL code so you can run it at your convenience.

Once you have compressed this table (a clustered index), keep in mind that any non-clustered indexes that this table may have are not automatically compressed for you. Remember, compression is based on a per object basis. If you want to compress the non-clustered indexes for this table, you will have to compress each one, one at a time.

While this wizard helps you to see how much compression either method offers, it does not suggest which compression method should be used, nor does it recommend whether compression should be used at all for this object. As the DBA, it will be your job to evaluate each compressible object to determine if it should have compression enabled or not. In other words, you must decide if the benefits of compression outweigh the negatives.

Backup Compression

For years, there have been third-party programs that allow you to compress and speed up SQL Server backups. In most regards, the backup compression included with the Enterprise Edition of SQL Server is very plain vanilla. In fact, if you already are using a third-party backup program, I would suggest you

continue using it, because SQL Server 2008 backup compression offers fewer features. In fact, the only option SQL Server 2008 backup compression offers you is to turn it off or on. That's it.

SQL Server 2008 backup compression, like the third-party add-ons, compresses backups, which not only saves backup space, but it can substantially reduce backup times. Unlike data compression, there is very little downside to using backup compression, other than the additional CPU resources required to perform the compression (or decompression during a restore). Assuming that you perform backups during slower times of the day, the additional CPU resources used will not be noticeable.

The time and space savings offered by backup compression depends on the data in your database. If you are heavily using data compression in your databases, or are using Transparent Data Encryption, then using backup compression probably won't offer you many benefits, as already compressed data, or encrypted data, is not very compressible.

Let's take a brief look at how you turn on SQL Server 2008 backup compression. While our example will use SSMS, you can use Transact-SQL to perform the same task. To backup AdventureWorks, right-click on the database in SSMS, select "Tasks," and then select "Back Up," and the backup dialog box appears.

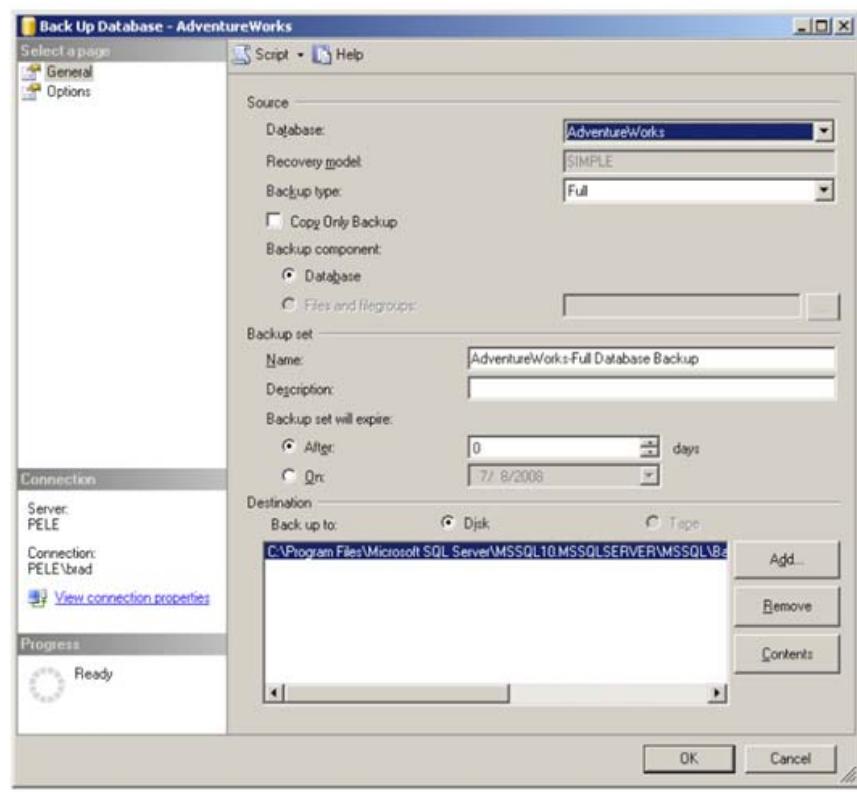


Figure 7: As with any backup, you can use the backup dialog box to make your selections.

Once you have selected your backup options, next click on "Options," and the following screen appears.

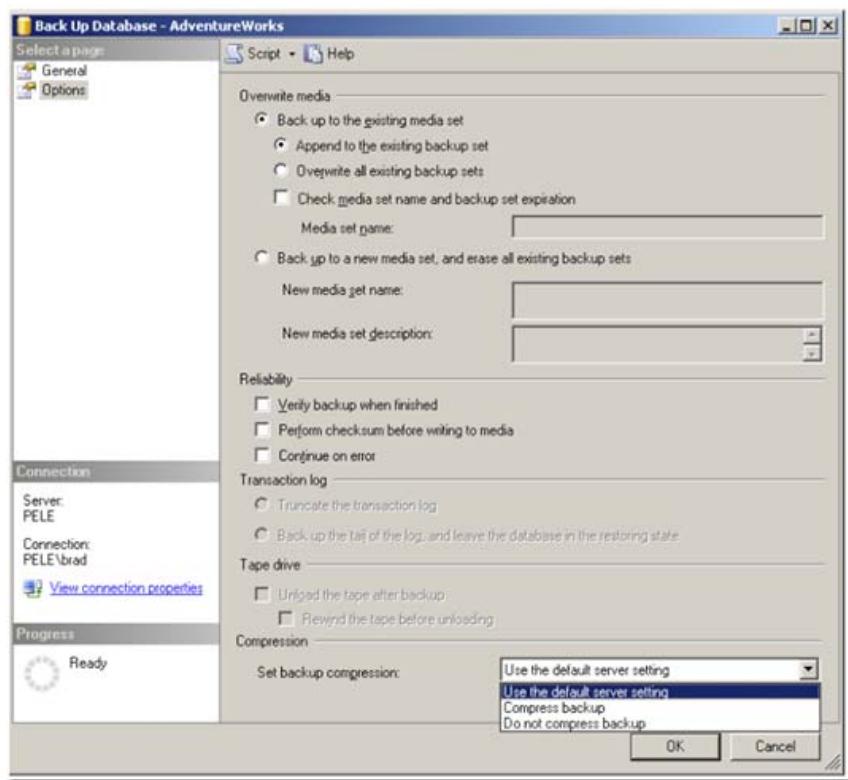


Figure 8: Backup compression options are limited.

At the top of figure 8 are the standard backup options, while at the bottom of the screen you see the options for backup compression. Notice that you only have three choices.

The first option, "Use the default server settings" tells the backup to use the server's default backup compression setting. In SQL Server 2008, there is a new sp_configure option called "backup compression default." By default, it is set to have backup compression off. If you want, you can set this option so that backup compression is turned on by default. So if you choose the "Use the default server settings" option above, then whatever option is set for the "backup compression default" will be used for the backup.

The "Compress Backup" option turns backup compression on, and the "Do not compress backup" option turns it off. Both of these options override the "backup compress default" server setting, whatever it happens to be.

Once you have chosen your backup compression method, you proceed with the backup just like any other SQL Server backup. If you need to restore a compressed backup, you don't have to do anything special, it will uncompress itself automatically. Although you can only compress backups using the Enterprise Edition of SQL Server 2008, you can restore a compressed backup to any edition of SQL Server 2008. On the other hand, you cannot restore a compressed SQL Server 2008 backup to any previous version of SQL Server.

Partitioning:

Why Use Both Partitioned Tables and Indexes?

Partitioned tables are a way to spread a single table over multiple partitions, and while doing so each partition can be on a separate filegroup. Following are several reasons for doing this:

Faster and easier data loading: If your database has a large amount of data to load, you might want to consider using a partitioned table. "A large amount of data," doesn't mean a specific amount of data, but any case in which the load operation takes longer than is acceptable in the production cycle. A partitioned table enables you to load the data to an empty table that's not in use by the "live" data, so it has less impact on concurrent live operations. Clearly, there will be an impact on the I/O subsystem, but if you also have separate filegroups on different physical disks, even this has a minimal impact on overall system performance. After the data is loaded to the new table, you can perform a switch to add the new table to the live data. This switch is a simple metadata change that quickly executes, which is why partitioned tables are a great way to load large amounts of data with limited impact to users who touch the rest of the data in the table.

Faster and easier data deletion or archival: For the same reasons, partitioned tables also help you to delete or archive data. If your data is partitioned on boundaries that are also the natural boundaries on which you add or remove data, the data is considered to be aligned. When your data is aligned, deleting or archiving data is as simple as switching a table out of the current partition, after which you can unload or archive it at your leisure. There is a bit of a catch to this part: With archiving, you often want to move the old data to slower or different storage. The switch operation is so fast because all it does is change metadata. It doesn't move any data around, so to actually move the data from the filegroup where it lived to the old, slow disk archival filegroup, you need to move the data, but you move it when the partition isn't attached to the existing partitioned table. Therefore, although this may take quite some time, it can have a minimal impact on any queries executing against the live data.

Faster queries: You are probably interested in an opportunity to get faster queries. When querying a partitioned table, the query optimizer can eliminate searching through partitions that it knows won't hold any results. This is referred to as partition elimination. This works only if the data in the partitioned table or index is aligned with the query. That is, the data must be distributed among the partitions in a way that matches the search clause on the query. You learn more details about this as you consider how to create a partitioned table. SQL Server 2008 offers some improvements for parallel query processing enhancements on partitioned tables and indexes.

Sliding windows: A sliding window is basically what was referred to earlier in the discussion about adding new data and then deleting or archiving old data. What you did was fill a new table, switch it into the live table, and then switch an existing partition out of the live table for archival or deletion. It's kind of like sliding a window of new data into the current partitioned table, and then sliding an old window of data out of the partitioned table.

Creating Partitioned Tables

Table partitioning requires SQL Server 2012 Enterprise Edition. There are also some expectations about the hardware in use, in particular the storage system; although these are implicit expectations, and you

can store the data anywhere you want. You just won't get the same performance benefits you would get if you had a larger enterprise storage system with multiple disk groups dedicated to different partitions.

SQL Server 2012 supports up to 15,000 partitions by default and is fully supported in 64-bit systems. In 32-bit systems, it is possible to create more than 1,000 table or index partitions, but it is not fully supported.

To create a partitioned table or index, perform the following steps:

1. Specify how the table or index is partitioned by the partitioning column, and the range of values included for each partition. Only one partitioning column can be specified. For example, to create four partitions based on a DateKey column, you execute the following command:

```
CREATE PARTITION FUNCTION DateKeyRange_PF (int)
```

```
AS RANGE LEFT FOR VALUES (20021231, 20031231, 20041231);
```

Either LEFT or RIGHT boundaries can be specified in the partition function. If no partition boundary is specified LEFT is used as default.

PARTITION NO.	DESCRIPTION
1	All records with DateKey <= 20021231
2	Records between Datekey>20021231 and Datekey<=20031231
3	Records between Datekey>20031231 and Datekey<=20041231
4	All records with DateKey > 20041231

2. To determine the partition number where a record will be placed based on the DateKey column value, use the \$PARTITION function as follows:

```
SELECT '20010601' DateKey, $PARTITION.DateKeyRange_PF(20010601) PartitionNumber
```

```
UNION
```

```
SELECT '20030601' DateKey, $PARTITION.DateKeyRange_PF(20030601) PartitionNumber
```

```
UNION
```

```
SELECT '20040601' DateKey, $PARTITION.DateKeyRange_PF(20040601) PartitionNumber
```

```
UNION
```

```
SELECT '20050601' DateKey, $PARTITION.DateKeyRange_PF(20050601)
```

```

CREATE PARTITION FUNCTION DateKeyRange_PF(int)
AS RANGE LEFT FOR VALUES (20021231, 20031231, 20041231);

SELECT '20010601' DateKey, $PARTITION.DateKeyRange_PF(20010601) PartitionNumber
UNION
SELECT '20030601' DateKey, $PARTITION.DateKeyRange_PF(20030601) PartitionNumber
UNION
SELECT '20040601' DateKey, $PARTITION.DateKeyRange_PF(20040601) PartitionNumber
UNION
SELECT '20050601' DateKey, $PARTITION.DateKeyRange_PF(20050601) PartitionNumber

```

The screenshot shows a SQL Server Management Studio window. The top pane contains T-SQL code for creating a partition function named DateKeyRange_PF. The bottom pane shows the results of a query that selects DateKey and its corresponding PartitionNumber based on the partition function. The results are displayed in a table with columns DateKey and PartitionNumber, containing four rows with values 20010601, 20030601, 20040601, and 20050601 respectively.

	DateKey	PartitionNumber
1	20010601	1
2	20030601	2
3	20040601	3
4	20050601	4

3. Now create a partition scheme. For example, create a partition scheme with four filegroups that can be used to hold the four partitions defined in the DateKeyRange_PF partition function as follows:

```

CREATE PARTITION SCHEME DateKeyRange_ps
AS PARTITION DateKeyRange_PF
TO (FileGroup1, FileGroup2, FileGroup3, FileGroup4, FileGroup5)

```

PARTITIONED TABLE

After creation of a partition scheme, a table may be defined to follow that scheme. In this case the table is called PARTITIONED. A partitioned table may have a partitioned index. Partition aligned index views may also be created for this table. These index and view may be based on different partition strategy (partition function and partition scheme).

There may be question in your mind if it is possible to partition your table using multiple columns. The answer may be YES or NO. Why? No, because there is no such direct support for this in SQL Server. Yes, because you can still do that by using persisted computed column based on any number of columns you want. It must be remembered that this is still a single dimension partition.

Now you might be wondering whether your existing tables could be partitioned or not. For partitioning your existing table just drop the clustered index on your table and recreate it on the required partition scheme.

```
CREATE TABLE
[ database_name . [ schema_name ] . | schema_name . ] table_name
( { <column_definition> | <computed_column_definition> }
[ <table_constraint> ] [ ,...n ] )
[ ON { partition_scheme_name ( partition_column_name ) | filegroup
| "default" } ]
[ { TEXTIMAGE_ON { filegroup | "default" } } [ ; ]
```

SQL Server 2008 has introduced an exciting new feature, 'FileStream'. Remember if you have any column specified which is based on *filestream* data, you have to follow some additional steps to get things to work with partitioning.

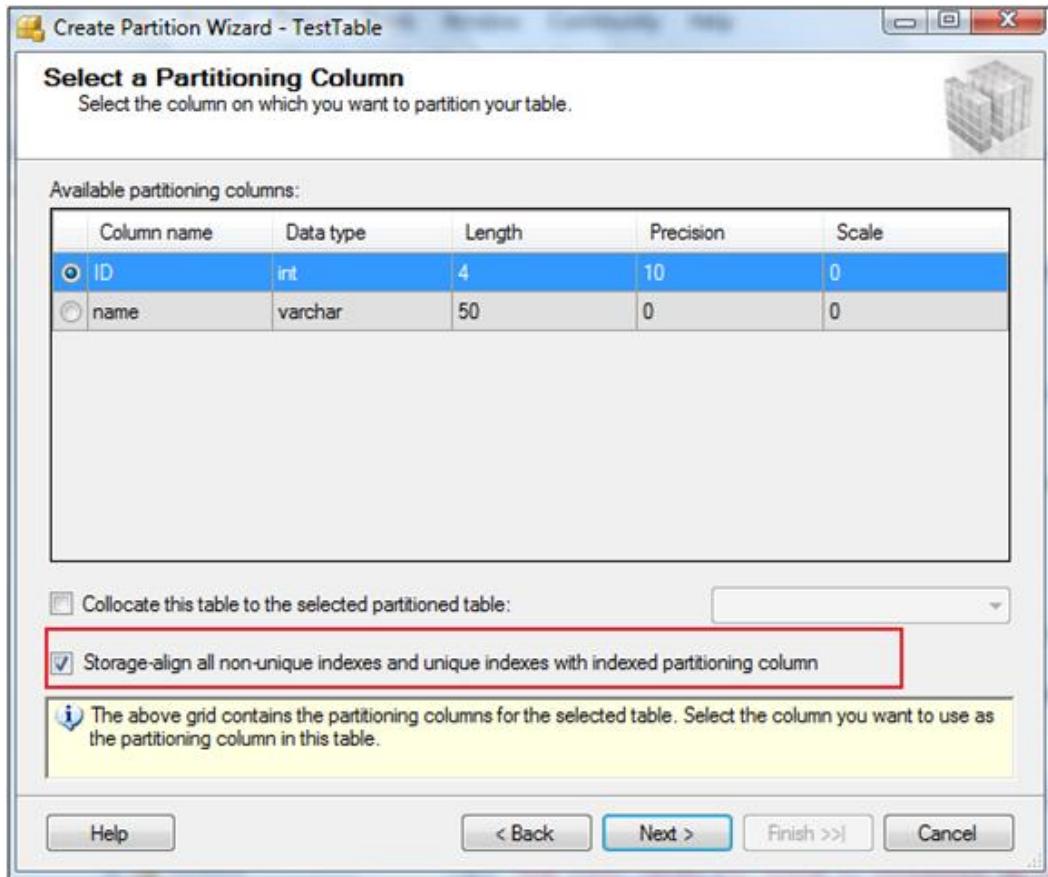
PARTITIONED INDEX

We can also partition our indexes, and this should contribute to improved performance. If indexes are partitioned they serve as the local indexes for each partition. We can also go with Global indexes on a partitioned table without caring about the different partitions of the table.

It must be remembered that indexes can be partitioned using a different partition key than the table. The index can also have different numbers of partitions than the table. We cannot, however, partition the clustered index differently from the table. To partition an index, ON clause is used, specifying the partition scheme along with the column when creating the index:

```
CREATE NONCLUSTERED INDEX MyNonClusteredIndex ON dbo.MyTable(MyID1)
ON MyPartitionScheme(MyID2);
```

You can see that Mytable is indexed on the MyID1 column, but the index is partitioned on the MyID2 column. If your table and index use the same Partition function then they are called Aligned. If they go further and also use the same partition scheme as well, then they are called Storage Aligned (note this in the figure below). If you use the same partition function for partitioning index as used by the table, then generally performance is improved.



PARTITION ALIGNED INDEX VIEWS

Partition aligned index views allow to efficiently create and manage summary aggregates in relational data. The query results are materialized immediately and persisted in physical storage in the database. This is an extension of Indexed views which existed in SQL Server 2005. Earlier, it was difficult to manage index views on partitioned table. This is because switching in and out the partition was not possible as the data was not distributed in a partitioned way in the indexed view. Indexed views were required to be dropped before this operation. In SQL Server 2008, this became possible with the introduction of Partition Aligned Index Views. In this way Indexed views have now evolved to become 'Partition Aware'.

It is said that these types of index views increase the speed and efficiency of queries on the partitioned data. The following conditions should be true if an index view has to be partition aligned with the table on which this view is defined.

- The partition functions of the indexes of the indexed view and table must define the same number of partitions, their boundary values and the partition must be based on the same column.

- The projection list of the view definition includes the partitioning column (as opposed to an expression that includes the partitioning column) of the partitioned table.
- Where the view definition performs a grouping, the partitioning column is one of the grouping columns included in the view definition.
- Where the view references several tables (using joins, sub queries, functions, and so on), the indexed view is partition-aligned with only one of the partitioned tables.

These views can be local or distributed. The local partitioned index view is the one in which all partitions lie on the same SQL Server instance. For a distributed one, different partitions of tables, queried in single view, reside on different SQL Server instances across the network. The distributed partitioned views are specially used to support federation of SQL Server instances.

Case study on Partitioning

Step 1 : Create New Test Database with two different filegroups

I have written tutorial using my C: Drive, however to take advantage of partition it is recommended that different file groups are created on separate hard disk to get maximum performance advantage of partitioning. Before running following script, make sure C: drive contains two folders – Primary and Secondary as following example has used those two folder to store different filegroups.

```
--- Step 1 : Create New Test Database with two different filegroups.
```

```
CREATE DATABASE TestDB
ON PRIMARY
(NAME='TestDB_Part1',
FILENAME=
'C:\Data\Primary\TestDB_Part1.mdf',
SIZE=3,
MAXSIZE=100,
FILEGROWTH=1 ),
FILEGROUP TestDB_Part2
(NAME = 'TestDB_Part2',
FILENAME =
'C:\Data\Secondary\TestDB_Part2.ndf',
SIZE = 3,
MAXSIZE=100,
FILEGROWTH=1 );
GO
```

Step 2 : Create Partition Range Function

Partition Function defines the range of values to be stored on different partition. For our example let us assume that first 10 records are stored in one filegroup and rest are stored in different filegroup. Following function will create partition function with range specified.

```
--- Step 2 : Create Partition Range Function
CREATE PARTITION FUNCTION TestDB_PartitionRange (INT)
```

```
AS RANGE LEFT FOR  
VALUES (10);  
GO
```

Step 3 : Attach Partition Scheme to FileGroups

Partition function has to be attached with filegroups to be used in table partitioning. In following example partition is created on primary and secondary filegroup.

```
USE TestDB;  
GO  
--- Step 3 : Attach Partition Scheme to FileGroups  
CREATE PARTITION SCHEME TestDB_PartitionScheme  
AS PARTITION TestDB_PartitionRange  
TO ([PRIMARY], TestDB_Part2);  
GO
```

Step 4 : Create Table with Partition Key and Partition Scheme

The table which is to be partitioned has to be created specifying column name to be used with partition scheme to partition tables in different filegroups. Following example demonstrates ID column as the Partition Key.

```
USE TestDB;  
GO  
--- Step 4 : Create Table with Partition Key and Partition Scheme  
CREATE TABLE TestTable  
(ID INT NOT NULL,  
Date DATETIME)  
ON TestDB_PartitionScheme (ID);  
GO
```

Step 5 : (Optional/Recommended) Create Index on Partitioned Table

This step is optional but highly recommended. Following example demonstrates the creation of table aligned index. Here index is created using same Partition Scheme and Partition Key as Partitioned Table.

```
USE TestDB;  
GO  
--- Step 5 : (Optional/Recommended) Create Index on Partitioned Table  
CREATE UNIQUE CLUSTERED INDEX IX_TestTable  
ON TestTable(ID)  
ON TestDB_PartitionScheme (ID);  
GO
```

Step 6 : Insert Data in Partitioned Table

Insert data in the partition table. Here we are inserting total of 3 records. We have decided that in table partition 1 Partition Key ID will contain records from 1 to 10 and partition 2 will contain rest of the records. In following example record with ID equals to 1 will be inserted in partition 1 and rest will be inserted in partition 2.

```
USE TestDB;  
GO  
--- Step 6 : Insert Data in Partitioned Table  
INSERT INTO TestTable (ID, Date) -- Inserted in Partition 1
```

```

VALUES (1,GETDATE());
INSERT INTO TestTable (ID, Date) -- Inserted in Partition 2
VALUES (11,GETDATE());
INSERT INTO TestTable (ID, Date) -- Inserted in Partition 2
VALUES (12,GETDATE());
GO

```

Step 7 : Test Data from TestTable

Query TestTable and see the values inserted in TestTable.

```

USE TestDB;
GO
--- Step 7 : Test Data from TestTable
SELECT *
FROM TestTable;
GO

```

Step 8 : Verify Rows Inserted in Partitions

We can query sys.partitions view and verify that TestTable contains two partitions and as per Step 6 one record is inserted in partition 1 and two records are inserted in partition 2.

```

USE TestDB;
GO
--- Step 8 : Verify Rows Inserted in Partitions
SELECT *
FROM sys.partitions
WHERE OBJECT_NAME(OBJECT_ID)='TestTable';
GO

```

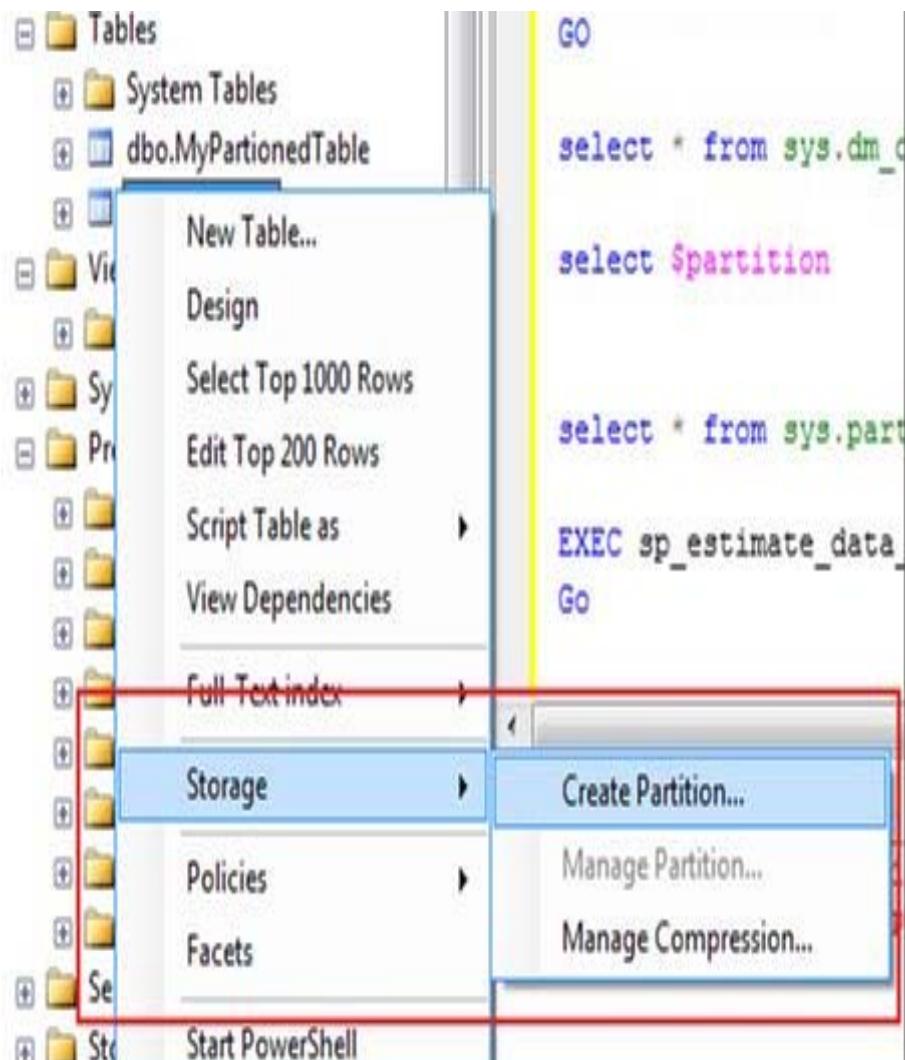
The screenshot shows the SQL Server Management Studio interface. The top tab bar has 'SQLAUTHORITY....\partition.sql' selected. Below it, the 'Object Explorer Details' pane is visible. The main query window displays the T-SQL code for verifying partitions. The 'Results' tab is selected at the bottom, showing a table with the following data:

	partition_id	object_id	index_id	partition_number	hobt_id	rows
1	72057594038452224	2073058421	1	1	72057594038452224	1
2	72057594038517760	2073058421	1	2	72057594038517760	2

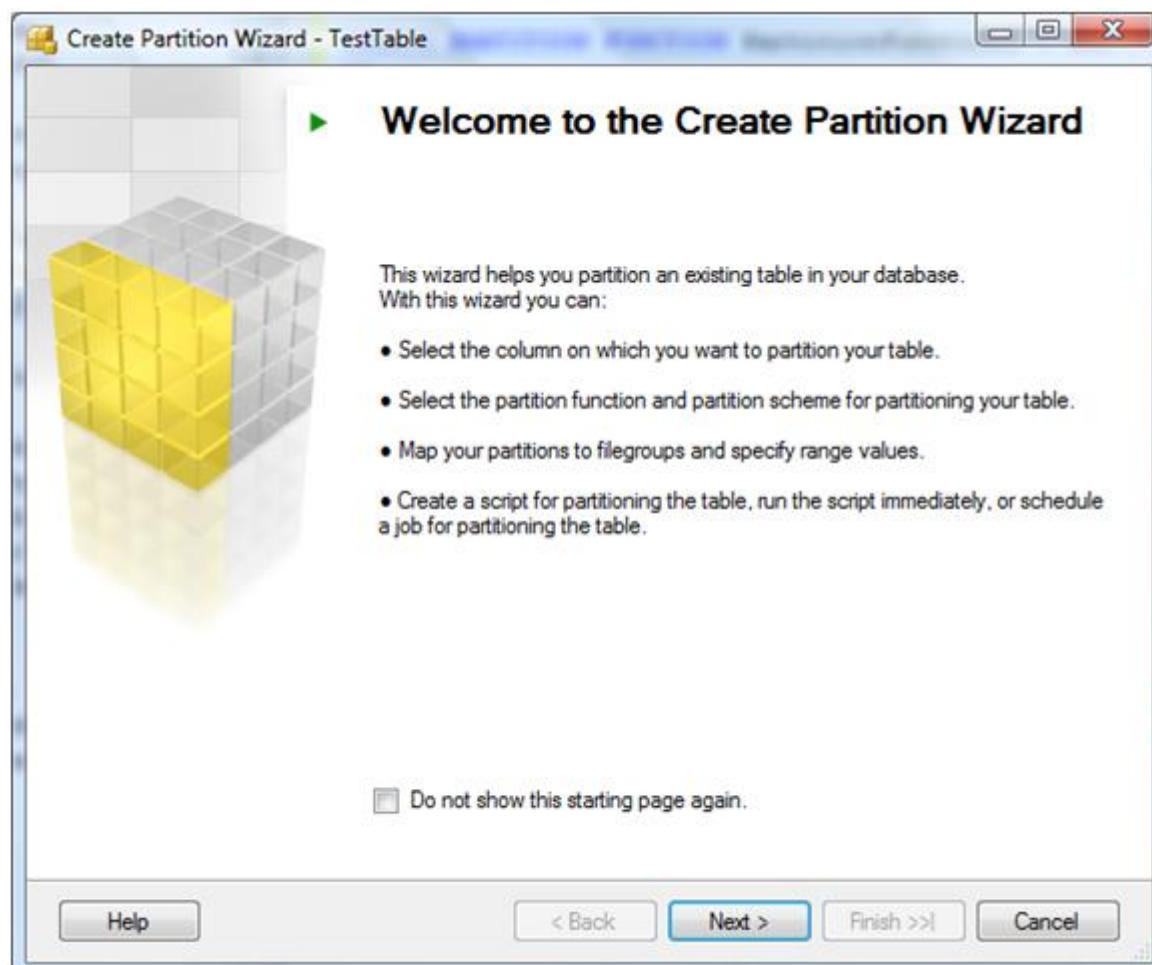
CREATE PARTITIONS

After the creation of a non-partitioned table, it can then be partitioned. To ensure the ease of partitioning, GUI support is available in SQL Server Management Studio. Not only can you create partitions in a non-partitioned table but you can also manage partitions in an already partitioned table.

Just right click a non-partitioned table and look at the following pop-up menu i.e. the Create Partition option under Storage menu:



When you select this option, an easy to use wizard is started to allow you to create partitions. Remember, you can also use already existing partition functions and scheme to partition a table. You can also create new ones and the wizard allows you to enter the details for the partition function, it allows the LEFT or RIGHT options, and for entering the details of the partition scheme. It also asks for the details of file groups for new partitions.



This wizard is very easy to use and allows the developer to generate partition functions, partition schemes and other options if none of the already existing ones fits the user's needs. There are also some third party solutions available to manage partitioning of your objects outside SQL Server Management Studio.

ADDITIONAL OPERATIONS WITH PARTITIONS

SQL Server also supports other operations with partitions. These operations help in managing a partitioned object. They are as follows:

1. Split
2. Merge
3. Switch

Remember that these operations are meta data operations and do not involve any movement of data.

ADDING A NEW PARTITION

This operation is supported to accommodate the continuous changes when a new Partition needs to be added if the already existing partitions are very large. This is technically called a Partition Split. ALTER TABLE statements support partition split with the required options. To split a partition, the ALTER PARTITION FUNCTION statement is used.

```
ALTER PARTITION FUNCTION MyPartitionFunction()
SPLIT RANGE (5000)
```

But before splitting, a new file group must be created and added to the partition scheme using the partition function being modified, otherwise, this statement would cause an error while executing.

REMOVING AN EXISTING PARTITION

The Merge operation is used to remove an existing partition. The syntax of MERGE statement is nearly the same as the SPLIT statement. This statement would remove the partition created in SPLIT command shown before this.

```
ALTER PARTITION FUNCTION MyPartitionFunction ()
MERGE RANGE (5000)
```

SWITCHING A PARTITION

This operation is used to switch a partition in or out of a partitioned table. It must be remembered that for switching in, the already existing partition must be empty within the destination table. In the following example, we are switching in a table MyNewPartition as 4th partition to MyPartitionedTable.

```
ALTER TABLE MyNewPartTable switch TO MyPartitionedTable PARTITION 4
```

Chapter – 23

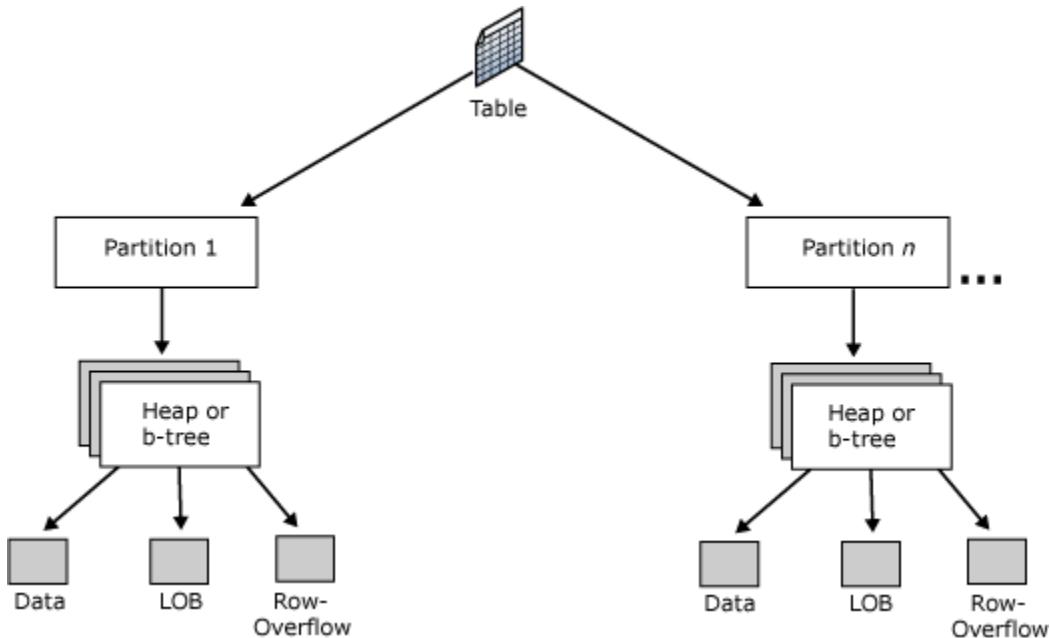
Indexing

Table & Index Architecture

Indexes are the other significant user-defined, on-disk data structure besides tables. An index provides fast access to data when the data can be searched by the value that is the index key. To really understand the benefit that indexes can provide and how to determine the best indexes for your environment, we need to know into the organization of Microsoft SQL Server indexes. Tables and indexes are stored as a collection of 8-KB pages.

Table Organization

The following illustration shows the organization of a table. A table is contained in one or more partitions and each partition contains data rows in either a heap or a clustered index structure. The pages of the heap or clustered index are managed in one or more allocation units, depending on the column types in the data rows.



Partitions

Table and index pages are contained in one or more partitions. A partition is a user-defined unit of data organization. By default, a table or index has only one partition that contains all the table or index pages. The partition resides in a single filegroup. A table or index with a single partition is equivalent to the organizational structure of tables and indexes in earlier versions of SQL Server.

When a table or index uses multiple partitions, the data is partitioned horizontally so that groups of rows are mapped into individual partitions, based on a specified column. The partitions can be put on one or more filegroups in the database. The table or index is treated as a single logical entity when queries or updates are performed on the data. To view the partitions used by a table or index, use the sys.partitions (Transact-SQL) catalog view.

Clustered Tables, Heaps, and Indexes

SQL Server tables use one of two methods to organize their data pages within a partition:

- Clustered tables are tables that have a clustered index. The data rows are stored in order based on the clustered index key. The clustered index is implemented as a B-tree index structure that supports fast retrieval of the rows, based on their clustered index key values. The pages in each level of the index, including the data pages in the leaf level, are linked in a doubly-linked list. However, navigation from one level to another is performed by using key values.
- Heaps are tables that have no clustered index. The data rows are not stored in any particular order, and there is no particular order to the sequence of the data pages. The data pages are not linked in a linked list. For more information, see Heap Structures.

Indexed views have the same storage structure as clustered tables.

When a heap or a clustered table has multiple partitions, each partition has a heap or B-tree structure that contains the group of rows for that specific partition. For example, if a clustered table has four partitions, there are four B-trees; one in each partition.

Nonclustered Indexes

Nonclustered indexes have a B-tree index structure similar to the one in clustered indexes. The difference is that nonclustered indexes do not affect the order of the data rows. The leaf level contains index rows. Each index row contains the nonclustered key value, a row locator and any included, or nonkey, columns. The locator points to the data row that has the key value.

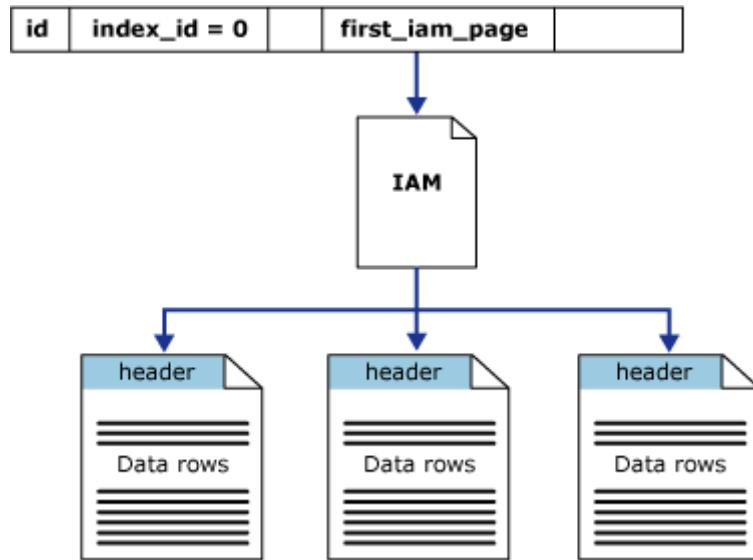
Heap Structures

A heap is a table without a clustered index. Heaps have one row in sys.partitions, with **index_id** = 0 for each partition used by the heap. By default, a heap has a single partition. When a heap has multiple partitions, each partition has a heap structure that contains the data for that specific partition. For example, if a heap has four partitions, there are four heap structures; one in each partition.

The column **first_iam_page** in the **sys.system_internals_allocation_units** system view points to the first IAM page in the chain of IAM pages that manage the space allocated to the heap in a specific partition. SQL Server uses the IAM pages to move through the heap. The data pages and the rows within them are not in any specific order and are not linked. The only logical connection between data pages is the information recorded in the IAM pages

Table scans or serial reads of a heap can be performed by scanning the IAM pages to find the extents that are holding pages for the heap. Because the IAM represents extents in the same order that they exist in the data files, this means that serial heap scans progress sequentially

through each file. Using the IAM pages to set the scan sequence also means that rows from the heap are not typically returned in the order in which they were inserted.



The following figure shows how the SQL Server Database Engine uses IAM pages to retrieve data rows in a single partition heap.

Clustered Index Structures

In SQL Server, indexes are organized as B-trees. Each page in an index B-tree is called an index node. The top node of the B-tree is called the root node. The bottom level of nodes in the index is called the leaf nodes. Any index levels between the root and the leaf nodes are collectively known as intermediate levels. In a clustered index, the leaf nodes contain the data pages of the underlying table. The root and intermediate level nodes contain index pages holding index rows. Each index row contains a key value and a pointer to either an intermediate level page in the B-tree, or a data row in the leaf level of the index. The pages in each level of the index are linked in a doubly-linked list.

Clustered indexes have one row in sys.partitions, with **index_id = 1** for each partition used by the index. By default, a clustered index has a single partition. When a clustered index has multiple partitions, each partition has a B-tree structure that contains the data for that specific partition. For example, if a clustered index has four partitions, there are four B-tree structures; one in each partition.

Depending on the data types in the clustered index, each clustered index structure will have one or more allocation units in which to store and manage the data for a specific partition. At a minimum, each clustered index will have one IN_ROW_DATA allocation unit per partition. The clustered index will also have one LOB_DATA allocation unit per partition if it contains large

object (LOB) columns. It will also have one ROW_OVERFLOW_DATA allocation unit per partition if it contains variable length columns that exceed the 8,060 byte row size limit.

The pages in the data chain and the rows in them are ordered on the value of the clustered index key. All inserts are made at the point where the key value in the inserted row fits in the ordering sequence among existing rows. The page collections for the B-tree are anchored by page pointers in the **sys.system_internals_allocation_units** system view.

For a clustered index, the **root_page** column in **sys.system_internals_allocation_units** points to the top of the clustered index for a specific partition. SQL Server moves down the index to find the row corresponding to a clustered index key. To find a range of keys, SQL Server moves through the index to find the starting key value in the range and then scans through the data pages using the previous or next pointers. To find the first page in the chain of data pages, SQL Server follows the leftmost pointers from the root node of the index.

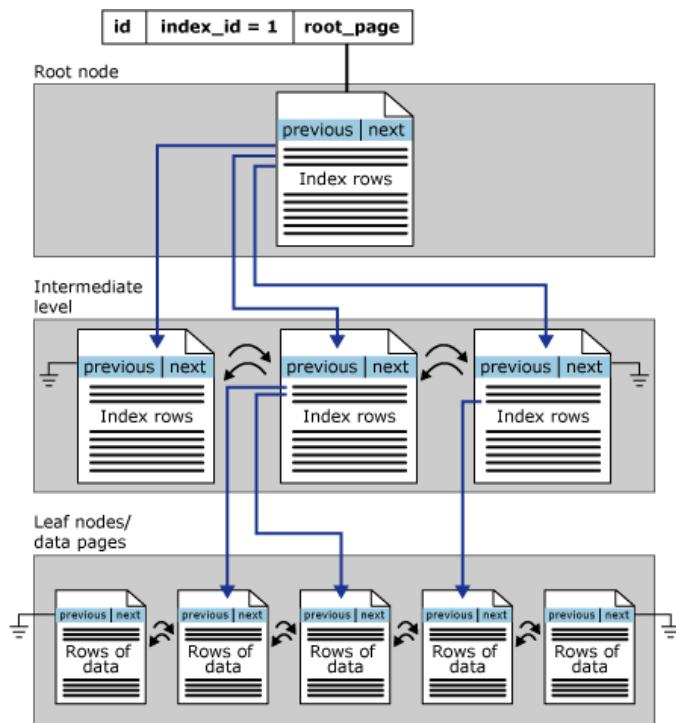


Figure shows the structure of a clustered index in a single partition.

Non clustered Index Structures

Non clustered indexes have the same B-tree structure as clustered indexes, except for the following significant differences:

- The data rows of the underlying table are not sorted and stored in order based on their non clustered keys.
- The leaf layer of a non clustered index is made up of index pages instead of data pages.

Non clustered indexes can be defined on a table or view with a clustered index or a heap. Each index row in the non clustered index contains the non clustered key value and a row locator. This locator points to the data row in the clustered index or heap having the key value.

The row locators in non clustered index rows are either a pointer to a row or are a clustered index key for a row, as described in the following:

- If the table is a heap, which means it does not have a clustered index, the row locator is a pointer to the row. The pointer is built from the file identifier (ID), page number, and number of the row on the page. The whole pointer is known as a Row ID (RID).
- If the table has a clustered index, or the index is on an indexed view, the row locator is the clustered index key for the row. If the clustered index is not a unique index, SQL Server makes any duplicate keys unique by adding an internally generated value called a **uniqueifier**. This four-byte value is not visible to users. It is only added when required to make the clustered key unique for use in non clustered indexes. SQL Server retrieves the data row by searching the clustered index using the clustered index key stored in the leaf row of the non clustered index.

Non clustered indexes have one row in sys.partitions with **index_id >0** for each partition used by the index. By default, a non clustered index has a single partition. When a non clustered index has multiple partitions, each partition has a B-tree structure that contains the index rows for that specific partition. For example, if a non clustered index has four partitions, there are four B-tree structures, with one in each partition.

Depending on the data types in the non clustered index, each non clustered index structure will have one or more allocation units in which to store and manage the data for a specific partition. At a minimum, each non clustered index will have one IN_ROW_DATA allocation unit per partition that stores the index B-tree pages. The non clustered index will also have one LOB_DATA allocation unit per partition if it contains large object (LOB) columns. The page collections for the B-tree are anchored by **root_page** pointers in the **sys.system_internals_allocation_units** system view.

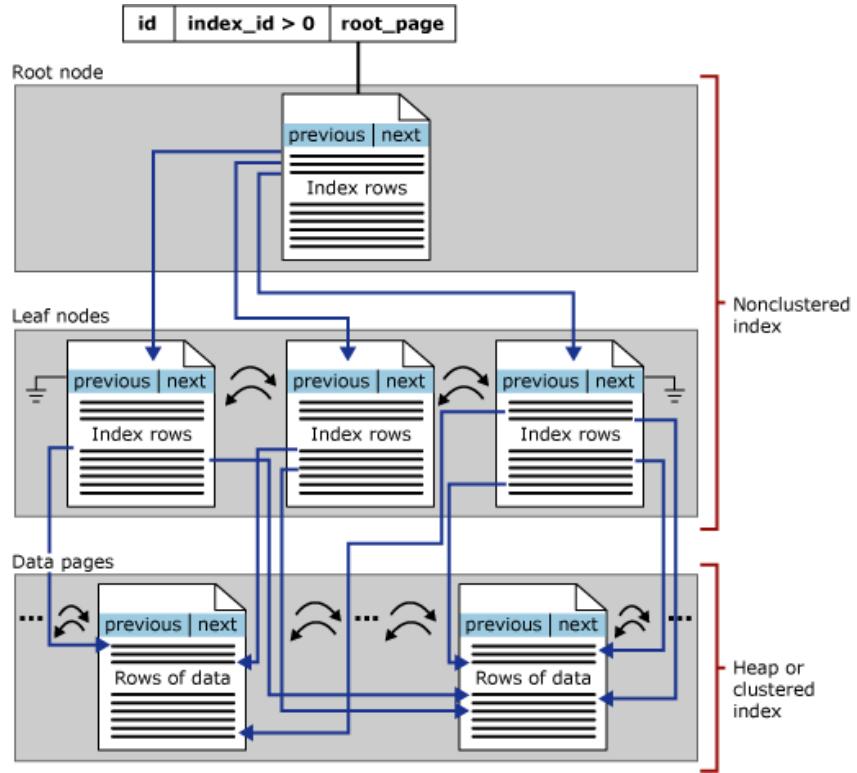


Figure shows the structure of a non clustered index in a single partition.

Understanding Indexes

Good index design starts with a good understanding of the benefits indexes provide. In books, table of contents help readers locate a section, chapter, or page of interest. SQL Server indexes serve the same function as a table of contents in a book. It enables SQL Server to locate and retrieve the data requested in a query as fast as possible.

Consider a 500-page book with dozens of sections and chapters and no table of contents. To locate a section of a book, readers would need to flip and read through every page until they locate the section of interest. Imagine if you have to do this for multiple sections of the book. It would be a time-consuming task.

This analogy also applies to SQL Server database tables. Without proper indexes, SQL Server has to scan through all the data pages that contain the data in a table. For tables with large amounts of data, this becomes time-consuming and resource-intensive. This is the reason why indexes are so important.

Indexes can be classified in several ways depending on the way they store data, their internal structure, their purpose, and the way they are defined. The following sections briefly describe these types of indexes.

Row-based Indexes

A row-based index is a traditional index in which data is stored as rows in data pages. These indexes include the following:

Clustered Indexes

Clustered indexes store and sort data based on the key column(s). There can only be one clustered index per table because data can be sorted in only one order. A clustered index is created by default when a table definition includes a primary key constraint.

Non-clustered Indexes

Non-clustered indexes contain index key values and row locators that point to the actual data row. If there is no clustered index, the row locator is a pointer to the row. When there is a clustered index present, the row locator is the clustered index key for the row.

Non-clustered indexes can be optimized to satisfy more queries, improve query response times, and reduce index size. The two most important of these optimized non-clustered indexes are described in the next two sections.

Covering Indexes

Covering indexes are non-clustered indexes that include non-key columns in the leaf level. These types of indexes improve query performance, cover more queries, and reduce IO operations as the columns necessary to satisfy a query are included in the index itself either as key or non-key columns. Covering indexes can greatly reduce bookmark lookups.

The ability to include nonkey columns enables indexes to be more flexible by including columns with data types not supported as index key columns. Nonkey columns also enable indexes to extend beyond the 16 key column limitation. Nonkey columns do not count towards the 900 byte index key size limit.

Filtered Indexes

Filtered indexes can take a WHERE clause to indicate which rows are to be indexed. Since you index only a portion of rows in a table, you can create only a non-clustered filtered index. If you try to create a filtered clustered index, SQL Server returns a syntax error.

Why do you need a non-clustered index with a subset of data in a table? A well-designed filtered index can offer the following advantages over full-table indexes:

Improved query performance and plan quality: If the index is deep (more pages because of more data), traversing an index takes more I/O and results in slow query performance. If you have a large table and you know that there are more user queries on a well-defined subset of data, creating a filtered index makes queries run faster because less I/O will be performed since the number of pages is less for the smaller amount of data in that filtered index. Moreover, stats on the full table may be less accurate compared to filtered stats with less data, which also helps improve query performance.

Reduced index maintenance costs: Maintaining a filtered index is less costly than maintaining a full index because of smaller data size. Obviously, it also reduces the cost of updating statistics because of the smaller size of the filtered index. As mentioned earlier, you must know your user queries and what kind of data they query often to create a well-defined filtered index with a subset of that data. If the data outside of the filtered index is modified frequently, it won't cost anything to maintain the filtered index. That enables you to create many filtered indexes when the data in those indexes is not modified frequently.

Reduced index storage costs: Less data, less space. If you don't need a full-table non-clustered index, creating a smaller dataset for a non-clustered filtered index takes less disk space.

Column-based Indexes

Column-based indexes are a new type of index introduced in SQL Server 2012 in which only column data is stored in the data pages. These indexes are based on the Vertipaq engine implementation, which is capable of high compression ratios and handles large data sets in memory.

How Indexes are Used by SQL Server?

A good understanding of how indexes are used by SQL Server is also important in a good index design. In SQL Server, the Query Optimizer component determines the most cost-effective option to execute a query. The Query Optimizer evaluates a number of query execution plans and selects the execution plan with the lowest cost.

The execution plan selected by the Query Optimizer may or may not make efficient use of indexes, or it may not use indexes at all. The following sections describe how execution plans can use indexes.

Table Scan

Indexes are not required by SQL Server to retrieve data requested by a query. In the absence of indexes or if determined to be least cost effective, SQL server scans every row of a table until the query is satisfied. This is known as a *table scan*. As you may suspect, table scans can

bring forth expensive IO operations for large tables. SQL Server has to read every single data page until it finds the data that satisfies the query. A table scan can take from a couple of seconds to several minutes. Some users may even experience time-outs by applications with short response-time thresholds.

Table scans generally occur when there is no clustered indexed available; in other words, when the table is a heap.

Index Scan and Index Seek

An *index scan* is similar to a table scan in that SQL Server has to read every single data page in the index until it finds the data that satisfies the query. Index scans can be both IO and memory intensive operations.

An *index seek* on the other hand, is a more efficient way of retrieving data because only data pages and rows that satisfy the query are read. Index seeks result in less data pages read, hence reducing IO and memory consumption.

Depending on how selective a query is, meaning what percentage of the total number of rows in a table is requested, SQL Server Query Optimizer can choose to do an index scan rather than an index seeks. The tipping point at which an index scan is preferred by the SQL Server Query Optimizer is not always a definitive percentage. There are many factors such as parallelism settings, memory availability, and number of rows that contribute in the decision for the more cost-effective option.

Bookmark Lookup

It is quite common to see queries that require additional columns than the ones included in a non-clustered index. To retrieve these additional columns, SQL Server needs to retrieve additional data pages to cover all requested columns. Bookmark lookups can become expensive operations when dealing with a large number of rows because more data pages need to be retrieved from disk and loaded into memory.

To avoid excessive bookmark lookup operations, the required columns that need to be covered by the query can be included in the index definition. These types of indexes are known as covering indexes

The following are the operators related to indexes. They are available in the SQL Server query execution plans.

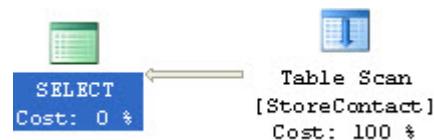
Table scan - A table scan results in reading the entire data in a table and returns the entire table or specific records. This is bad for performance, if the table has numerous records doing a table scan will affect the performance severely. In some cases if there are fewer records its fine to have table scan.

So if you see that SQL Server has performed a Table Scan, take a note of how many rows are in the table. If there aren't many, then in this case, a Table Scan is a good thing.

In the below query in adventureworks database we don't have an index on customerid field and it results in table scan. Also when a Select * is done in a table it will fetch the entire results and hence will do a table scan.

```
Select * from sales.storecontact
```

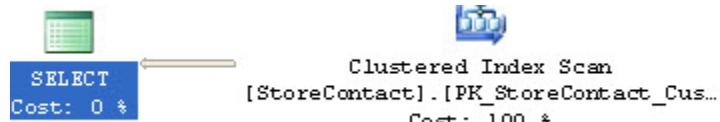
```
Select * from sales.storecontact where customerid=322
```



Clustered Index scan – This is nothing but tables scan in a table which has Clustered index. Since the clustered index leaf page contains the data itself, performing a clustered index scan will scan all the entire records which will affect performance. But as I mentioned earlier if the records are fewer it wouldn't affect much.

For the below query the optimizer does an Clustered index scan to retrieve the records,

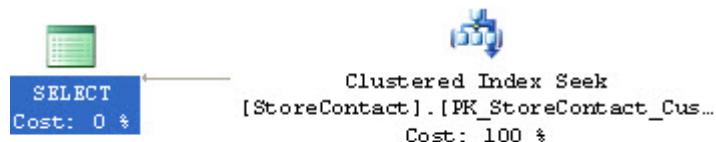
```
Select * from sales.storecontact where contacttypeid=15
```



Clustered index seek – A seek will retrieve only selective records from a table when compared to scan which will traverse all the records in a table. If the seek operation is done using a clustered index it's a clustered index seek. Basically, any seek operation will vertically traverse through the B-Tree and fetch the records.

Consider the below query for which I created a clustered index on Customerid field. The optimizer uses the clustered index and performs a clustered index seek.

```
Select * from sales.storecontact where customerid=322
```



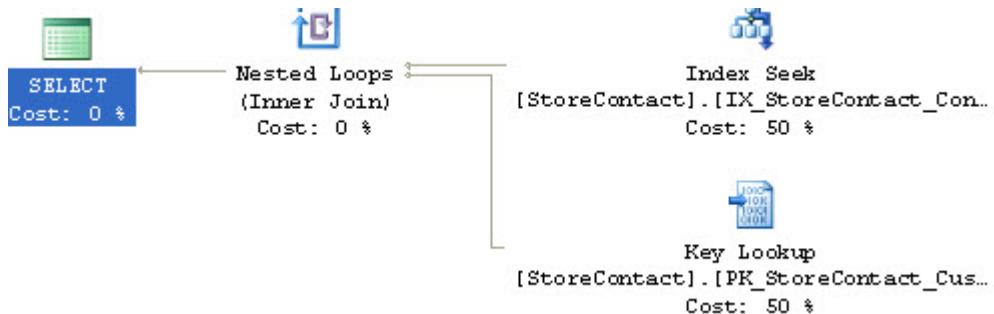
The clustered index seek will traverse through the records where the customerid=322 and fetch the output. When compared to table scan which will traverse through all the records, an index seek is very helpful in reading the number of records quickly and is good for performance.

Index scan – Since a scan touches every row in the table whether or not it qualifies, the cost is proportional to the total number of rows in the table. An index scan is nothing but a scan on the nonclustered index. When index scan happens, all the rows in the leaf level are scanned.

Index seek – An index seek uses the nonclustered index to seek the records in a table and is considered better for performance if there is high selectivity.

For the below query the optimizer does an index seek using the NC index on contacted field. Since the NC covers only the contactid it will not be able to fetch all the records with an index seek alone. So its uses seek to fetch the records which have contactid=322 and then does a key lookup using the clustered index key to fetch the other fields records.

Select * from sales.storecontact where contactid=322



The key lookup is an expensive operation if there are numerous records. Since key lookup increases as the IO we might have to avoid it in some cases. Index with included columns can help to overcome this situation and cover the entire query and in turn causes an index seek.

Index Fragmentation in SQL Server

Index fragmentation is a phenomenon where the index contents are scattered. Normally the contents are in contiguous fashion which helps in fast retrieval of the underlying data. When the indexes are fragmented the data access becomes time consuming because of the scattered data that needs to be searched and read.

Fragmentation occurs as data is modified. The following are the two types of Index fragmentation:

1. Internal fragmentation
2. External fragmentation

Internal fragmentation:

This happens when space is available within your index page i.e. when the index pages have not been filled as full as possible. Due to internal fragmentation the index is taking up more space than it needs to. Thus when scanning the index it results in more read operations. Internal fragmentation also happens due to specifying a low value of fill factor (which determines the % of space to be filled in a leaf level page).

This is also caused by rows that are removed by DELETE statements or when pages are split and only filled to about half. Empty space on pages means there are less rows per page, which in turn means more page reads.

External Fragmentation:

External fragmentation occurs when the pages are not contiguous on the index. If the pages in a book are NOT ordered in a logical way (page 1, then page 2, then page 3 and so on) causing you to go back and forward to compound the information and make sense of the reading. External fragmentation happens when there are frequent UPDATES and INSERTS in a table having small amount of free space in the index page.

Since the page is already full or only has less free space left and if it is not able to accommodate the new row inserted or updated, as a result Page split happens in order to allocate the new row. Due to page split, original page will be split such that half the rows are left on the original page and the other half is moved to the new page. Mostly the new page is not contiguous to the page being split. Page split is an expensive operation and should always be avoided.

How to determine fragmentation?

The following query will give the fragmentation information of a particular table named person.address in adventureworks database. Please modify the query to replace the database name and table name according to your requirements.

```
SELECT CAST(DB_NAME(database_id) AS varchar(20)) AS [Database Name],  
CAST(OBJECT_NAME(object_id) AS varchar(20)) AS [TABLE NAME], Index_id,  
Index_type_desc, Avg_fragmentation_in_percent, Avg_page_space_used_in_percent  
FROM  
sys.dm_db_index_physical_stats(DB_ID('AdventureWorks'),OBJECT_ID('person.address'),NULL,  
NULL,'Detailed')
```

If you wish to identify the fragmentation information for the tables in a particular database please use the below query. I am using it to find the fragmentation in Adventureworks database.

```
SELECT CAST(DB_NAME(database_id) AS varchar(20)) AS [Database Name],  
CAST(OBJECT_NAME(object_id) AS varchar(20)) AS [TABLE NAME], Index_id,
```

```
Index_type_desc, Avg_fragmentation_in_percent, Avg_page_space_used_in_percent  
FROM sys.dm_db_index_physical_stats(DB_ID('AdventureWorks'),NULL,NULL,NULL,'Detailed')
```

Take a look at the output of the following columns in the above query:

Avg_fragmentation_in_percent:

If the value is >5 and <30 you need to REORGANIZE the index using ALTER index REORGANIZE command. If the value is >30 you need to REBUILD the indexes using ALTER index REBUILD command.

Avg_page_space_used_in_percent:

This value represents the amount of page space used in an index. If the value is <75% and >60% we need to REORGANIZE the indexes else REBUILD the indexes.

Defragmenting the Indexes:

We need to either use ALTER INDEX REBUILD or ALTER INDEX REORGANIZE commands to remove the fragmentation from the indexes. Generally its advisable to schedule a job to do this operations in OFF-Production hours as they consume lots of resources.

Column store Index Overview

Because of a proliferation of data being captured across devices, applications, and services, organizations today are tasked with storing massive amounts of data to successfully operate their businesses. Using traditional tools to capture, manage, and process data within an acceptable time is becoming increasingly challenging as the data users want to capture continues to grow. For example, the volume of data is overwhelming the ability of data warehouses to execute queries in a timely manner, and considerable time is spent tuning queries and designing and maintaining indexes to try to get acceptable query performance. In many cases, so much time might have elapsed between the time the query is launched and the time the result sets are returned that organizations have difficulty recalling what the original request was about. Equally unproductive are the cases where the delay causes the business opportunity to be lost.

With the many issues organizations are facing as one of their primary concerns, the Query Processing and Storage teams from the SQL Server product group set to work on new technologies that would allow very large data sets to be read quickly and accurately while transforming the data into useful information and knowledge for organizations in a timely manner. The Query Processing team reviewed academic research in columnstore data representations and analyzed improved query-execution capabilities for data warehousing. In addition, they collaborated with the SQL Server product group Analysis Services team to gain a stronger understanding about the other team's work with their columnstore implementation known as *PowerPivot* for SQL Server 2008 R2. Their research and analysis led the Query Processing team to create the new columnstore index and query optimizations based on vector-based execution capability, which significantly improves data-warehouse query performance.

When developing the new columnstore index, the Query Processing team committed to a number of goals. They aimed to ensure that end users who consume data had an interactive

and positive experience with all data sets, whether large or small, which meant the response time on data must be swift. These strategies also apply to ad hoc and reporting queries. Moreover, database administrators might even be able to reduce their needs for manually tuning queries, summary tables, indexed views, and in some cases OLAP cubes. All these goals naturally impact total cost of ownership (TCO) because hardware costs are lowered and fewer people are required to get a task accomplished.

How Is Data Stored When Using a Columnstore Index?

With traditional tables (heaps) and indexes (B-trees), SQL Server stores data in pages in a row-based fashion. This storage model is typically referred to as a *row store*. Using column stores is like turning the traditional storage model 90 degrees, where all the values from a single column are stored contiguously in a compressed form. The columnstore index stores each column in a separate set of disk pages rather than storing multiple rows per page, which has been the traditional storage format. The following examples illustrate the differences.

Let's use a common table populated with employee data, as illustrated in Table, and then evaluate the different ways data can be stored. This employee table includes typical data such as an employee ID number, employee name, and the city and state the employee is located in.

EmployeeID	Name	City	State
1	Ross	San Francisco	CA
2	Sherry	New York	NY
3	Gus	Seattle	WA
4	Stan	San Jose	CA
5	Lijon	Sacramento	CA

Depending on the type of index chosen—traditional or columnstore—database administrators can organize their data by row (as shown in Table) or by column (as shown in Table).

Row Store
1 Ross San Francisco CA
2 Sherry New York NY
3 Gus Seattle WA
4 Stan San Jose CA
5 Lijon Sacramento CA

Columnstore
1 2 3 4 5
Ross Sherry Gus Stan Lijon
San Francisco New York Seattle San Jose Sacramento
CA NY WA CA CA

As you can see, the major difference between the columnstore format in Table and the row store method in Table is that a columnstore index groups and stores data for each column and then joins all the columns to complete the whole index, whereas a traditional index groups and stores data for each row and then joins all the rows to complete the whole index.

Now that you understand how data is stored when using columnstore indexes compared to traditional B-tree indexes, let's take a look at how this new storage model and advanced query optimizations significantly speed up the retrieval of data. The next section describes three ways that SQL Server columnstore indexes significantly improve the speed of queries.

Creating a Columnstore Index

Creating a columnstore index is very similar to creating any other traditional SQL Server indexes. You can use the graphical user interface in SQL Server Management Studio or Transact-SQL. Many individuals prefer to use the graphical user interface because they want to avoid typing all of the column names, which is what they have to do when creating the index with Transact-SQL.

A few questions arise frequently when creating a columnstore index. Database administrators often want to know the following:

- Which columns should be included in the columnstore index?
- Is it possible to create a clustered columnstore index?

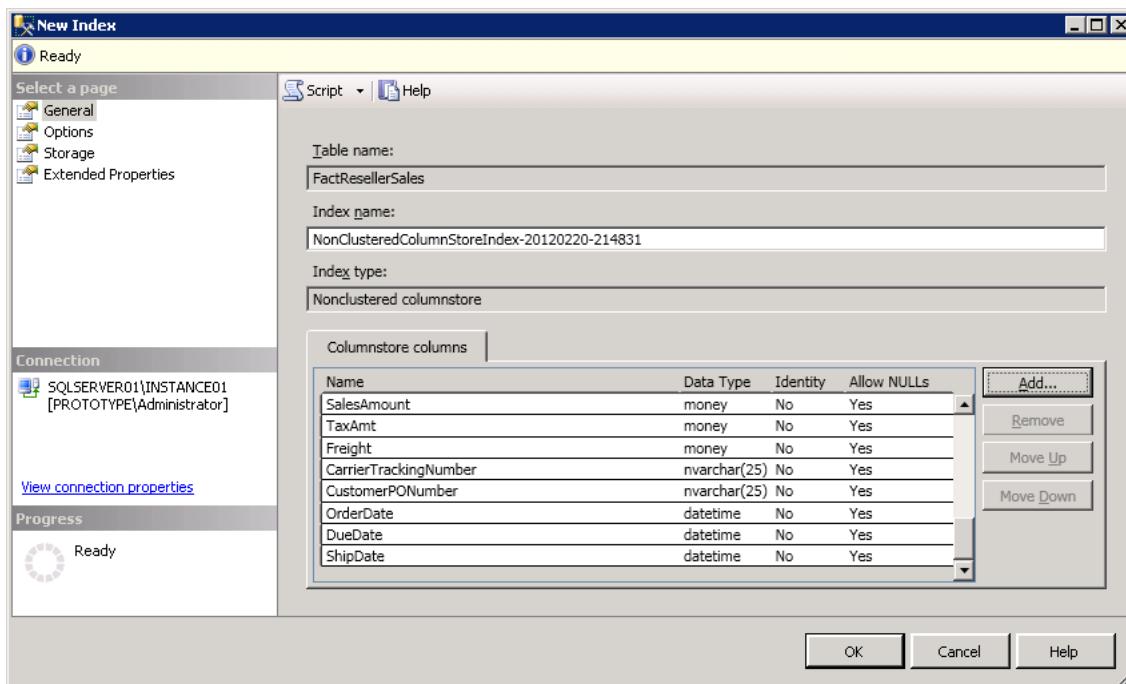
When creating a columnstore index, a database administrator should typically include all of the supported columnstore index columns associated with the table. Note that you don't need to

include all of the columns. The answer to the second question is “No.” All columnstore indexes must be nonclustered; therefore, a clustered columnstore index is not allowed.

The next sections explain the steps for creating a columnstore index using SQL Server Management Studio.

Here are the steps for creating a columnstore index using SQL Server Management Studio (SSMS):

1. In SQL Server Management Studio, use Object Explorer to connect to an instance of the SQL Server Database Engine.
2. In Object Explorer, expand the instance of SQL Server, expand Databases, expand a database, and expand a table in which you would like to create a new columnstore index.
3. Expand the table, right-click the Index folder, choose New Index, and then click Non-Clustered Columnstore Index.
4. On the General tab, in the Index name box, type a name for the new index, and then click Add.
5. In the Select Columns dialog box, select the columns to participate in the columnstore index and then click OK.
6. If desired, configure the settings on the Options, Storage, and Extended Properties pages. If you want to maintain the defaults, click OK to create the index,



Best Practices on Indexing

- Periodically, run the Index Wizard or Database Engine Tuning Advisor against current Profiler traces to identify potentially missing indexes.
- Remove indexes that are never used.
- Don't accidentally create redundant indexes.
- As a rule of thumb, every table should have at least a clustered index. Generally, but not always, the clustered index should be on a column that monotonically increases — such as an identity column, or some other column where the value is increasing — and is unique. In many cases, the primary key is the ideal column for a clustered index.
- Since you can only create one clustered index per table, take extra time to carefully consider how it will be used. Consider the type of queries that will be used against the table, and make an educated guess as to which query (the most common one run against the table, perhaps) is the most critical, and if this query will benefit from having a clustered index.
- If a column in a table is not at least 95% unique, then most likely the query optimizer will not use a non-clustered index based on that column. Because of this, you generally don't want to add non-clustered indexes to columns that aren't at least 95% unique.
- Keep the "width" of your indexes as narrow as possible. This reduces the size of the index and reduces the number of disk I/O reads required to read the index, boosting performance.
- If possible, avoid adding a clustered index to a GUID column (uniqueidentifier data type). GUIDs take up 16-bytes of storage, more than an Identity column, which makes the index larger, which increases I/O reads, which can hurt performance.
- Indexes should be considered on all columns that are frequently accessed by the JOIN, WHERE, ORDER BY, GROUP BY, TOP, and DISTINCT clauses.
- Don't automatically add indexes on a table because it seems like the right thing to do. Only add indexes if you know that they will be used by the queries run against the table.
- When creating indexes, try to make them unique indexes if at all possible. SQL Server can often search through a unique index faster than a non-unique index because in a unique index, each
- Row is unique, and once the needed record is found, SQL Server doesn't have to look any further.
- If you perform regular joins between two or more tables in your queries, performance will be optimized if each of the joined columns has appropriate indexes.
- Don't automatically accept the default value of 100 for the fill factor for your indexes. It may or may not best meet your needs. A high fill factor is good for seldom changed data, but highly
- Modified data needs a lower fill factor to reduce page splitting.

- Don't over index your OLTP tables, as every index you add increases the time it takes to perform INSERTS, UPDATES, and DELETES. There is a fine line between having the ideal number of
- Indexes (for SELECTs) and the ideal number to minimize the overhead that occurs with indexes during data modifications.
- If you know that your application will be performing the same query over and over on the same table, consider creating a non-clustered covering index on the table. A covering index, which is a form of a composite index, includes all of the columns referenced in SELECT, JOIN, and WHERE clauses of a query. Because of this, the index contains the data you are looking for and SQL Server doesn't have to look up the actual data in the table, reducing logical and/or physical I/O, and boosting performance.

Case study 1: Is your indexes are being used effectively?

In this fast moving world, data is the heart and soul of any enterprise. As the data is growing very rapidly day by day, the biggest challenge which enterprises face today is to store the data in such a way that it can be retrieved quickly whenever required. The most common thought which comes in the mind of database administrators who basically works on performance improvement is to add indexes for tables to improve the data retrieval. However adding too many indexes on a table can sometimes reduce the performance of the table considerably. So it is very important for the database administrator to know whether the indexes created on the tables are used effectively or not. If there are indexes created on a table and they are not used, then they should be drop, as having unwanted index will slow down Insert, Update or Delete operations on the underlying tables.

It has been always a challenge for database administrators to figure out which indexes on a table are helpful and which aren't. In SQL Server 2005, Microsoft introduced Dynamic Management Views (DMV) which return server state information that can be used by developers or database administrators to monitor the health of a SQL Server Instance and identify potential performance issues. Dynamic Management Views basically reflect all the activities on the instance of SQL Server since the last restart of SQL Server. All the Dynamic Management Views exist in the SYS schema and they can be easily identify as they follow the naming convention of dm_*. The list of all the Dynamic Management Views that are available on SQL Server 2005 and higher versions can be obtained by running the below TSQL code.

```
USE master
GO
SELECT * FROM sys.sysobjects WHERE NAME LIKE 'dm_%'
GO
```

Unfortunately in the SQL Server editions prior to SQL Server 2005 there is no easy way to identify indexes which are helpful and which aren't. In SQL Server 2000 the only way to identify if an index is being used or not was to capture a workload in profiler and then run it against the Index Tuning Wizard.

Some of the disadvantage of having too many indexes on a database table

- a) Insert, Update and Delete operations will become very slow if there are many indexes created on a table. This happens because when the Insert, Update or a Delete operation occurs against a table all the indexes will be updated, thereby reducing query performance
- b) Indexes are basically stored on disk; the amount of disk space required by the index depends on the size of database table, and the number and type of columns used in the index definition.
- c) The more the indexes, the more disk space that is required to store the indexes.

Example to verify whether indexes on a SQL Server table are used effectively or not It is not advisable to simply go ahead and disable or drop any index on any SQL Server table without doing the proper investigation. As a database administrator or a database developer you need to make sure that you drop only those indexes which are not or rarely used by queries.

In this example we will be using the HumanResources.Employee table of AdventureWorks database. The first step will be to find out how many indexes are created on the HumanResources.Employee table. This can be done by using the sp_helpindex stored procedure which accepts ObjectOwner.TableName as a parameter.

Use AdventureWorks

GO

sp_helpindex 'HumanResources.Employee'

GO

The screenshot shows the SQL Server Management Studio interface. In the top pane, titled 'Identify Indexes On Employee...', there is a green bar with the text 'Use AdventureWorks' and a 'GO' button. Below this, a green bar contains the command 'sp_helpindex 'HumanResources.Employee'' followed by another 'GO' button. In the bottom pane, titled 'Results', there is a table with five rows of data. The table has three columns: 'index_name', 'index_description', and 'index_keys'. The data is as follows:

index_name	index_description	index_keys
1 AK_Employee_LoginID	nonclustered, unique located on PRIMARY	LoginID
2 AK_Employee_NationalIDNumber	nonclustered, unique located on PRIMARY	NationalIDNumber
3 AK_Employee_rowguid	nonclustered, unique located on PRIMARY	rowguid
4 IX_Employee_ManagerID	nonclustered located on PRIMARY	ManagerID
5 PK_Employee_EmployeeID	clustered, unique, primary key located on PRIMARY	EmployeeID

You can see from the above image that there are five indexes defined on the HumanResources.Employee table in the AdventureWorks database. In SQL Server 2005 and above, the information related to index usage is stored in the sys.dm_db_index_usage_stats Dynamic Management Views (DMV). By executing the below mentioned query you can identify the current index usage information for the HumanResources.Employee table in the AdventureWorks database. However, in order to

access the information stored in the sys.dm_db_index_usage_stats Dynamic Management Views (DMV) you need to have VIEW SERVER STATE permissions. Only members of the sysadmin fixed server role can grant VIEW SERVER STATE permissions to other users.

```
USE AdventureWorks
GO
DECLARE @TABLENAME SYSNAME
SET @TABLENAME= 'HumanResources.Employee'
SELECT DB_NAME(DATABASE_ID) AS [DATABASE NAME]
, OBJECT_NAME(SS.OBJECT_ID) AS [OBJECT NAME]
, I.NAME AS [INDEX NAME]
, I.INDEX_ID AS [INDEX ID]
, USER_SEEKS AS [NUMBER OF SEEKS]
, USER_SCANS AS [NUMBER OF SCANS]
, USER_LOOKUPS AS [NUMBER OF BOOKMARK LOOKUPS]
, USER_UPDATES AS [NUMBER OF UPDATES]
FROM
    SYS.DM_DB_INDEX_USAGE_STATS SS
    INNER JOIN SYS.INDEXES I
        ON I.OBJECT_ID = SS.OBJECT_ID
        AND I.INDEX_ID = SS.INDEX_ID
WHERE DATABASE_ID = DB_ID()
    AND OBJECTPROPERTY (SS.OBJECT_ID,'IsUserTable') = 1
    AND SS.OBJECT_ID = OBJECT_ID(@TABLENAME)
ORDER BY USER_SEEKS
    , USER_SCANS
    , USER_LOOKUPS
    , USER_UPDATES ASC
GO
```

Query To Identify Index Usage

```

USE AdventureWorks
GO

DECLARE @TABLENAME SYSNAME
SET @TABLENAME='HumanResources.Employee'

SELECT DB_NAME(DATABASE_ID) AS [DATABASE NAME]
      ,OBJECT_NAME(SS.OBJECT_ID) AS [OBJECT NAME]
      ,I.NAME AS [INDEX NAME]
      ,I.INDEX_ID AS [INDEX ID]
      ,USER_SEEKS AS [NUMBER OF SEEKS]
      ,USER_SCANS AS [NUMBER OF SCANS]
      ,USER_LOOKUPS AS [NUMBER OF BOOKMARK LOOKUPS]
      ,USER_UPDATES AS [NUMBER OF UPDATES]
FROM   sys.dm_db_index_usage_stats SS
INNER JOIN sys.indexes I
        ON I.OBJECT_ID = SS.OBJECT_ID
       AND I.INDEX_ID = SS.INDEX_ID
WHERE DATABASE_ID = DB_ID()
      AND OBJECTPROPERTY(SS.OBJECT_ID,'IsUserTable') = 1
      AND SS.OBJECT_ID = OBJECT_ID(@TABLENAME)
ORDER BY USER_SEEKS
      ,USER_SCANS
      ,USER_LOOKUPS
  
```

Results

	DATABASE NAME	OBJECT NAME	INDEX NAME	INDEX ID	NUMBER OF SEEKS	NUMBER OF SCANS	NUMBER OF BOOKMARK LOOKUPS	NUMBER OF UPDATES
1	AdventureWorks	Employee	IX_Employee_ManagerID	5	20	0	0	0
2	AdventureWorks	Employee	AK_Employee_rowguid	4	20	0	0	0
3	AdventureWorks	Employee	AK_Employee_NationalIDNumber	3	20	0	0	0
4	AdventureWorks	Employee	AK_Employee_LoginID	2	20	0	0	0
5	AdventureWorks	Employee	PK_Employee_EmployeeID	1	20	0	80	0

Query executed successfully. AKMEHTA|SQL2008 (10.0 RTM)

From the above image you can see that all the indexes are used 20 times. This information is from the time when the SQL Server Service was last restarted. Once the SQL Server Service comes up after the restart all the values in sys.dm_db_index_usage_stats Dynamic Management Views (DMV) are reset to zero.

In the next step let us execute the below mentioned select statements which utilizes the indexes defined on the HumanResources.Employee table of the AdventureWorks database and see whether the changes are getting reflected in the sys.dm_db_index_usage_stats Dynamic Management Views (DMV):

```

Use Adventure Works
GO
SELECT * FROM HumanResources.Employee WITH (INDEX = 1)
WHERE EmployeeID = '200'
GO
SELECT * FROM HumanResources.Employee WITH (INDEX = 2)
WHERE LoginID = 'adventure-works\hazem0'
GO
  
```

```

SELECT * FROM HumanResources.Employee WITH (INDEX = 3)
WHERE NationalIDNumber='398223854'
GO
SELECT * FROM HumanResources.Employee WITH (INDEX = 4)
WHERE rowguid='05C84608-F445-4F9D-BB5C-0828C309C29D'
GO

```

Queries Utilizing Indexes.sql ...)*

Use AdventureWorks

GO

SELECT * FROM HumanResources.Employee WITH (INDEX = 1) WHERE EmployeeID = 200

GO

SELECT * FROM HumanResources.Employee WITH (INDEX = 2) WHERE LoginID = 'adventure-works\hazem0'

GO

SELECT * FROM HumanResources.Employee WITH (INDEX = 3) WHERE NationalIDNumber='398223854'

GO

SELECT * FROM HumanResources.Employee WITH (INDEX = 4) WHERE rowguid='05C84608-F445-4F9D-BB5C-0828C309C29D'

GO

Results Messages

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title	BirthDate	MaritalStatus	Gender	
1	200	398223854	1268	adventure-works\hazem0	148	Quality Assurance Manager	1967-11-27 00:00:00.000	S	M

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title	BirthDate	MaritalStatus	Gender	
1	200	398223854	1268	adventure-works\hazem0	148	Quality Assurance Manager	1967-11-27 00:00:00.000	S	M

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title	BirthDate	MaritalStatus	Gender	
1	200	398223854	1268	adventure-works\hazem0	148	Quality Assurance Manager	1967-11-27 00:00:00.000	S	M

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title	BirthDate	MaritalStatus	Gender	
1	200	398223854	1268	adventure-works\hazem0	148	Quality Assurance Manager	1967-11-27 00:00:00.000	S	M

You could see that the above mentioned queries were executed with a specific index hint. This is done in order to make use of specific indexes when executing queries against the HumanResources.Employee table. Now let us execute the below query to see whether the usage of the index is updated in the sys.dm_db_index_usage_stats Dynamic Management Views (DMV) or not.

Query To Identify Index Usage—

```

USE AdventureWorks
GO

DECLARE @TABLENAME sysname
SET @TABLENAME= 'HumanResources.Employee'

SELECT DB_NAME(DATABASE_ID) AS [DATABASE NAME]
, OBJECT_NAME(SS.OBJECT_ID) AS [OBJECT NAME]
, I.NAME AS [INDEX NAME]
, I.INDEX_ID AS [INDEX ID]
, USER_SEEKS AS [NUMBER OF SEEKS]
, USER_SCANS AS [NUMBER OF SCANS]
, USER_LOOKUPS AS [NUMBER OF BOOKMARK LOOKUPS]
, USER_UPDATES AS [NUMBER OF UPDATES]
FROM
SYS.DIM_DB_INDEX_USAGE_STATS SS
INNER JOIN SYS.INDEXES I
ON I.OBJECT_ID = SS.OBJECT_ID
AND I.INDEX_ID = SS.INDEX_ID
WHERE DATABASE_ID = DB_ID()
AND OBJECTPROPERTY(SS.OBJECT_ID,'IsUserTable') = 1
AND SS.OBJECT_ID = OBJECT_ID(@TABLENAME)
ORDER BY USER_SEEKS
, USER_SCANS
, USER_LOOKUPS

```

Results

DATABASE NAME	OBJECT NAME	INDEX NAME	INDEXID	NUMBER OF SEEKS	NUMBER OF SCANS	NUMBER OF BOOKMARK LOOKUPS	NUMBER OF UPDATES
1 AdventureWorks	Employee	IX_Employee_ManagerID	5	20	0	0	0
2 AdventureWorks	Employee	AK_Employee_rowguid	4	21	0	0	0
3 AdventureWorks	Employee	AK_Employee_NationalIDNumber	3	21	0	0	0
4 AdventureWorks	Employee	AK_Employee_LoginID	2	21	0	0	0
5 AdventureWorks	Employee	PK_Employee_EmployeeID	1	21	0	83	0

Query executed successfully.

AKMEHTA(SQL2008 (10.0 RTM))

You could see that the values for INDEX ID from 1...4 has incremented by 1 to 21 from the previous value of 20 and for INDEX ID 5 the value has remained unchanged. In this way you can identify which indexes are really helpful and which are very rarely used. If you are interested to know when each of the above indexes were last used then you can use the query below:

```

USE AdventureWorks
GO
DECLARE @TABLENAME sysname
SET @TABLENAME= 'HumanResources.Employee'
SELECT DB_NAME(DATABASE_ID) AS [DATABASE NAME]
, OBJECT_NAME(SS.OBJECT_ID) AS [OBJECT NAME]
, I.NAME AS [INDEX NAME]
, I.INDEX_ID AS [INDEX ID]
, USER_SEEKS AS [NUMBER OF SEEKS]
, LAST_USER_SEEK AS [LAST USER SEEK]
, USER_SCANS AS [NUMBER OF SCANS]
, LAST_USER_SCAN AS [LAST USER SCAN]
, USER_LOOKUPS AS [NUMBER OF BOOKMARK LOOKUPS]
, LAST_USER_LOOKUP AS [LAST USER LOOKUP]
, USER_UPDATES AS [NUMBER OF UPDATES]
, LAST_USER_UPDATE AS [LAST USER UPDATE]

```

```

FROM
    SYS.DM_DB_INDEX_USAGE_STATS SS
INNER JOIN SYS.INDEXES I
    ON I.OBJECT_ID = SS.OBJECT_ID
        AND I.INDEX_ID = SS.INDEX_ID
WHERE DATABASE_ID = DB_ID()
    AND OBJECTPROPERTY(SS.OBJECT_ID,'IsUserTable') = 1
    AND SS.OBJECT_ID = OBJECT_ID(@TABLENAME)
ORDER BY USER_SEEKS
    , USER_SCANS
    , USER_LOOKUPS
    , USER_UPDATES ASC
GO

```

Before dropping an index it would be ideal to disable the index on the table and see if there is any performance hit when the index is disabled. Disabling a non clustered index prevents users from accessing the particular non clustered index defined on the underlying table. However, the index definition remains in metadata and index statistics are also kept on nonclustered indexes. You need to keep in mind that if you disable the clustered index by any chance then you will not be able to access the data in the underlying table until the index is dropped or rebuilt.

Disable IX_Employee_ManagerID index on HumanResources.Employee table

```

USE AdventureWorks
GO
ALTER INDEX IX_Employee_ManagerID ON HumanResources.Employee
DISABLE
GO

```

Enable IX_Employee_ManagerID index on HumanResources.Employee table

```

USE AdventureWorks
GO
ALTER INDEX IX_Employee_ManagerID ON HumanResources.Employee REBUILD
GO

```

Finally after the analysis when it is discovered that the queries are not using the IX_Employee_ManagerID index then you can execute the below TSQL code to drop the index on the HumanResources.Employee table.

Drop IX_Employee_ManagerID index on HumanResources.Employee table

```

USE AdventureWorks
GO
DROP INDEX IX_Employee_ManagerID ON HumanResources.Employee
GO

```

Conclusion

Using the sys.dm_db_index_usage_stats Dynamic Management View, you can easily identify indexes which are used often by queries and also indexes which are rarely used. Once you have identified the indexes, the best approach is to disable the indexes for some time and see if

there is any performance degradation once the indexes are disabled. If there is no performance degradation then you can go ahead and drop the index thereby saving disk space and improving performance during Insert, Update and Delete operations.

Case study 2: Analyze and Fix Index Fragmentation in SQL Server 2012

It is very common that over time SQL Server tables and indexes tend to become fragmented. The fragmentation generally happens when data within the underlying tables on which an index exists is modified. The data modification basically can be an insert, update or a delete operation. The indexes over time become ineffective because they get fragmented. In this case study you will see an example of how an index gets fragmented and the steps which database administrator needs to take to fix index fragmentations.

Example to Analyze and Fix Index Fragmentation in SQL Server 2012

Follow the below mentioned steps to see how an index fragmentation occurs on a table which has indexes defined on it. And finally you will see the steps which you need to take to fix index fragmentation issues.

Create AnalyzeFragmentation Database

First let us create a new database named AnalyzeFragmentation for this example. Database can be created by executing the below mentioned TSQL Query.

```
Use master  
GO  
CREATE DATABASE AnalyzeFragmentation  
GO
```

Create FindAndFixFragmentation Table in AnalyzeFragmentation Database

The next step will be to create a new table named FindAndFixFragmentation within the AnalyzeFragmentation database.

```
USE AnalyzeFragmentation  
GO  
/* Create FindAndFixFragmentation Table*/  
  
CREATE TABLE [dbo].[FindAndFixFragmentation]  
(  
    [AddressID] [int] NOT NULL,  
    [AddressLine1] [nvarchar](60) NOT NULL,  
    [City] [nvarchar](30) NOT NULL,  
    [PostalCode] [nvarchar](15) NOT NULL,  
    [ModifiedDate] [datetime] NOT NULL,  
    [RowGUID] [UNIQUEIDENTIFIER] NOT NULL  
)  
ON [PRIMARY]  
GO
```

Populate the FindAndFixFragmentation Table using the below TSQL code

The next step will be to populate the FindAndFixFragmentation table which you have created earlier by executing the below mentioned TSQL code. For this example we will be using the data which is available in Person.Address table available in AdventureWorks database.

```
USE AnalyzeFragmentation
GO
/* Populate FindAndFixFragmentation table with data from AdventureWorks.Person.Address */
INSERT INTO FindAndFixFragmentation
SELECT
AddressID,
AddressLine1,
City,
PostalCode,
ModifiedDate,
RowGUID
FROM AdventureWorks.Person.Address
GO
```

Create a Clustered Index on FindAndFixFragmentation Table using the below TSQL code

The next step will be to create a clustered index named CL_FindAndFixFragmentation_Index on FindAndFixFragmentation table using the below mentioned TSQL code.

```
/* Drop the index if it is already existing*/
IF EXISTS (SELECT * FROM sys.indexes WHERE object_id =
OBJECT_ID(N'[dbo].[FindAndFixFragmentation]') AND name =
N'CL_FindAndFixFragmentation_Index')
DROP INDEX [CL_FindAndFixFragmentation_Index] ON [dbo].[FindAndFixFragmentation]
GO
/* Create Clustered Index on FindAndFixFragmentation(RowGUID) */
CREATE CLUSTERED INDEX [CL_FindAndFixFragmentation_Index] ON
[dbo].[FindAndFixFragmentation]
(
    [RowGUID] ASC
)
WITH (FILLFACTOR = 90) ON [PRIMARY]
GO
```

You can see that we are creating a clustered index on FindAndFixFragmentation table with a Fill Factor 90. The fill factor option is basically provided for fine tuning index data storage and to improve performance. Whenever an index is created or it is rebuilt, the fill factor value basically determines the percentage of space on each leaf level page that needs to be filled with data. Based on the fill factor value a percentage of free space is allocated on every single page. By default the fill factor value is 0 or 100 which means there will be no free space allocated on each leaf level page. The value for fill factor is defined in percentages and this can be any value

in between 1 to 100. In this example the fill factor value provide is 90 which mean on every single page there will be a 10 percentage of free space left to accommodate future growth.

Query to Find Existing Fragmentation on FindAndFixFragmentation Table

Next step will be to execute the below mentioned TSQL query to know the existing fragmentation on FindAndFixFragmentation table. The important values which need to be noted by the database administrators are AvgPageFragmentation and PageCounts. The value for AvgPageFragmentation is 0.341296928327645, which means there is a very little fragmentation existing on the table at this point of time. However the value for PageCounts is 293, which mean the data is stored in that many data pages on SQL Server. This query will be executing many a times in this case study.

```
/* Find index fragmentation */
SELECT
    DB_NAME(DATABASE_ID) AS [DatabaseName],
    OBJECT_NAME(OBJECT_ID) AS TableName,
    SI.NAME AS IndexName,
    INDEX_TYPE_DESC AS IndexType,
    AVG_FRAGMENTATION_IN_PERCENT AS AvgPageFragmentation,
    PAGE_COUNT AS PageCounts
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL , NULL, N'LIMITED') DPS
INNER JOIN sysindexes SI
ON DPS.OBJECT_ID = SI.ID AND DPS.INDEX_ID = SI.INDID
GO
```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'FindAndFixInd...akmehta (54)*' contains the following T-SQL code:

```

/* Find index fragmentation */
SELECT
    DB_NAME(DB_ID) AS [DatabaseName],
    OBJECT_NAME(OBJECT_ID) AS TableName,
    SI.NAME AS IndexName,
    INDEX_TYPE_DESC AS IndexType,
    AVG_FRAGMENTATION_IN_PERCENT AS AvgPageFragmentation,
    PAGE_COUNT AS PageCounts
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL, NULL, N'LIMITED') DPS
INNER JOIN sysindexes SI
    ON DPS.OBJECT_ID = SI.ID AND DPS.INDEX_ID = SI.INDID
GO

```

The results pane displays a table with the following data:

DatabaseName	TableName	IndexName	IndexType	AvgPageFragmentation	PageCounts
AnalyzeFragmentation	FindAndFixFragmentation	CL_FindAndFixFragmentation_Index	CLUSTERED INDEX	0.341296328327645	293

At the bottom of the results pane, a message states: 'Query executed successfully.' and shows the session details: AKMEHTA|SQL2008 (10.0 RTM) | akmehta (54) | AnalyzeFragmentation | 00:00:00 | 1 rows.

Perform Update Operation on FindAndFixFragmentation Table

Next step will be to perform updates on FindAndFixFragmentation table by executing the below mentioned T-SQL code. This query will modify all the data for RowGUID column on which we have created clustered index with fill factor as 90.

```

/* Update all the rows within to FindAndFixFragmentation table create index fragmentation */
USE AnalyzeFragmentation
GO
UPDATE FindAndFixFragmentation
SET RowGUID = NEWID()
GO
Execute the query to find existing fragmentation on FindAndFixFragmentation table as shown in the below snippet.

```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'FindAndFixInd...akmehta (54)*' contains the following T-SQL code:

```

/* Find index fragmentation */
SELECT 
    DB_NAME(DATABASE_ID) AS [DatabaseName],
    OBJECT_NAME(OBJECT_ID) AS TableName,
    SI.NAME AS IndexName,
    INDEX_TYPE_DESC AS IndexType,
    AVG_FRAGMENTATION_IN_PERCENT AS AvgPageFragmentation,
    PAGE_COUNT AS PageCounts
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, N'LIMITED') DPS
INNER JOIN sysindexes SI
    ON DPS.OBJECT_ID = SI.ID AND DPS.INDEX_ID = SI.INDID
GO

```

The results pane displays a single row of data:

DatabaseName	TableName	IndexName	IndexType	AvgPageFragmentation	PageCounts
AnalyzeFragmentation	FindAndFixFragmentation	CL_FindAndFixFragmentation_Index	CLUSTERED INDEX	99.0049751243781	603

At the bottom of the interface, a status bar indicates: 'Query executed successfully.' and 'AKMEHTA|SQL2008 (10.0 RTM) akmehta (54) AnalyzeFragmentation 00:00:00 1 rows'.

Now you can see that the value for AvgPageFragmentation has changed from 0.341296928327645 to 99.0049751243781, which means index is completely fragmentation. At the same time the value for PageCounts has changed from 293 to 603, which mean more number of data pages are required to store the content. Now the question which comes to your mind is how this can be fixed.

There are two methods to fix index fragmentation issues in SQL Server 2005 and higher versions. The two methods are Reorganize or Rebuild Index. The Reorganize Index is an online operation, however Rebuild Index is not an online operation until you have specified the option ONLINE=ON while performing the Rebuild. Next step will be to perform first REORGANIZE Index option and then finally perform we will perform the REBUILD and see which options is the best.

Perform Reorganize Index Operation on Clustered Index of FindAndFixFragmentation Table
First let us perform REORGANIZE Index operation on the clustered index, and then execute the query as shown in the snippet to find the fragmentation on FindAndFixFragmentation table.

```

/* Reorganize [CL_FindAndFixFragmentation_Index] index on
FindAndFixFragmentation */
ALTER INDEX [CL_FindAndFixFragmentation_Index] ON
FindAndFixFragmentation
REORGANIZE;
GO

```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'FindAndFixInd...akmehta (S4)*' contains the following T-SQL code:

```

/* Find index fragmentation */
SELECT
    DB_NAME(DATABASE_ID) AS [DatabaseName],
    OBJECT_NAME(OBJECT_ID) AS TableName,
    SI.NAME AS IndexName,
    INDEX_TYPE_DESC AS IndexType,
    AVG_FRAGMENTATION_IN_PERCENT AS AvgPageFragmentation,
    PAGE_COUNT AS PageCounts
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, N'LIMITED') DPS
INNER JOIN sysindexes SI
ON DPS.OBJECT_ID = SI.ID AND DPS.INDEX_ID = SI.INDID
GO

```

The 'Results' tab displays the output of the query:

DatabaseName	TableName	IndexName	IndexType	AvgPageFragmentation	PageCounts
AnalyzeFragmentation	FindAndFixFragmentation	CL_FindAndFixFragmentation_Index	CLUSTERED INDEX	5.70469798657718	298

At the bottom of the results window, a status bar indicates: 'Query executed successfully.' and '1 rows'.

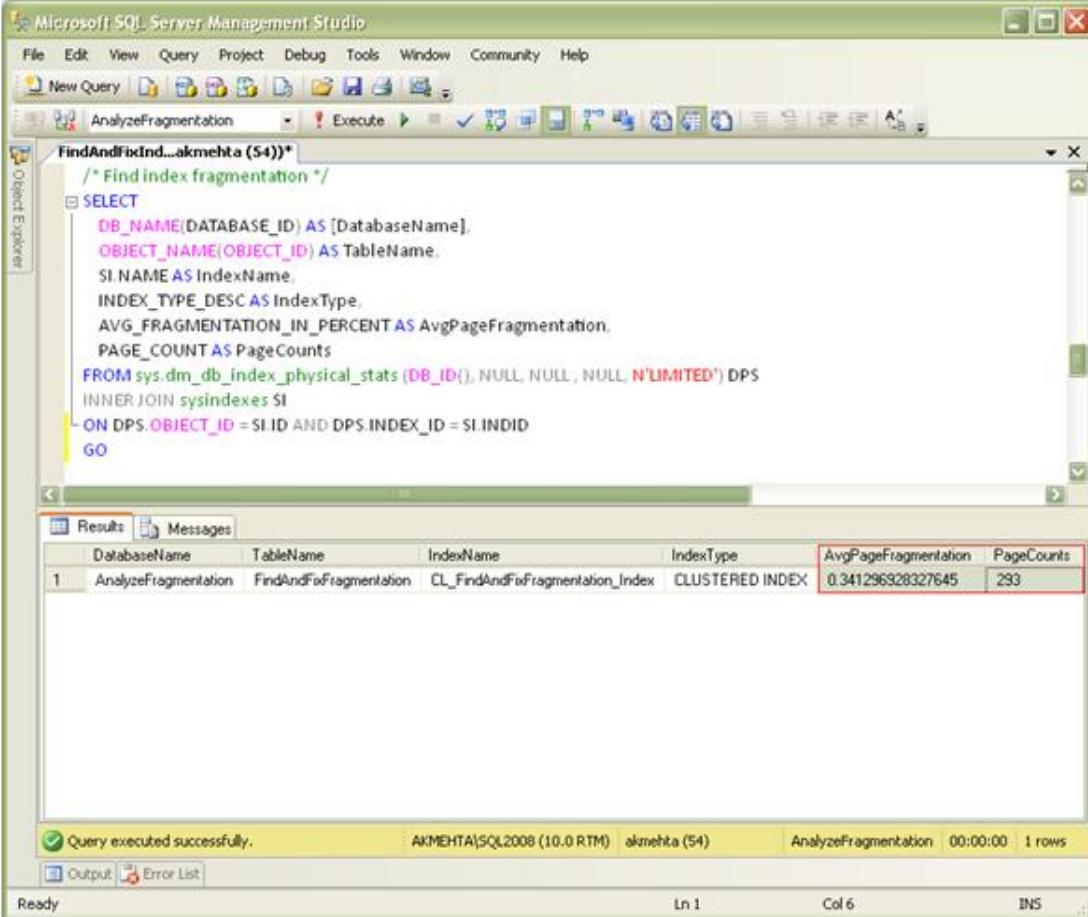
Once we have performed the REORGANIZE Index operation you can see that the value for AvgPageFragmentation has changed from 99.0049751243781 to 5.70469798657718, which means index fragmentation is much better than how it was earlier. And at the same time the value for PageCounts has also come down from 603 to 298, this is considerable improvement.

Perform Rebuild Index Operation on Clustered Index of FindAndFixFragmentation Table

Now let us perform REBUILD Index operation on the clustered index, when you are using the

Rebuild index operation it basically drops and recreates the index. The important thing what we need to see is does this results in reducing the index fragmentation further down from 5.70469798657718. Once you have performed the Rebuild operation execute the query as shown in the snippet to check the fragmentation on FindAndFixFragmentation table.

```
/* Rebuild [CL_FindAndFixFragmentation_Index] index on
FindAndFixFragmentation */
ALTER INDEX [CL_FindAndFixFragmentation_Index] ON
FindAndFixFragmentation
REBUILD WITH (FILLFACTOR = 90, ONLINE=ON)
GO
```



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'FindAndFixInd...akmehta (54)*' contains the following T-SQL code:

```
/* Find index fragmentation */
SELECT
    DB_NAME(DB_ID) AS [DatabaseName],
    OBJECT_NAME(OBJECT_ID) AS TableName,
    SI.NAME AS IndexName,
    INDEX_TYPE_DESC AS IndexType,
    AVG_FRAGMENTATION_IN_PERCENT AS AvgPageFragmentation,
    PAGE_COUNT AS PageCounts
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, N'LIMITED') DPS
INNER JOIN sysindexes SI
    ON DPS.OBJECT_ID = SI.ID AND DPS.INDEX_ID = SI.INDID
GO
```

The 'Results' tab displays the output of the query:

DatabaseName	TableName	IndexName	IndexType	AvgPageFragmentation	PageCounts
AnalyzeFragmentation	FindAndFixFragmentation	CL_FindAndFixFragmentation_Index	CLUSTERED INDEX	0.341296928327645	293

At the bottom of the interface, a status bar indicates: 'Query executed successfully.' and 'AKMEHTA|SQL2008 (10.0 RTM) akmehta (54) AnalyzeFragmentation | 00:00:00 | 1 rows'.

You can see that the value for AvgPageFragmentation is back to 0.341296928327645, which means the fragmentation is same as it was when we began this exercise. And at the same time the value for PageCounts is back to 293. This proves that using REBUILD Index operation is better than REORGANIZE Index operation.

Reorganize Index

Reorganize Index uses minimal system resources and it is performed online. The biggest advantage is it does not require locks for long time therefore it does not block updates or other user queries. If the index fragmentation ranges in between 5% to 30% then it is better to perform Reorganize Index.

Rebuild Index

Rebuild Index basically drops and recreates the index; this is by far the best approach. If the index fragmentation is greater than 30% then the best strategy will be to use Rebuild Index instead of Reorganize Index.

Conclusion

Database Administrators should always make sure that fragmentation of indexes is handled on time. If the indexes are fragmented then the query response will not only be very slow; the data storage will also require more disk space. In this case study you have seen an example where the clustered index gets fragmented over time and the steps which you need to perform to resolve index fragmentation issues.

Case study 3: How to Identify Missing Indexes Using SQL Server DMVs

It has been always challenging for database administrators to identify indexes that are missing on a table. In SQL Server 2005, Microsoft introduced Dynamic Management Views (DMVs). The role of DMVs is to return SQL Server state information; which can be used by database administrators and database developers to monitor the health of an SQL Server Instance and identify potential performance issues. DMVs reflect all the activities on the instance of SQL Server since the last restart. Unfortunately in SQL Server editions prior to SQL Server 2005 there is no easy way to identify missing indexes on a table. In SQL Server 2000 the only way to identify if an index needs to be created is to capture a workload in SQL Profiler and then run it against the Index Tuning Wizard. However, in order to access the information stored in DMVs you need to have VIEW SERVER STATE permissions. Only members of the sysadmin fixed server role can grant VIEW SERVER STATE permissions to other users. In this case study example demonstrating how to identify missing indexes on a table is provided.

The Dynamic Management Views (DMV) which can be used to identify missing indexes on a table are:

- sys.dm_db_missing_index_details
- sys.dm_db_missing_index_group_stats
- sys.dm_db_missing_index_groups
- sys.dm_db_missing_index_columns

Below is an example explaining how to analyse missing indexes on a table using Dynamic Management Views. This example uses the Sales.Store table which is available in AdventureWorks database.

Identify Existing Indexes on a Table

Execute the below TSQL to identify existing indexes on Sales.Store table.

```

USE AdventureWorks
GO
sp_helpindex [Sales.Store]
GO

```

index_name	index_description	index_keys
AK_Store_rowguid	nonclustered, unique located on PRIMARY	rowguid
IX_Store_SalesPersonID	nonclustered located on PRIMARY	SalesPersonID
PK_Store_CustomerID	clustered, unique, primary key located on PRIMARY	CustomerID

Query executed successfully. AkMEHTA\SQL2008 (10.0 RTM) almehta (51) AdventureWorks 00:00:00 3 rows

Sample User Query Executed against Sales.Store Table of AdventureWorks

Next step will be to execute the below mentioned TSQL code against AdventureWorks database.

```

USE AdventureWorks
SELECT Name,CustomerID,ModifiedDate FROM Sales.Store WHERE (Name='Sharp Bikes'
and CustomerID <> " AND ModifiedDate > '2004-10-01')

```

Once the above query is run, the relevant information related to missing index will be gathered by dynamic management views. Now to examine the information that is gathered by missing index DMV's one by one.

Analysis Information Captured by sys.dm_db_missing_index_details DMV

Analyse the information gathered by sys.dm_db_missing_index_details DMV by executing the TSQL code shown below.

```
SELECT * FROM sys.dm_db_missing_index_details
```

index_handle	database_id	object_id	equality_columns	inequality_columns	included_columns	statement
1	1	2130106629	[Name]	[CustomerID], [ModifiedDate]	NULL	[AdventureWorks].[Sales].[Store]

The important columns to look are equality_columns, inequality_columns, included_columns and statement. Below is a brief explanation of the valuable information which these columns contain.

- equality_columns:- The equality_columns = [Name] means within the sample user query [Name] column was used in there WHERE clause and with an equal operator, SQL Server query optimizer is informing us that [Name] column is a good candidate for creating a new index.
- inequality_columns:- The inequality_columns = [CustomerID] and [ModifiedDate] means within the sample user query [CustomerID] and [ModifiedDate] columns were used in the WHERE clause, however they used operators other than the equal operator. In the sample query [CustomerID] was using "<>" NOT equal operator and [ModifiedDate] is using ">" greater than operator.
- included_columns:- The included_columns = "NULL" means when creating a new index on [Name] column, there is no need to add any other column of the Sales.Store table as an included column. If there are values displayed then those columns can be added as included columns when creating an index; this will cover the query thereby improving performance.
- statement:- This column has the value [AdventureWorks].[Sales].[Store] which informs DBA against which table the query was executed.

Analysis Information Captured by sys.dm_db_missing_index_group_stats DMV

Below is an analysis of the information gathered by sys.dm_db_missing_index_group_stats DMV after executing the TSQL query:

```
SELECT          *          FROM          sys.dm_db_missing_index_group_stats
```

group_handle	unique_compiles	user_seeks	user_scans	last_user_seek	last_user_scan	avg_total_user_cost	avg_user_impact	
1	2	1	21	0	2009-01-11 00:13:31.560	NULL	0.0787440540740741	95.09

The important columns to focus on are unique_compiles, user_seeks, last_user_seek, avg_total_user_cost and avg_user_impact. A brief explanation of the information contained in each column follows:

- unique_compiles:- The value 1 indicates that the query has been compiled once after the SQL Server was restarted, meaning there was no recompilation by the sample user query even though the query was executed 21 times (value obtained from user_seeks).
- user_seeks:- The value 21 represents that the user query was executed 21 times, once the SQL Server was started.
- last_user_seek:- This column contains data and time data from when the user query was last executed.
- avg_total_user_cost:- This column represent the average total user cost each time when the user query was executed.
- avg_user_impact:- This column represents the value in percentages. It informs us the amount of improvement which you can get if the index is created.

Analysis Information Captured by sys.dm_db_missing_index_groups DMV

Below is an screenshot of the information gathered by sys.dm_db_missing_index_groups DMV following execution of the TSQL query:

```
SELECT * FROM sys.dm_db_missing_index_groups
```

	index_group_handle	index_handle
1	2	1

Analysis Information Captured by sys.dm_db_missing_index_columns DMV

Below is an analysis of the information gathered by sys.dm_db_missing_index_columns DMV following execution the TSQL query:

```
SELECT * FROM sys.dm_db_missing_index_columns(1)
```

	column_id	column_name	column_usage
1	2	Name	EQUALITY
2	1	CustomerID	INEQUALITY
3	6	ModifiedDate	INEQUALITY

The important columns to pay attention to are column_name and column_usage. A brief explanation of the information each column contains follows:

- column_name:- This column represent the name of the column within the user table, in our example the table used is Sales.Store of AdventureWorks database and the columns used are Name, CustomerID and ModifiedDate.
- column_usage:- This column represents how the user column is used within the user query. It displays which operator was used against each of the columns.

Query to Identifying Missing Indexes

The below query can be used to identify all the missing indexes on a user table. This query uses all the missing index DMV's and provides a consolidated result. However execute the below query only when the server has cached most of the query plans. This is very important to get a clear result of the missing indexes. SQL Server can store upto a maximum of 500 missing index information once the SQL Server is started within the DMV's. The missing index feature turned on by default in an SQL Server instance.

```
SELECT
statement AS [database.schema.table],
column_id , column_name, column_usage,
migs.user_seeks, migs.user_scans,
migs.last_user_seek, migs.avg_total_user_cost,
```

```

migs.avg_user_impact
FROM sys.dm_db_missing_index_details AS mid
CROSS APPLY sys.dm_db_missing_index_columns (mid.index_handle)
INNER JOIN sys.dm_db_missing_index_groups AS mig
ON mig.index_handle = mid.index_handle
INNER JOIN sys.dm_db_missing_index_group_stats AS migs
ON mig.index_group_handle=migs.group_handle
ORDER BY mig.index_group_handle, mig.index_handle, column_id
GO

```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "Identify Missing Indexes.sql" is open, displaying the T-SQL code provided above. Below the code, the "Results" tab is selected, showing a table with the following data:

database schema table	column_id	column_name	column_usage	user_seeks	user_scans	last_user_seek	avg_total_user_cost	avg_user_impact
[AdventureWorks].[Sales].[Store]	1	CustomerID	INEQUALITY	21	0	2009-01-11 00:13:31.560	0.0787440540740741	95.09
[AdventureWorks].[Sales].[Store]	2	Name	EQUALITY	21	0	2009-01-11 00:13:31.560	0.0787440540740741	95.09
[AdventureWorks].[Sales].[Store]	6	ModifiedDate	INEQUALITY	21	0	2009-01-11 00:13:31.560	0.0787440540740741	95.09

The status bar at the bottom indicates "Query executed successfully." and "AKMEHTA(SQL2008 (10.0 RTM)) almehta (51) AdventureWorks 00:00:00 3 rows".

The next step is to create a nonclustered index for name column of Sales.Store table of AdventureWorks database as suggested by missing index DMV's.

Create Non Clustered Index for Name Column

Execute the below TSQL code to create a non clustered index on Sales.Store table.

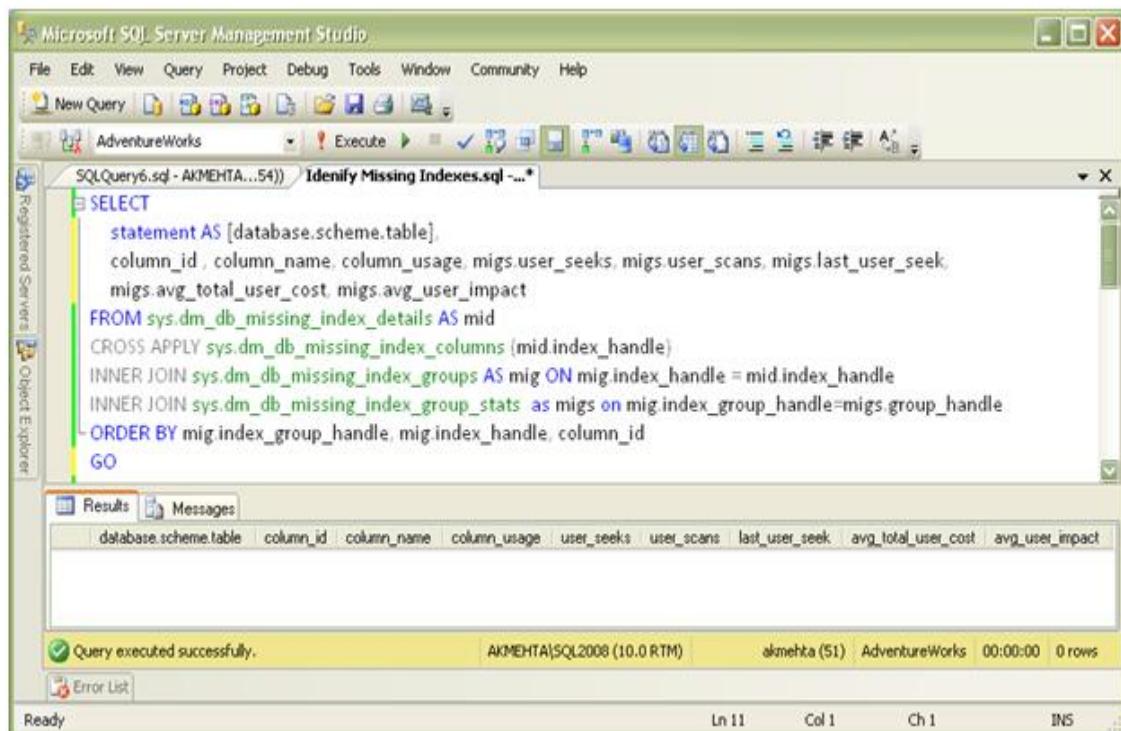
```

USE [AdventureWorks]
GO
CREATE NONCLUSTERED INDEX [IX_Store_Name] ON [Sales].[Store]
(
    [Name] ASC
)

```

```
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
```

Once the nonclustered index is created, the information that was available in the DMV has disappeared, shown in the snippet below.



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'Identify Missing Indexes.sql - ...' displays the following T-SQL code:

```
SELECT
    statement AS [database.schema.table],
    column_id, column_name, column_usage, migs.user_seeks, migs.user_scans, migs.last_user_seek,
    migs.avg_total_user_cost, migs.avg_user_impact
FROM sys.dm_db_missing_index_details AS mid
CROSS APPLY sys.dm_db_missing_index_columns (mid.index_handle)
INNER JOIN sys.dm_db_missing_index_groups AS mig ON mig.index_handle = mid.index_handle
INNER JOIN sys.dm_db_missing_index_group_stats AS migs ON mig.index_group_handle = migs.group_handle
ORDER BY mig.index_group_handle, mig.index_handle, column_id
GO
```

The 'Results' tab shows a table structure with columns: database.schema.table, column_id, column_name, column_usage, user_seeks, user_scans, last_user_seek, avg_total_user_cost, and avg_user_impact. Below the table, a message indicates 'Query executed successfully.' and shows statistics: AKMEHTA|SQL2008 (10.0 RTM), akmehta (51), AdventureWorks, 00:00:00, 0 rows.

Disadvantage of having too many indexes on a table

- Insert, Update and Delete operations will become very slow if there are many indexes created on a table. This occurs when the Insert, Update or a Delete operation against a table results in all the indexes being updated, reducing query performance.
- Indexes are stored on disk and the amount of space required by the index depends on the size of database table, and the number and type of columns used within the index definition.
- A greater number of indexes result in more disk space being required to store them.

Limitations of Missing Index Feature

- A DMV can store information from a maximum of 500 missing indexes.
- Unable to provide recommendations for clustered, indexed views and partitioning.

- Once the SQL Server is restarted all the information related to missing indexes is lost.
To keep the information for later use, the DBA needs to backup all the data available within all the missing index DMV prior to the restart of SQL Server.

Conclusion

Database Administrators need to analyse the impact of an index created for Insert, Update and Delete operations before accepting recommendations given by the missing index DMVs.

Chapter – 24

Performance Tuning

Introduction to Performance Troubleshooting Methodology:

SQL Server is an enterprise-level database server that has been used by organizations to run their mission-critical applications worldwide. These applications impose the toughest requirements in terms of availability, performance, and scalability. Very few enterprise workloads, if any, have requirements that exceed these. Microsoft SQL Server 2005 has successfully met or exceeded the requirements of these workloads under demanding conditions.

You may wonder what we mean by performance because the word may mean different things to different people. For the discussions here, we will focus on the following three terms.

- **Response time** Refers to the interval between the time when a request is submitted and when the first character of the response is received.
- **Throughput** Refers to the number of transactions that can be processed in a fixed unit of time.
- **Scalability** Refers to how the throughput and/or the response time changes as we add more hardware resources. In simple terms, scalability means that if you are hitting a hardware bottleneck, you can alleviate it simply by adding more resources.

Factors That Impact Performance

Most users perceive the performance of their SQL Server based on the responsiveness of their application. While SQL Server itself does play a part in application responsiveness, before we can isolate SQL Server as the source of a performance issue, we first need to understand the factors that impact the performance of your application. We will look into these factors for completeness before we switch our focus to troubleshooting performance problems in SQL Server.

At a high level, many factors can impact the performance and scalability that can be achieved in your application. We will be looking at the following:

- Application Architecture
- Application Design
- Transactions and Isolation Levels
- Transact-SQL Code
- Hardware Resources
- SQL Server Configuration

In this I am explaining about the Transactions and Isolation levels, Hardware resources and SQL server configuration

Transactions and Isolation Levels

Applications interact with SQL Server using one or more transactions. A transaction can be started explicitly by executing the BEGIN TRANSACTION Transact-SQL statement or implicitly, by SQL Server, for each statement that is not explicitly encapsulated by the transaction. Transactions are very fundamental to database systems. A transaction represents a unit of work that provides the following four fundamental properties:

- Atomicity Changes done under a transaction that are either all commit or are rolled back. No partial changes are allowed. It is an all-or-nothing proposition.
- Consistency Changes done under a transaction database from one consistent state to another. A transaction takes a database from one consistent state to another.
- Isolation Changes done by a transaction are isolated from other concurrent transactions until the transaction commits.
- Durability Changes done by committed transactions are permanent.

Hardware Resources

Hardware resources are the horsepower you need to run your application. It will do you no good if you have a well-designed application, but it is running on hardware that is not on a par with the demands of the workload. Most often, an application is tested in a small-scale environment with simulated workload. While this does serve a useful purpose, clearly it is not a replacement for measuring the performance of the workload and the hardware it actually runs on in production. So when your application hits performance issues, it can be the hardware, however, that is not necessarily so. Any hardware resource (CPU, I/O, memory, or network) that is pushed beyond its operational capacity in any application tier will lead to a slowdown in your application. Note, too, that hardware bottlenecks may also be caused by poor application designs, which ultimately need to be addressed. In such cases, upgrading hardware is a short-term solution at best.

SQL Server Configuration

SQL Server is designed to be self-tuning to meet the challenges of the workload. In most cases, it just works well with out-of-the-box configuration settings. However, in some cases, you may need to tweak configuration parameters for maximum performance. Inside SQL Server 2005: The Storage Engine (Microsoft Press, 2006) has some additional details about configuration options. In this section, we will list the options that you will most likely need to monitor to track down performance issues. The relevant configuration options can be considered to be either CPU-related or memory-related options.

CPU-Related Configuration Options

The CPU-related configuration options are used, for example, to control the number of CPUs or sockets that can be used by a SQL Server instance, maximum degree of parallelism, and the number of workers. Some commonly used options in this category include the following:

- Affinity Mask This option can be used to control the mapping of CPUs to the SQL Server process. By default, a SQL Server uses all processors available on the Server box. A general recommendation is not to use affinity mask option, as SQL Server performs

best in the default setting. You may, however, want to use this option under two situations.

- First, if you are running other applications on the box, the Windows operating system may move around process threads to different CPUs under heavy load. By using affinity masks, you can bind each SQL Server scheduler to its own CPU. This can improve performance by eliminating thread migration across processors, thereby reducing context switching.
- Second, you can use this parameter to limit the number of CPUs on which a SQL Server can run. This is useful if you are running multiple SQL Server instances on the same Server box and want to limit CPU resources taken by each SQL Server and to minimize their interference.
- Lightweight Pooling When this configuration is enabled, SQL Server makes use of Windows fibers. A worker can map to a Windows thread or to a fiber. A fiber is like a thread, but it is cheaper than normal thread because switching between two workers (that is, fiber threads) can be done in user mode instead of kernel mode. So if your workload is experiencing a CPU bottleneck with significant time spent in kernel mode and in context switching, you may benefit by enabling this option. You must test your workload with this option before enabling it in the production system, as more often than not this option may cause performance regression. You should also keep in mind that CLR integration is not supported under lightweight pooling.
- Max Worker Threads You can think of workers as the SQL Server threads that execute user or batch requests. A worker is bound to a batch until it completes. So the maximum number of workers limits the number of batches that can be executed concurrently. By default, SQL Server sets the Max Worker Threads as described in the following table.

Number of CPUs	32-bit computer	64-bit computer
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960

Number of CPUs	32-bit computer	64-bit computer
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960

- For most installations, this default setting is fine. Each worker takes 512-KB memory on a 32-bit, 2 MB on X64, and 4 MB on IA64. To preserve memory, the SQL Server starts off with a smaller number of workers. The pool of workers grows or shrinks based on the demand. You may want to change this configuration under two conditions. The first is if you know that your application uses a smaller number of workers. By configuring it to a lower number, the SQL Server does not need to reserve memory for the maximum

number of workers. The second is if you have many long-running batches (presumably involving lock waits), such that the number of workers needed may exceed the default configuration. This is not a common case, and you may want to look at your application design to analyze this.

- **Max Degree of Parallelism** This configuration parameter controls the maximum number of processors or cores that can be deployed to execute a query in parallel. Parallel queries provide better response time but take more CPU resources. You need to look into this configuration parameter if you are encountering CPU bottleneck, described later in this chapter.

Memory-Related Configuration Options

The memory-related configuration options are used to control the memory consumed by SQL Server. Some of the commonly used configuration options in this category include the following:

Max and Min Server Memory This is perhaps the most critical of the configuration options, especially in 32-bit configurations, from a performance perspective. This configuration parameter is often confused with the total memory configured for a SQL Server, but it is not the same. It represents the configured memory for the buffer pool. SQL Server, or any database server for that matter, is a memory-hungry application. You want to make as much memory available for SQL Server as possible. The recommendation on 64 bits is to put an upper limit in place to reserve memory for the OS and allocations that come from outside the Buffer Pool. You will also need to cap the memory usage by SQL Server when other applications (including other instances of SQL Server) are running on the same Server box.

AWE Enabled On a 32-bit box, the SQL Server process can only address 2 GB of virtual memory, or 3 GB if you have added the /3 GB parameter to the boot.ini file and rebooted the computer, allowing the /3 GB parameter to take effect. If you have physical memory greater than 4 GB and you have enabled this configuration option, the SQL Server process can make use of memory up to 64 GB normally, and up to 16 GB if you have used /3 GB parameter. Additionally, the SQL Server process requires Lock Pages in Memory privilege in conjunction with AWE-Enabled option. There are some restrictions in terms of the SQL Server SKU and the version of the Windows operating system under which you are running. You should be aware of certain key things when using the AWE option. First, though it allows the SQL Server process to access up to 64 GB of memory for the buffer pool; the memory available to query plans, connections, locks, and other critical structures is still limited to less than 2 GB. Second, the AWE-mapped memory is non pageable and can cause memory starvation to other applications running on the same Server box. Starting with SQL Server 2005, the AWE memory can be released dynamically, but it still cannot be paged out. SQL Server may release this memory in response to physical memory pressure. Third, this option is not available on a 64-bit environment. However, if the SQL Server process has been granted Lock Pages in Memory privilege, the buffer pool will lock pages in memory and these pages cannot be paged out.

Database Tuning Advisor - DTA

The Database Engine Tuning Advisor is a new tool in Microsoft SQL Server 2005 that enables you to tune databases for improved query processing. Database Engine Tuning Advisor examines how queries are processed in the databases you specify and then it recommends how

you can improve query processing performance by modifying physical design structures such as indexes, indexed views, and partitioning. It replaces the Index Tuning Wizard from Microsoft SQL Server 2000, and offers many new features.

Perhaps an index was created using the wrong columns, or maybe users have started querying different data over time, which would require the creation of new indexes. If any of this is true, your databases need tuning. To do that, you need to use the *Database Engine Tuning Advisor*. Before you can run the Database Engine Tuning Advisor, you need to create a *workload file*. You get this by running and saving a trace in Profiler (usually by creating a trace with the Tuning template) or the query file(.sql file) or XML format file. Maximum time profiler is considered to be the best, since it will provide you the actual work flow. If you are using profiler then it is best to get this workload during times of peak database activity to make sure you give the advisor an accurate load. First you need to create a workload file to use with the advisor as said above.

Here in this case study I'm going to check DTA with Adventureworks db and ProductDescription table. I've deleted the index and primary key to check whether DTA is capable to detect this. After this step I've executed a select query on the above table and saved the query as a SQL file. I've used the above file in DTA as workload file. When DTA finishes its work, it perfectly detects the missing index and recommended to create the same.

Deleting Index & primary key on ProductDescription table:

I've deleted the primary key and index "PK_ProductDescription_ProductDescriptionID" available on the table ProductDescription.

Creating workload file

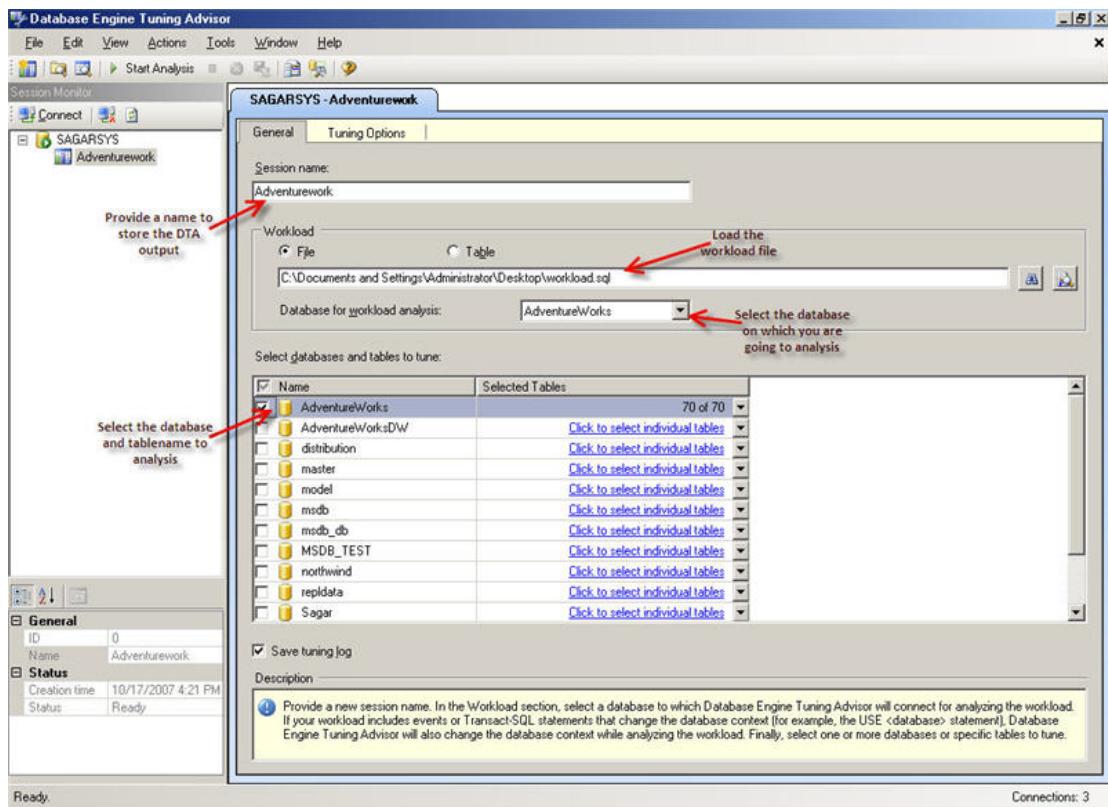
Save the below query as SQL file. I've saved it as workload.sql.

```
select ProductDescriptionID,ModifiedDate from Production.ProductDescription  
where Description like '%frame%'
```

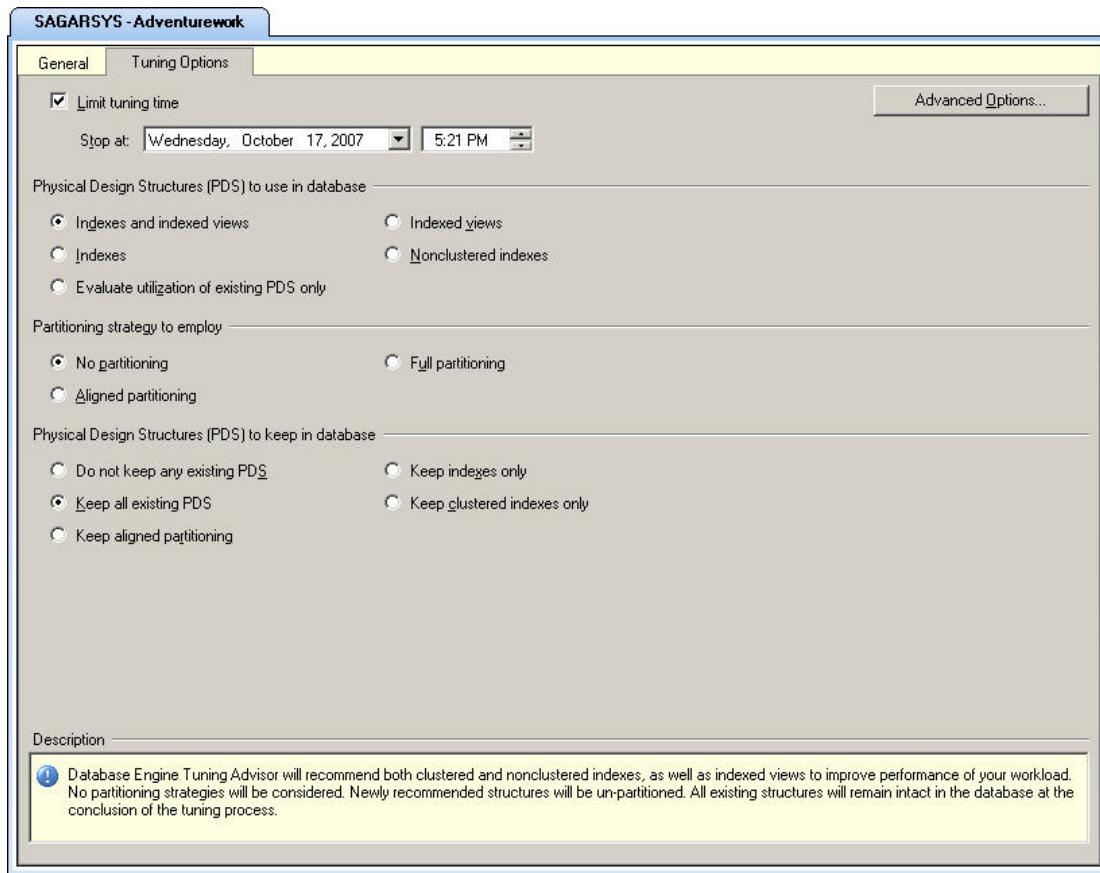
You can also use profiler (Select tuning template) to get workload file.

Using DTA

- *) Goto Start --> Programs --> Microsoft SQL Server 2005 --> Performance Tools --> Database Engine Tuning Advisor
- *) Connect to the server by correct authentication mode
- *) In the left pane you can see "Session Monitor" where the connected servername will be displayed
- *) Right click on the server and click on New session, you will get a window as below



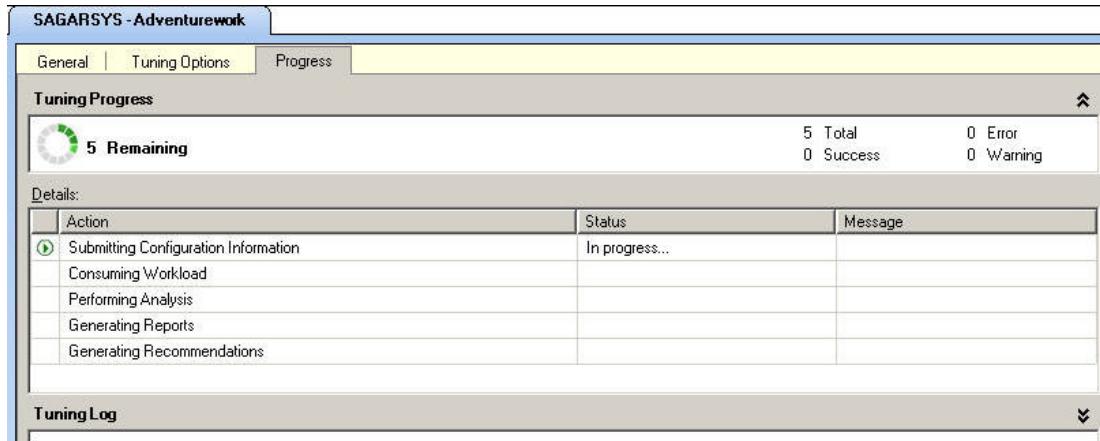
- *) Provide the sessionname, databasename and workload file as show in the figure above**
- *) Then goto "Tunning Options" tab and you can see the default settings to create index or partitions as per recommendations as show below**



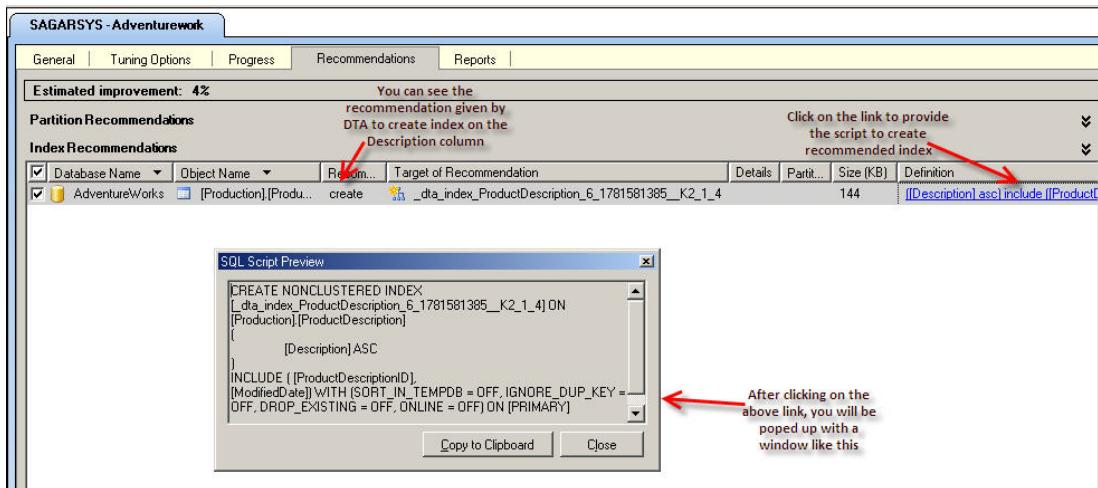
*) Once its done, then click on the "start analysis" button as shown below



*) Once analysis started you will see the below window



*) As I told earlier once the analysis completed it has recommended to create a index on the description column, you can also get the index creation script from the result itself as show below



Best Practices on Performance Tuning

- Regularly monitor your SQL Servers for blocked transactions.
- Regularly monitor system performance using System Monitor. Use System Monitor for both real-time analysis and for historical/baseline analysis.
- If running SQL Server 2005, SP2 or later, install the free SQL Server Performance Dashboard. It can be used for real-time monitoring and performance troubleshooting.
- Regularly monitor activity using Profiler. Be sure that traces are taken during the busiest times of the day so you get a more representative trace of what is going on in

each server. When running the Profiler, do not collect more data than you need to collect.

- Perform performance monitoring from a computer that is not the SQL Server you are monitoring. Run monitoring tools on a separate desktop or server.

Case study1: How to Collect Profiler Data for Correlation Analysis

While it is possible to correlate most Profiler events to most Performance Monitor counters, the area of greatest correlation is between Profiler Transact-SQL events and Performance Monitor counters that indicate resource bottlenecks.

This is where I focus my efforts, and the following sections describe how I collect Profiler data for correlation with Performance Monitor. As always, feel free to modify my suggestions to suit your own needs. The key, as always when using Profiler, is to capture only those events and data columns you really need in order to minimize the workload on your production server when the trace is running.

Events and Data Columns

This template collects data on the following events:

- RPC:Completed
- SP:StmtCompleted
- SQL:BatchStarting
- SQL:BatchCompleted
- Showplan XML

In addition, I include these data columns:

- Duration
- ObjectName
- TextData
- CPU
- Reads
- Writes
- IntegerData
- DatabaseName
- ApplicationName
- StartTime
- EndTime
- SPID
- LoginName
- EventSequence
- BinaryData

Note that in order to perform an accurate correlation between Profiler and Performance Monitor data, you need to capture both the StartTime and EndTime data columns as part of your trace.

Filters

The only filter I create is based on **Duration**, because I want to focus my efforts on those SQL Statements that are causing the most problems. Selecting the ideal Duration for the filter is not always easy. Generally, I might initially capture only those events that are longer than 1000 milliseconds in duration. If I find that there are just too many events to easily work with, I might "raise the bar" to 5000 or 10000 milliseconds. You will need to experiment with different durations to see what works best for you.

In the example for this article, I use 1000 milliseconds. I don't filter on DatabaseName, or any other data column, because I want to see every statement for the entire SQL Server instance. Performance Monitor counters measure the load on an instance as a whole, not just the load on a single database.

Ordering and Grouping Columns

I don't do any grouping, but I generally order the data columns in an order that works well for me. You can perform any grouping or aggregation you want, but it won't affect the correlation analysis, and so I generally omit it.

How to Collect Performance Monitor Data for Correlation Analysis

I assume you know the basics of using Performance Monitor, but in case you don't know how to create logs, I will describe in this section how to set up a Performance Monitor log to capture counter activity, which can then be correlated with Profiler trace events.

NOTE:

Performance Monitor comes in different versions, depending on the operating system, and the way logs are created differs from version to version. In this example, I am using the version of Performance Monitor that is included with Windows Vista and Windows 2008.

The activity data collected by Performance Monitor can be displayed "live" on a graph, or you can store it in a log file, using what is called a *user defined data collector set*. In order to correlate Performance Monitor data with Profiler trace data, you must store the activity data in a log file. This log file can then be imported into Profiler for the correlation analysis.

Performance monitor provides a wizard to help you do this, which entails three main steps:

- Creating a new Log file definition
- Selecting Performance Counters
- Creating and saving the Log file

Defining a new Performance Monitor Log File

On starting Performance Monitor, you will see a screen similar to the one shown in Figure 1-1. By default, Performance Monitor operates in "live graphing" mode, which shows the graph being created on the screen in real time.

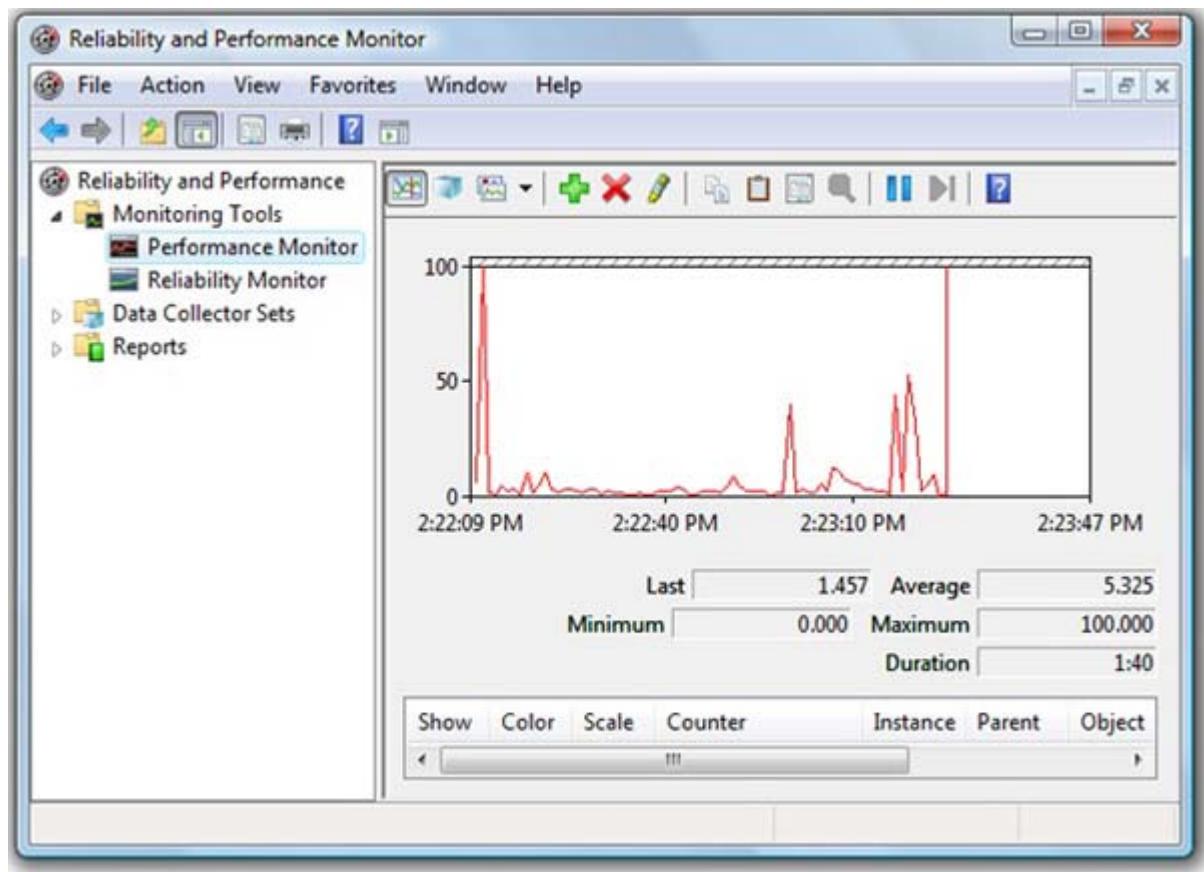


Figure 1-1: Start Performance Monitor. The appearance of the screen will vary somewhat from OS to OS.

In Vista and SQL Server 2008, a Performance Monitor log is referred to as a **Data Collector Set**. To set up a new data collector set (i.e. log file), double-click on "Data Collector Sets" then right-click on "User Defined" and select "New | Data Collector Set", as shown in Figure 1-2:

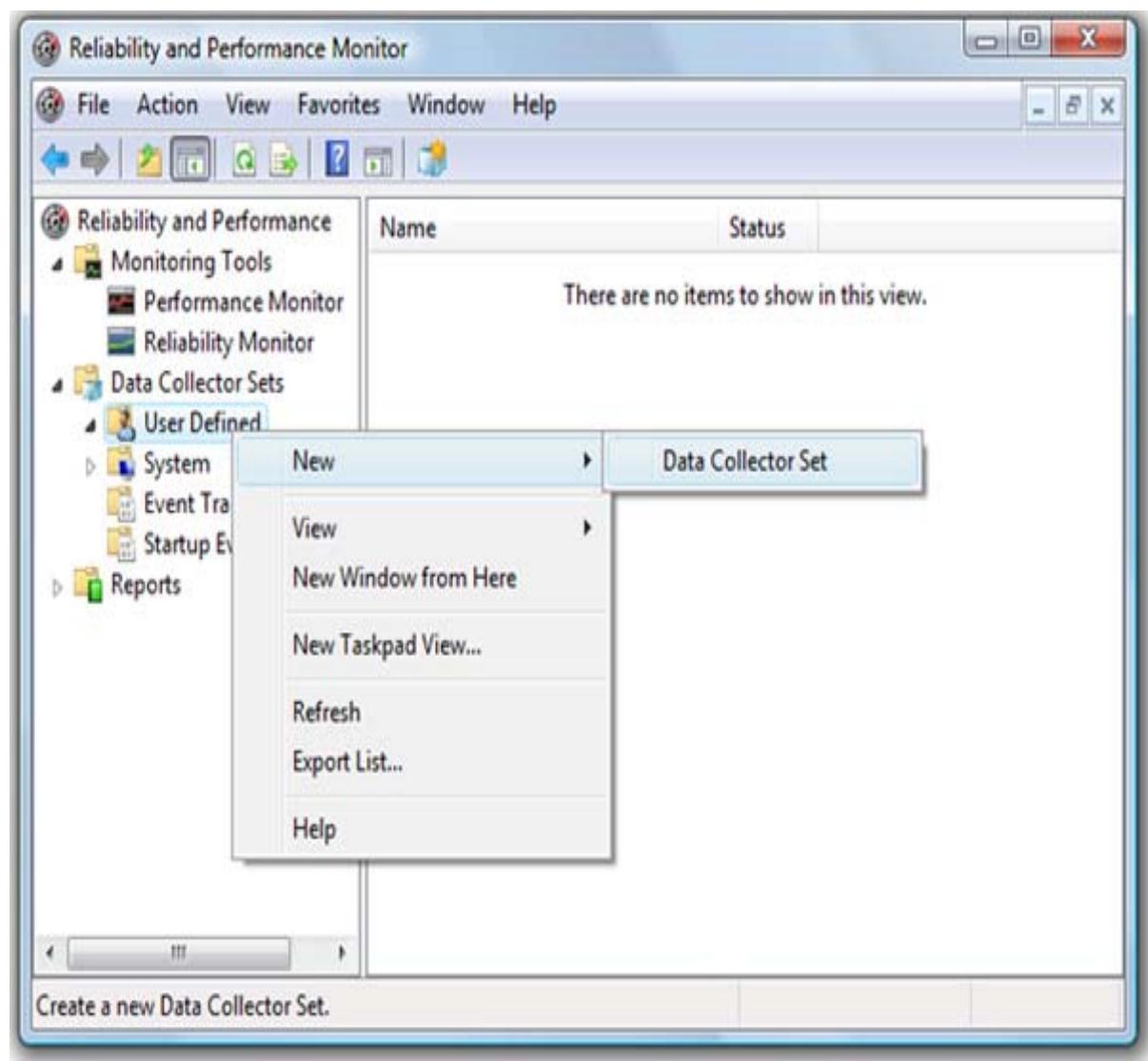


Figure 1-2: You need to create a new "Data Collector Set."

You will be presented with the "Create a new Data Collector Set" screen, as shown in Figure 1-3:

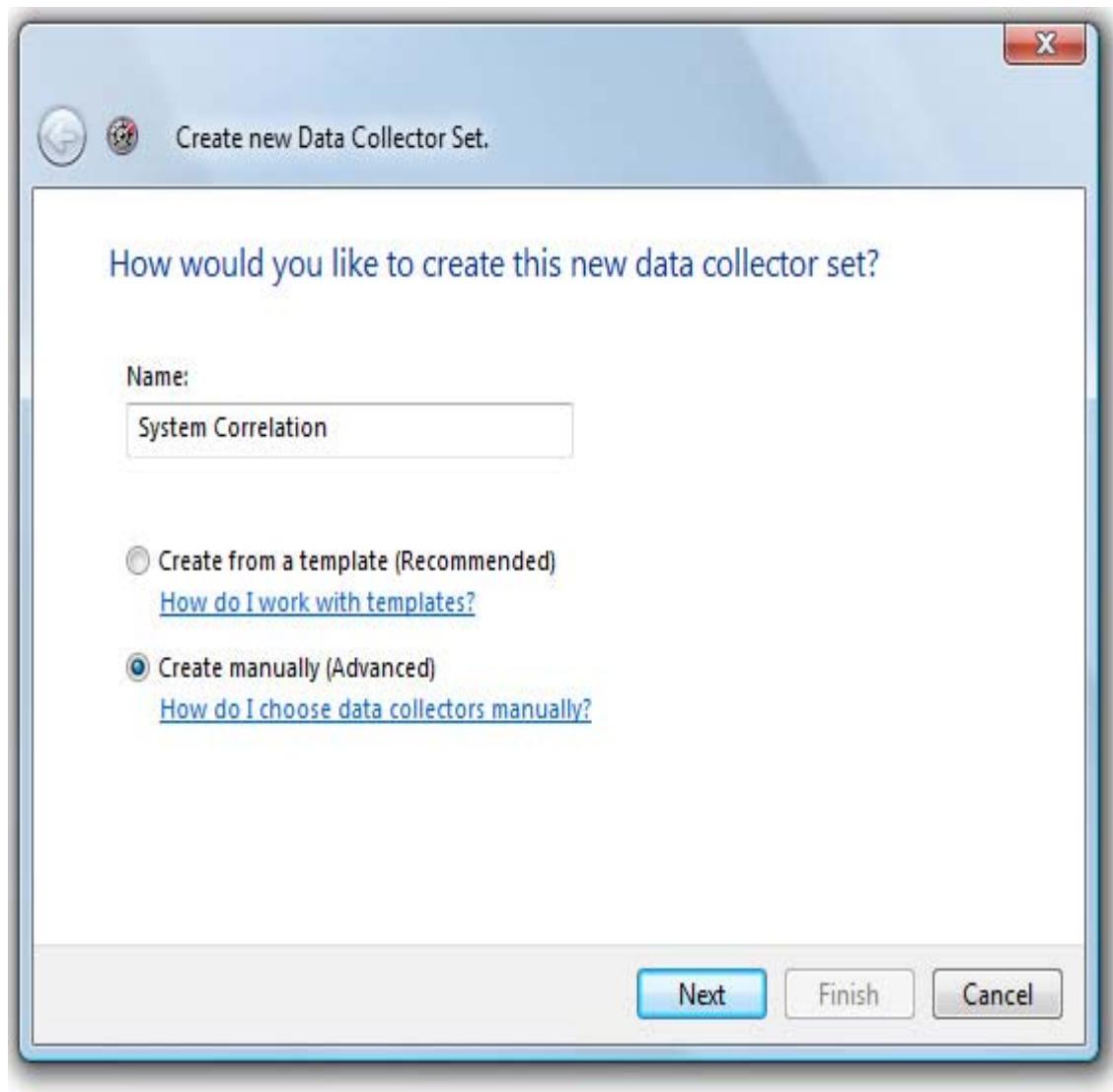


Figure 1-3: Give the Data Collector Set its own name.

Assign the Data Collector Set a name, such as "System Correlation". At the bottom of the screen, select "Create Manually" and click "Next". I recommend that you use the manual option over the template option because you have more flexibility when selecting the events you want to collect. The screen shown in Figure 1-4 appears:

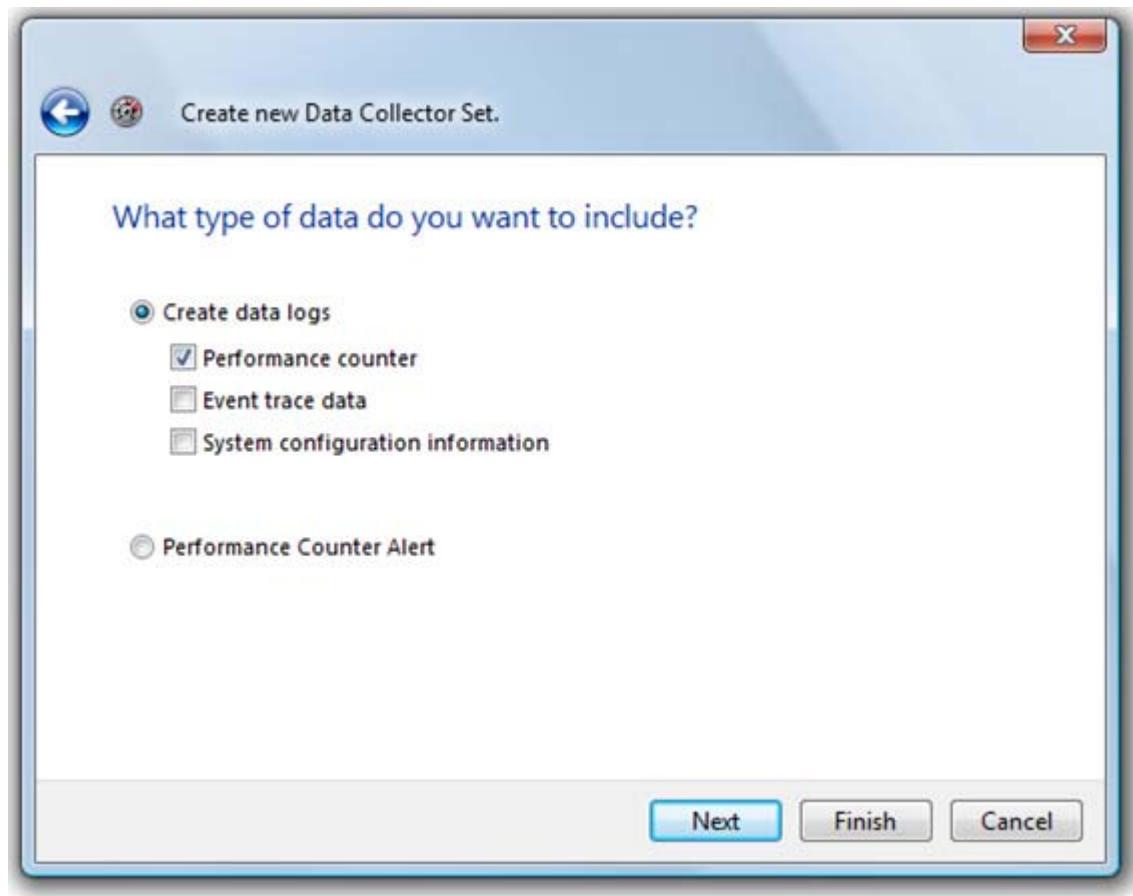


Figure 1-4: You want to create a "Performance Counter" data collector set.

To create our Performance Monitor log, check the box next to "Performance Counter" and click "Next". The other events that can be collected are of no use to us when performing our correlation analysis.

Selecting Performance Counters for the Log File

The next screen in the wizard, shown in Figure 1-5, allows you to select the counters you'd like to record and save in the log file.

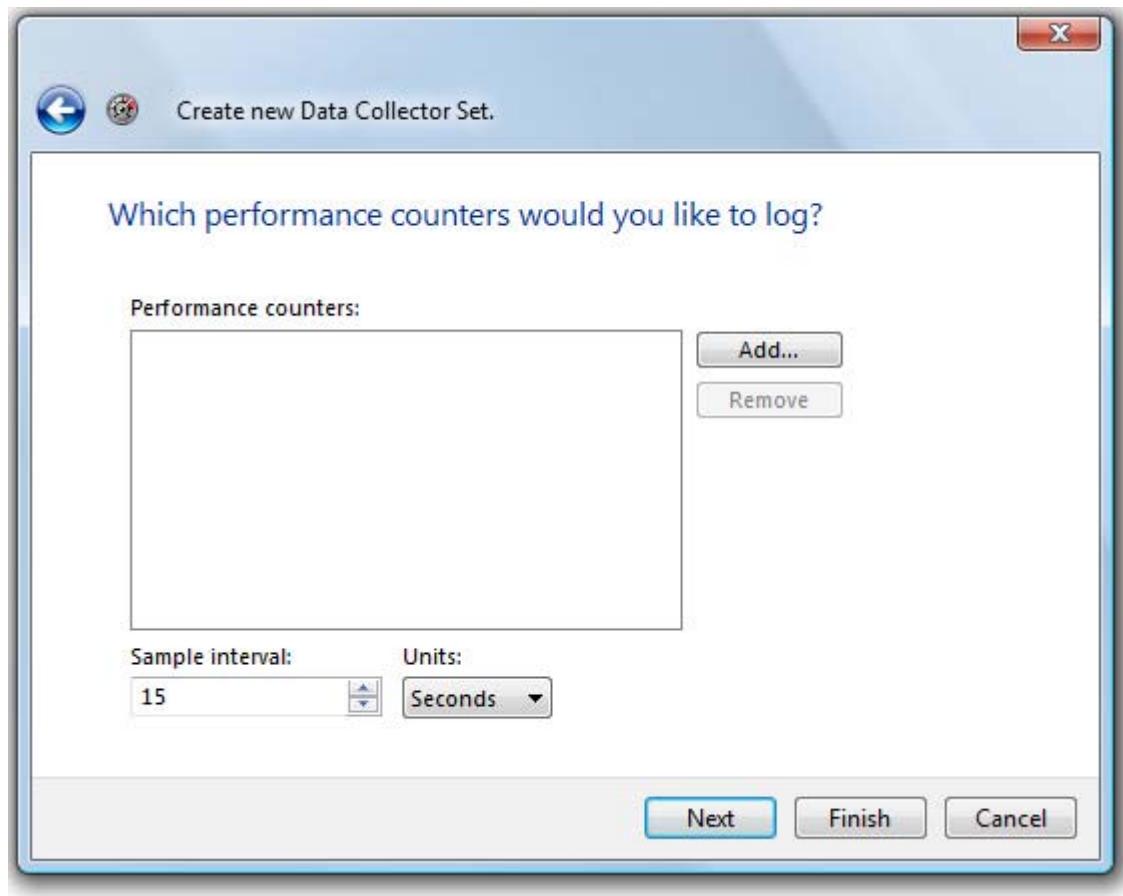


Figure 1-5: You now need to select the Performance Monitor counters you want to capture as part of your log file.

Performance Monitor offers several hundred counters, many more than there are Profiler events. However, you don't want to select more counters than you need, as it will just make correlation analysis that much more difficult. My goal is to select only those Performance Monitor counters I need to identify key CPU, disk I/O, and memory bottlenecks within SQL Server. With this in mind, I generally track two different counters for each of the three key bottleneck areas:

- **LogicalDisk: % Disk Time** – Indicates the activity level of a particular logical disk. The higher the number, the more likely there is an I/O bottleneck. Be sure to select those counters for the logical drives that contain your mdf and ldf files. If you have these separated on different logical disks, then you will need to add this counter for each logical disk.
- **LogicalDisk: Avg. Disk Queue Length** – If a logical disk gets very busy, then I/O requests have to be queued. The longer the queue, the more likely there is an I/O

bottleneck. Again, be sure to select those counters for each logical drive that contains your mdf and ldf files.

- **Memory: Available Mbytes** – Measures how much RAM is currently unused, and so available for use by SQL Server and the OS. Generally speaking, if this drops below 5mb, this is a possible indication of a memory bottleneck.
- **Memory: Pages/sec** – Measures how much paging the OS is performing. A high number may indicate a potential memory bottleneck.
- **Processor: % Processor Time: _Total** – Measures the percentage of available CPUs in the computer that are busy. Generally speaking, if this number exceeds 80% for long periods of time, this may be an indication of a CPU bottleneck.
- **System: Processor Queue Length** – If the CPUs get very busy, then CPU requests have to be queued, waiting their turn to execute. The longer the queue, the more likely there is a CPU bottleneck.

The first two are for LogicalDisk, the second two are for Memory, and the last two (although they have different instance names) are for the Processor. I find that using two counters per area, rather than one, provides just enough information to identify the cause of most bottlenecks. You will probably want to modify the above list to suit your own needs and environment, but it's a good starting point.

Having selected your performance counters, the screen will look similar to Figure 1-6:

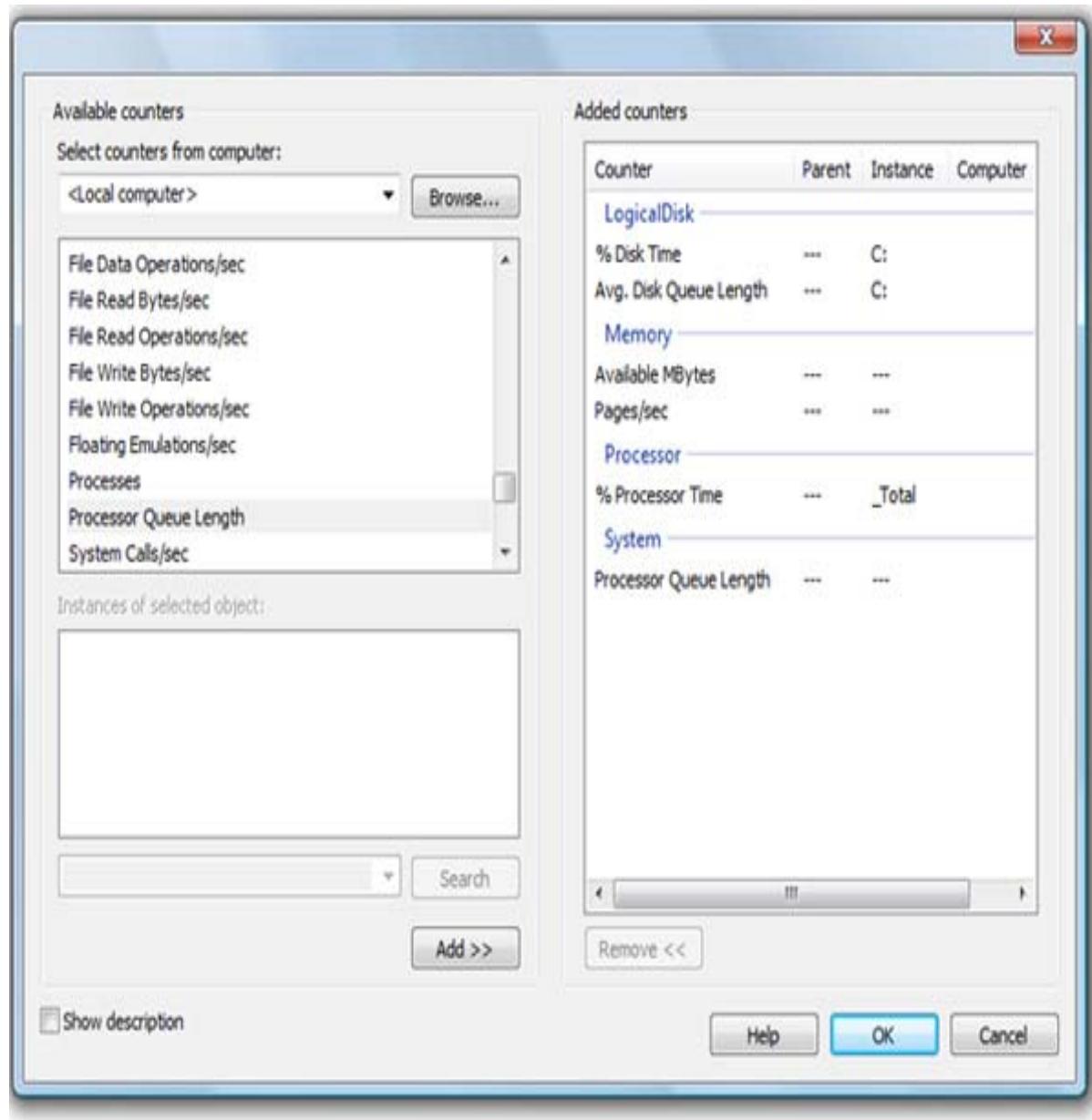


Figure 1-6: Select those counters that best meet your needs.

Click "OK" to proceed, and the screen shown in Figure 1-7 returns:

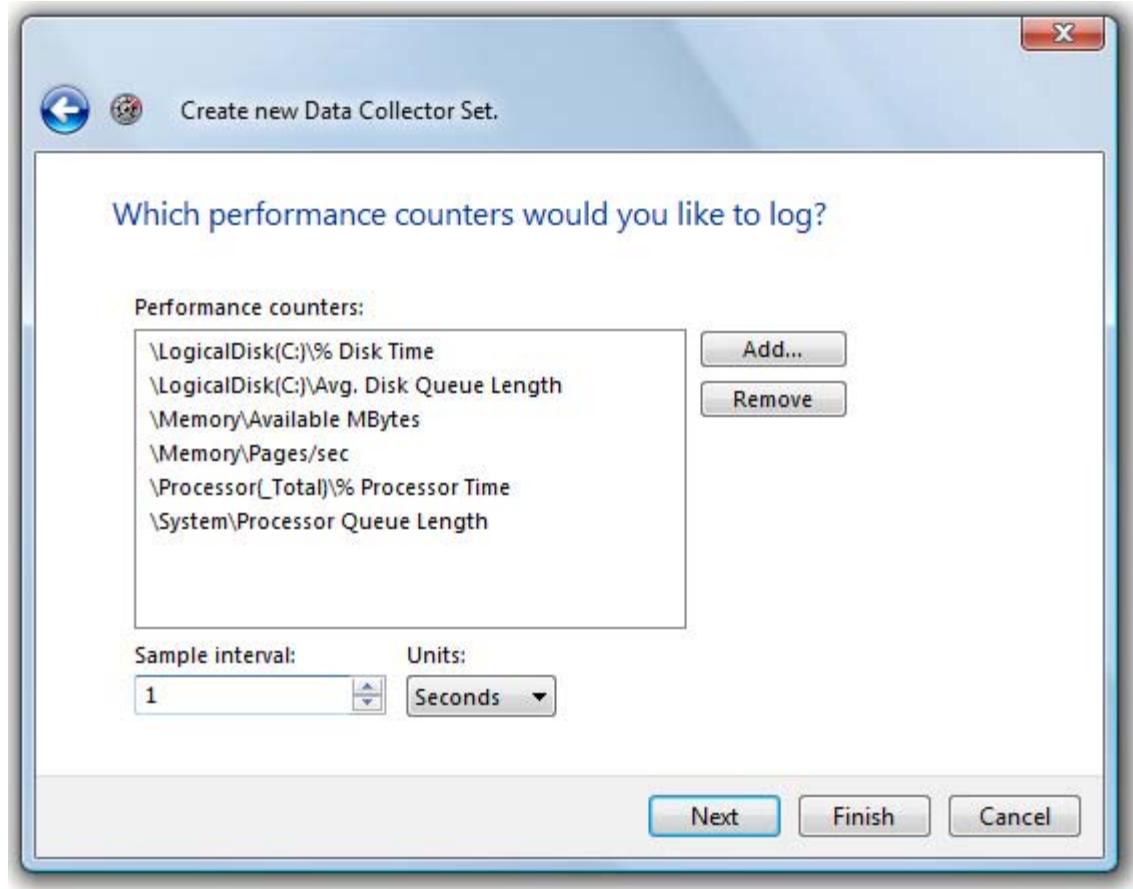


Figure 1-7: Set the "Sample Interval."

The next step is to choose how often Performance Monitor counter data is to be collected. The default value is once every 15 seconds. However, when it comes to performing a Profiler and Performance Monitor correlation analysis, accurate timing is important, so I highly recommend that you select a sample interval of 1 second.

The upside to this is that a 1-second interval will help you to better see the correlation between Profiler events and Performance Monitor counters. The downside is that you will collect a lot of data very quickly. Generally, this is not a problem if you capture a minimum number of counters and don't run your trace for hours at a time.

Creating and Saving the Log File

Once you have entered the sample interval, click "Next", and the screen shown in Figure 1-8 appears:

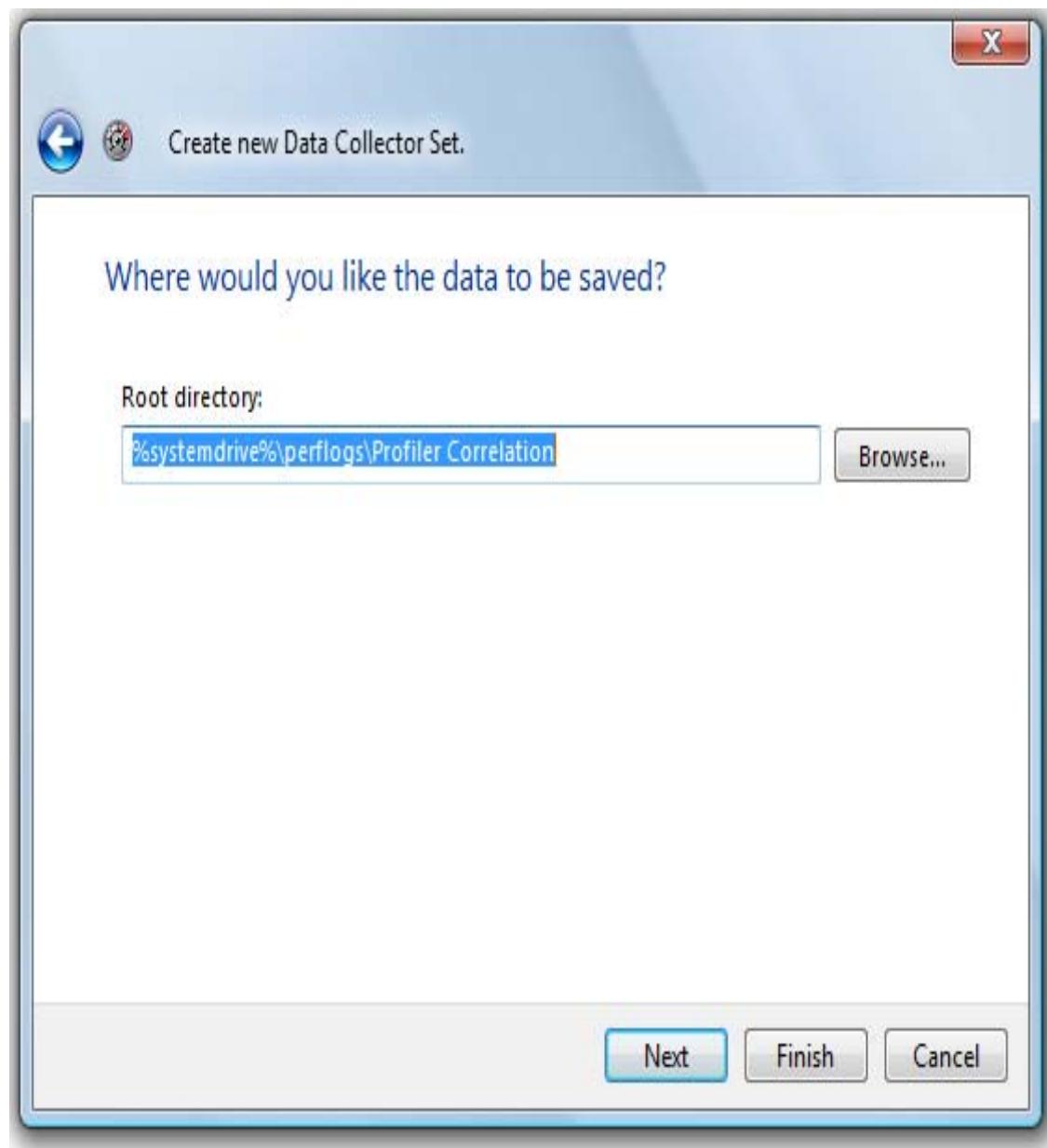


Figure 1-8: Specify where you want Performance Monitor logs to be stored.

Specify where you would like to store the Performance Monitor log. Any place will work, but you don't want to forget this location as you will need to be able to retrieve the log file for importing into Profiler later. Once you have specified the log location, click "Next" to continue and the screen shown in Figure 1-9 appears:

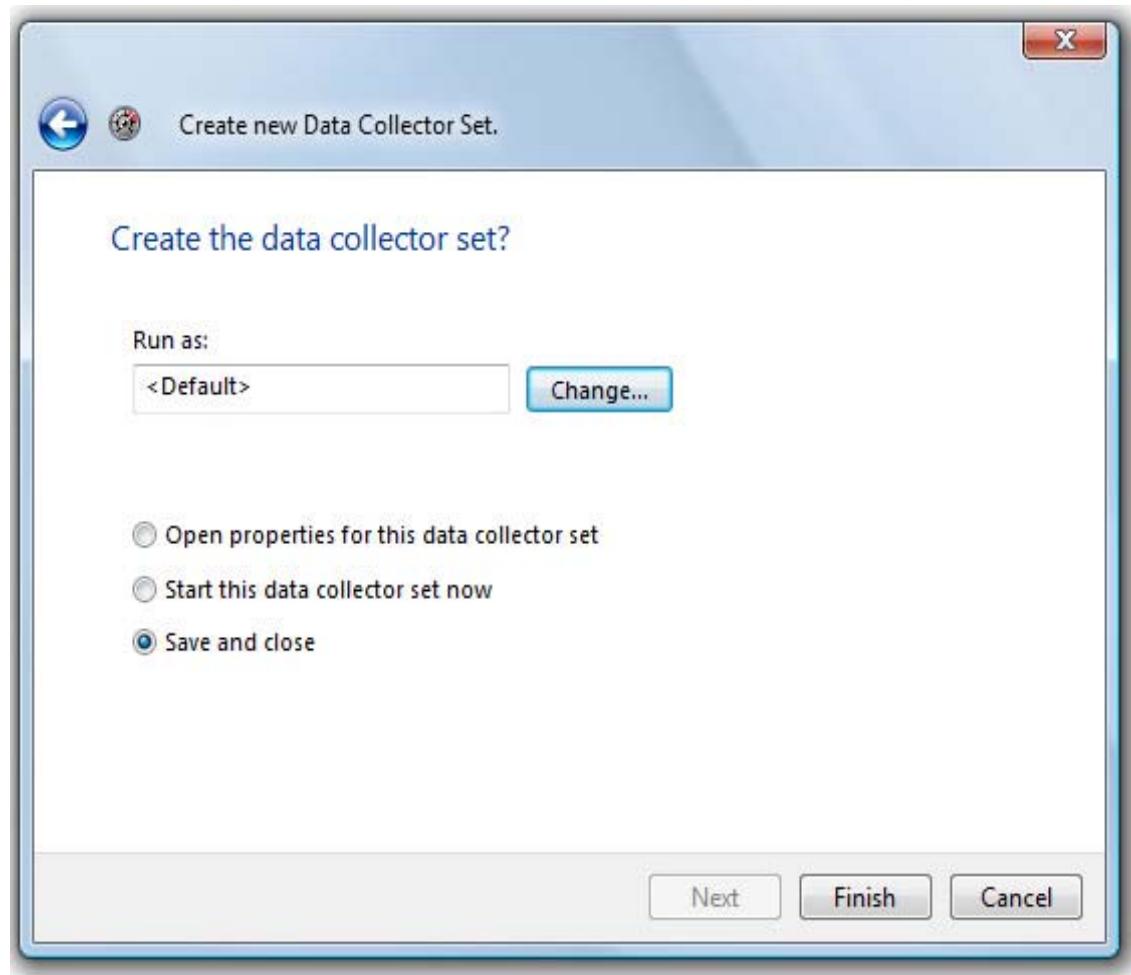


Figure 1-9: You can specify if this data collector set is to be the default set or not.

If this is the only data collector set you have, then it will be set to the default data collector set. If you have other data collector sets, you can choose one of those to be the default. From the perspective of this analysis, the default setting is irrelevant. Either way, make sure that "Save and Close" is selected, and then click "Finish" to save your work. The wizard will close, returning you to the Performance Monitor.

You will see your new Log file, Profiler Correlation, listed under Data Collector sets, as shown in Figure 1-10:

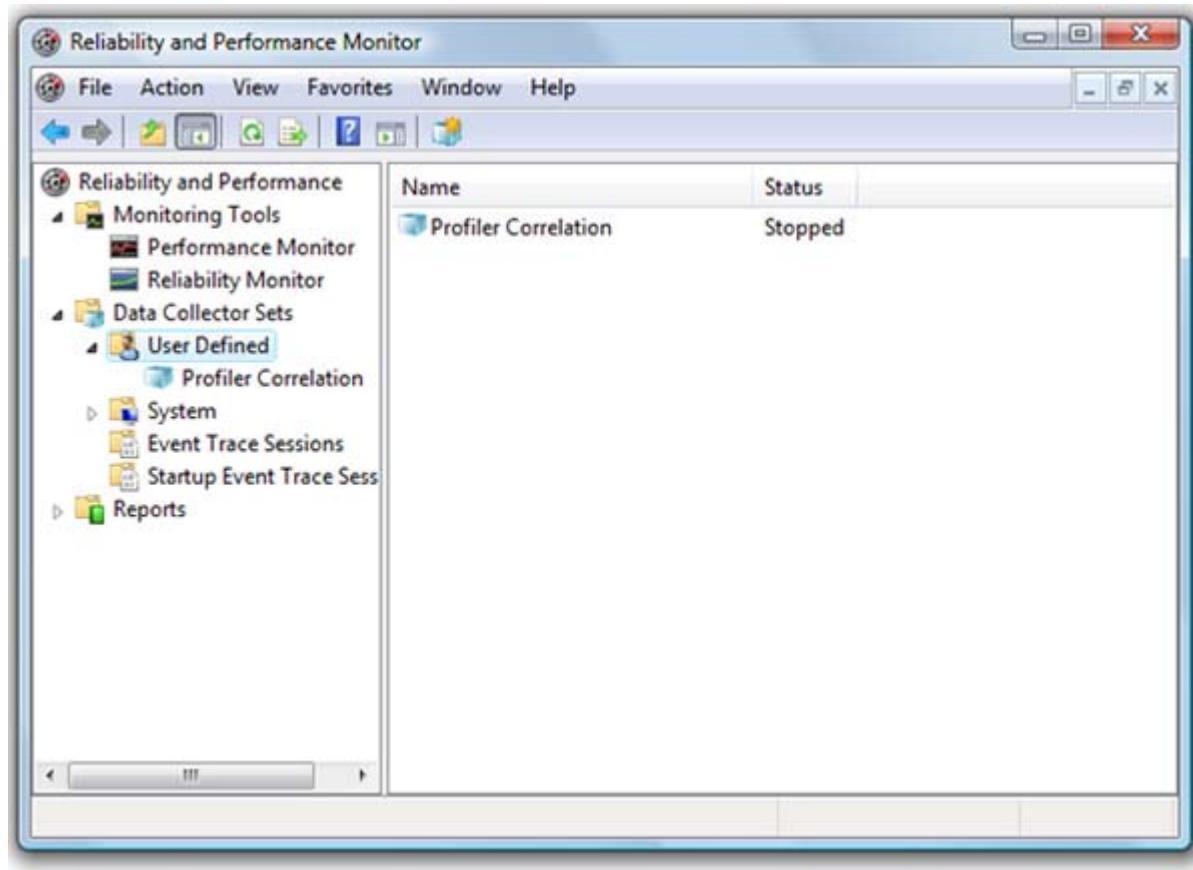


Figure 1-10: Once you are done, your screen should look similar to this.

Collecting Performance Monitor Data

We are now done with the hard work. The only thing left to do is to start collecting the "Profiler Correlation" data. To do this, right-click on the "Profiler Correlation" data collection set and select "Start", as shown in Figure 1-11:

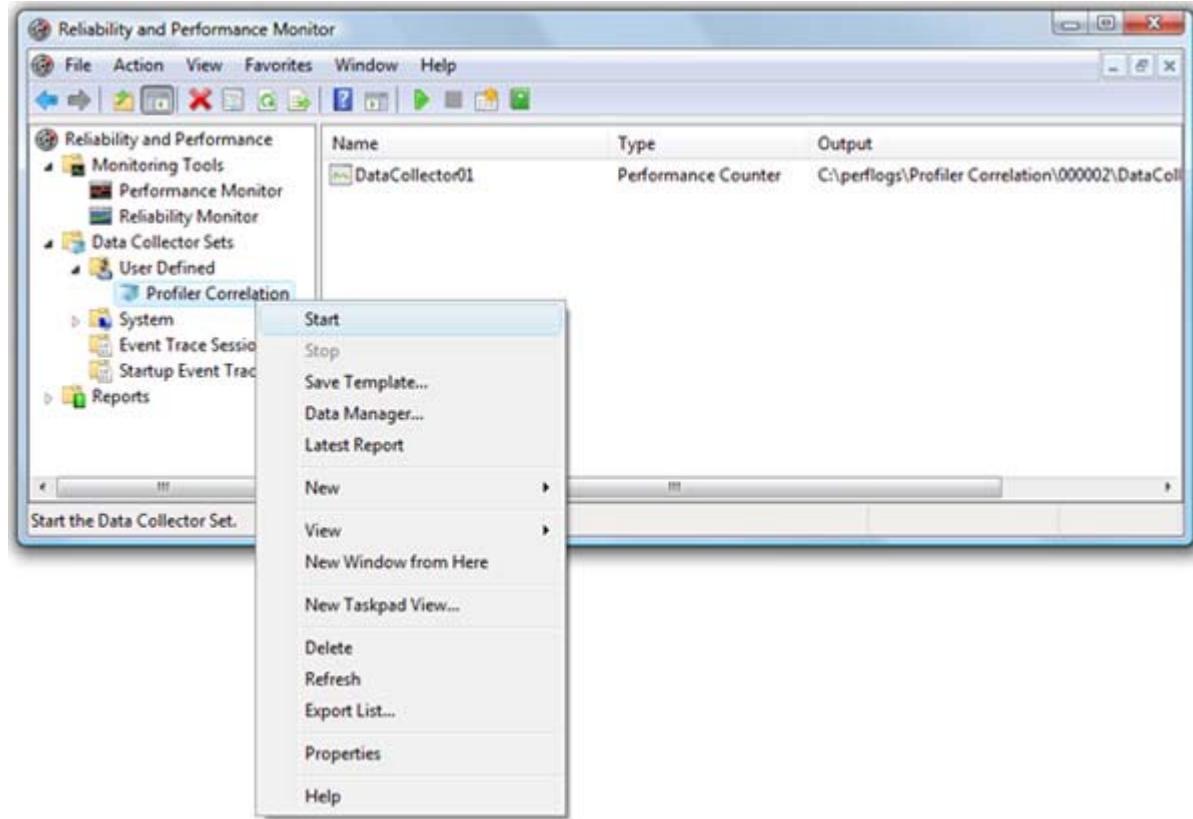


Figure 1-11: Right-click on the name of the Data Collector Set to start and stop it.

To stop collecting activity data, right-click on the "Profiler Correlation" data collection set and click "Stop". You can also schedule Data Collection Sets to start and stop automatically, using the Vista-based Performance Monitor main menu.

In the next section, we will look at the best ways to start and stop both Profiler traces and Performance Monitor data collection sets.

How to Capture Profiler Traces and Performance Monitor Logs

Now that we've created both a Profiler trace and a Performance Monitor Data Collector Set, we are ready to start both the tools and begin collecting the data that we will later correlate. Here are some points to keep in mind when running these tools to capture data:

- Run your copy of Profiler and Performance Monitor on a computer other than the SQL Server you are monitoring.
- Both the Profiler trace and the Performance Monitor logs should be started and stopped at about the same time. For example, if you decide to run an analysis for a two-hour

period, start both tools at the 8:00 AM and end them at 10:00 AM. If the traces don't occur at the same time, they cannot be correlated.

- Be sure that the SQL Server instance that you are monitoring, and the computer on which you are running Profiler and Performance Monitor, are in the same time zones. If they are not, the data cannot be correlated correctly.
- Make sure that the physical server running your SQL Server instance is not doing other work that could interfere with the analysis, such as running a different program or performing a backup. The only way to perform an accurate correlation analysis is to ensure the SQL Server is performing only regular production activity.
- As I have said many times, be sure that you only collect the minimum necessary number of events, data columns, and counters that you need for your analysis, especially when collecting data every second. We want to minimize the impact that Profiler and Performance Monitor have on the system.
- Run your trace and log files at a "representative" time of day. For example, if you find that your server's resources are peaking almost every morning between 9:00 AM and 11:00 AM, then this is the best time to capture your data.
- Monitor the size of the Profiler trace file and Performance Monitor log file during the capture, to ensure that not too much data is being collected. If the file sizes get too big, you may have to stop your data capture sooner than you planned.

Once you have completed capturing the data for both tools, you are ready to perform the correlation analysis.

How to Correlate SQL Server 2012 Profiler Data with Performance Monitor Data

Correlating Performance Monitor and Profiler data is a straightforward process that simply involves importing both sets of data into Profiler. Start Profiler and load the trace file you want to correlate. It should be displayed on the screen, as shown in Figure 1-12, just as for any other trace:

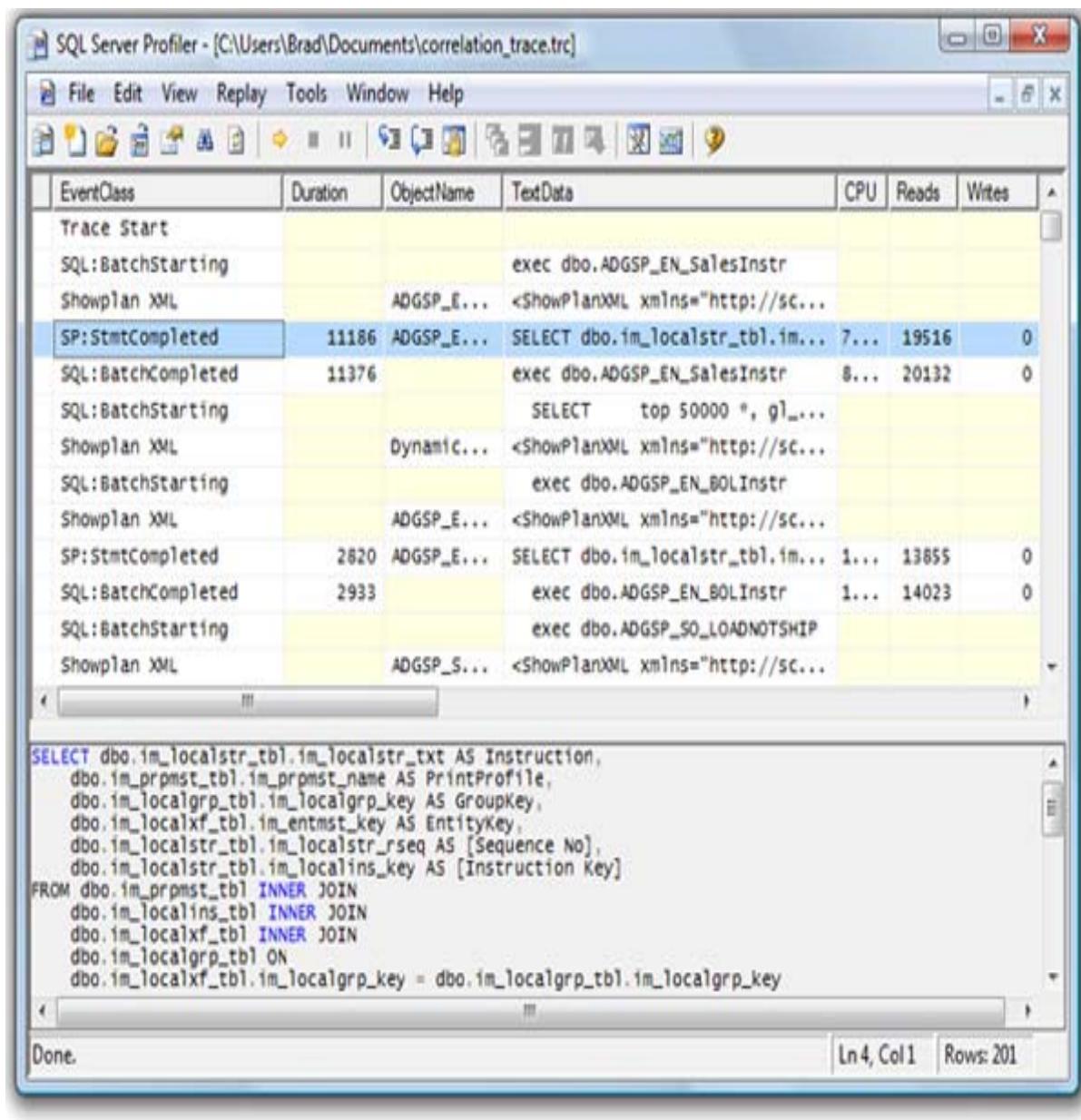


Figure 1-12: Start Profiler and load the trace file.

From the main menu of Profiler, select File | Import Performance Data, as shown in Figure 1-13:

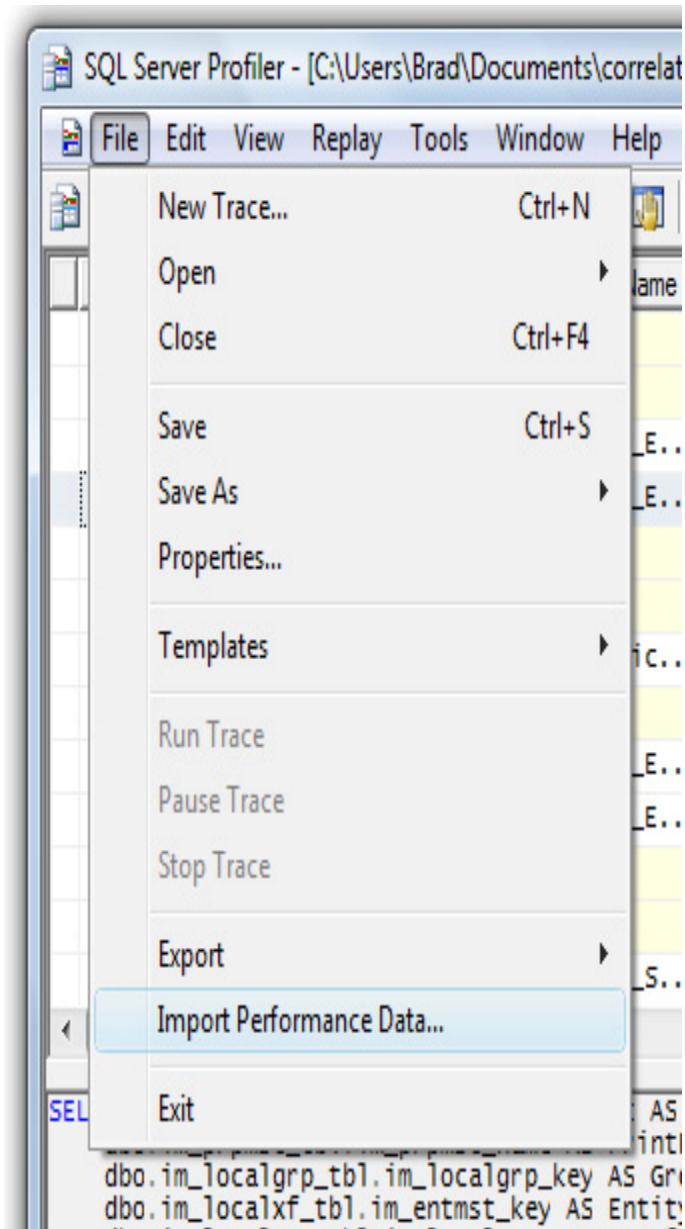


Figure 1-13: Select File|Import Performance Data

NOTE:

If the "Import Performance Data" option is grayed out, exit Profiler, then restart it, reload the trace data, and try again.

The screen shown in Figure 1-14 appears:

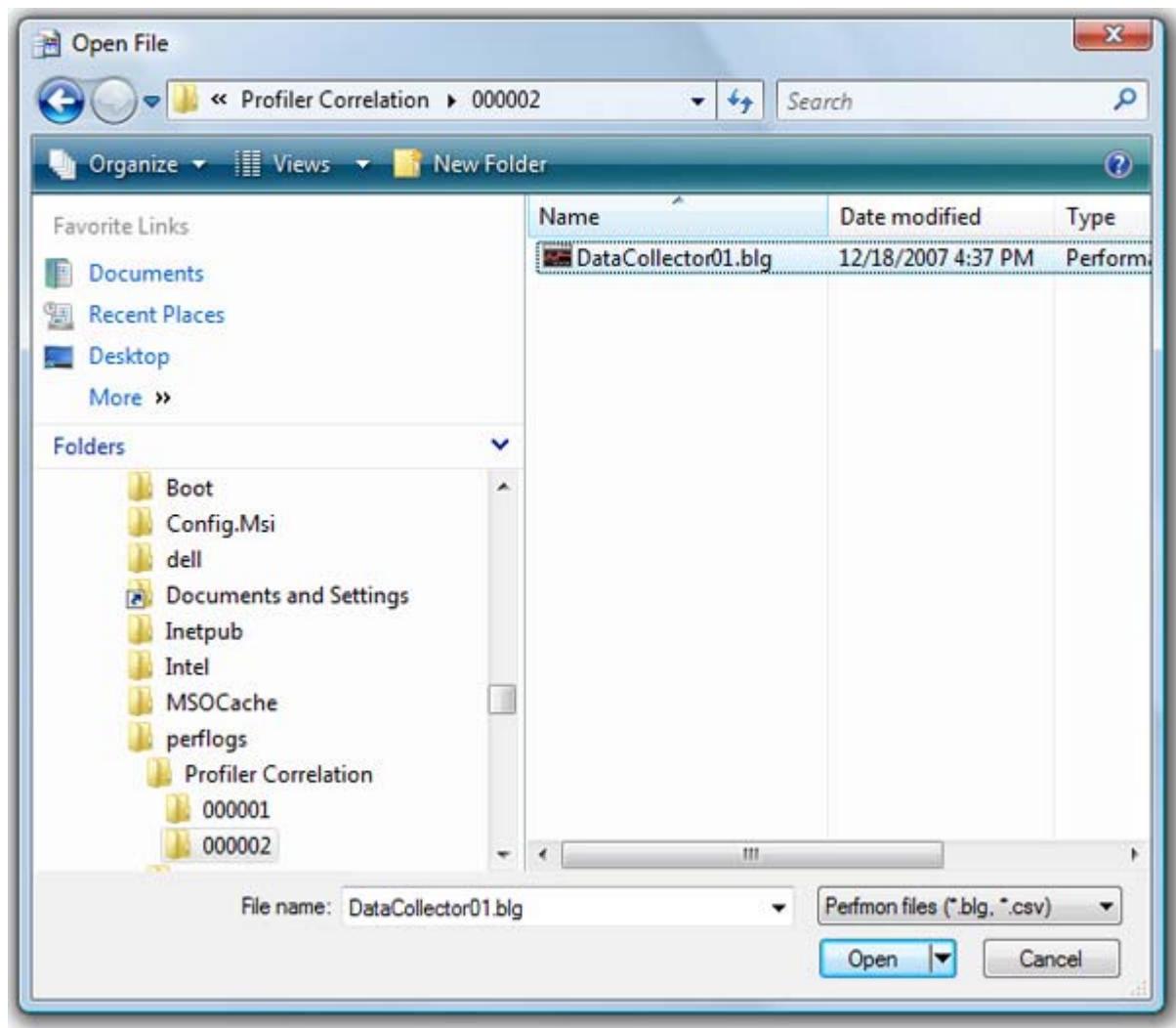


Figure 1-14: Select the Performance Monitor log you want to correlate with your Profiler data.

Locate your Performance Monitor log file and then click "Open":

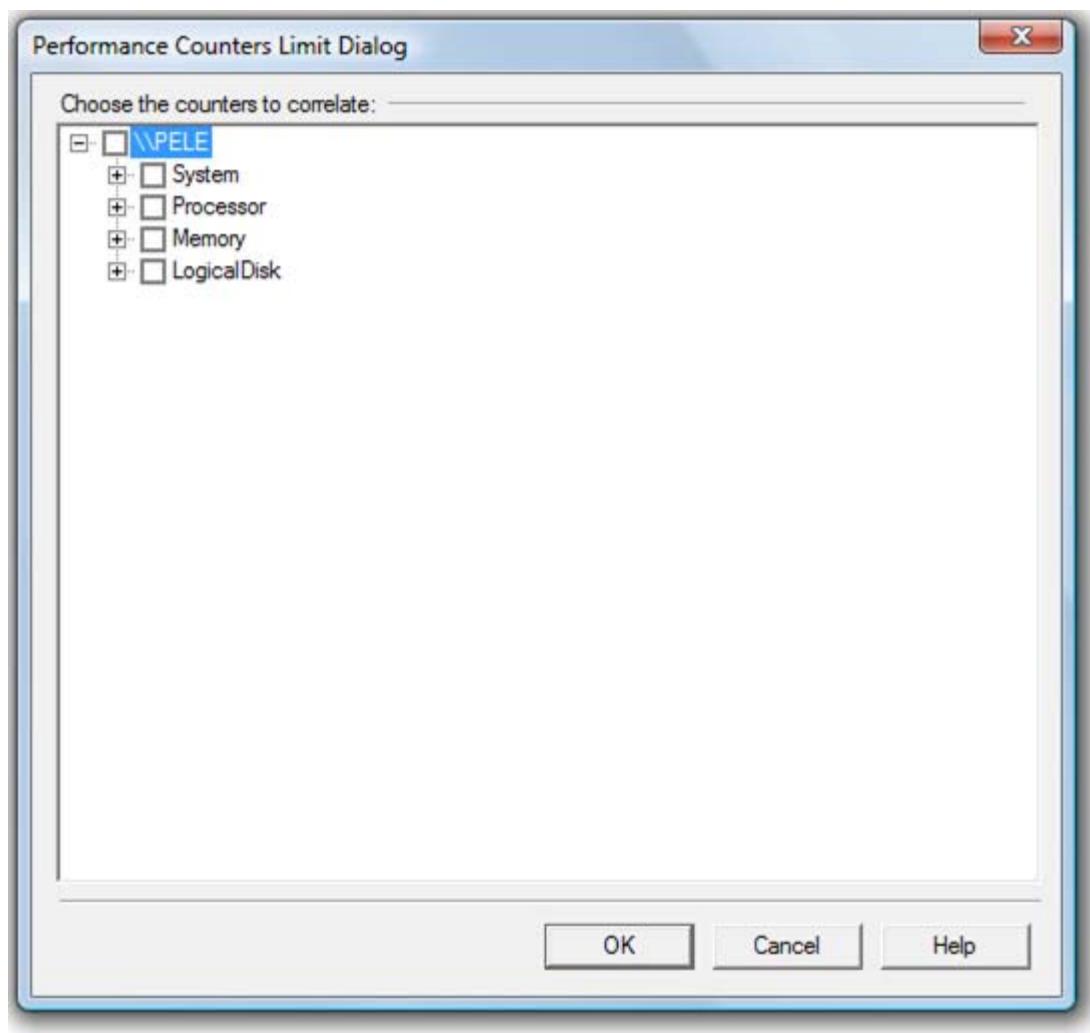
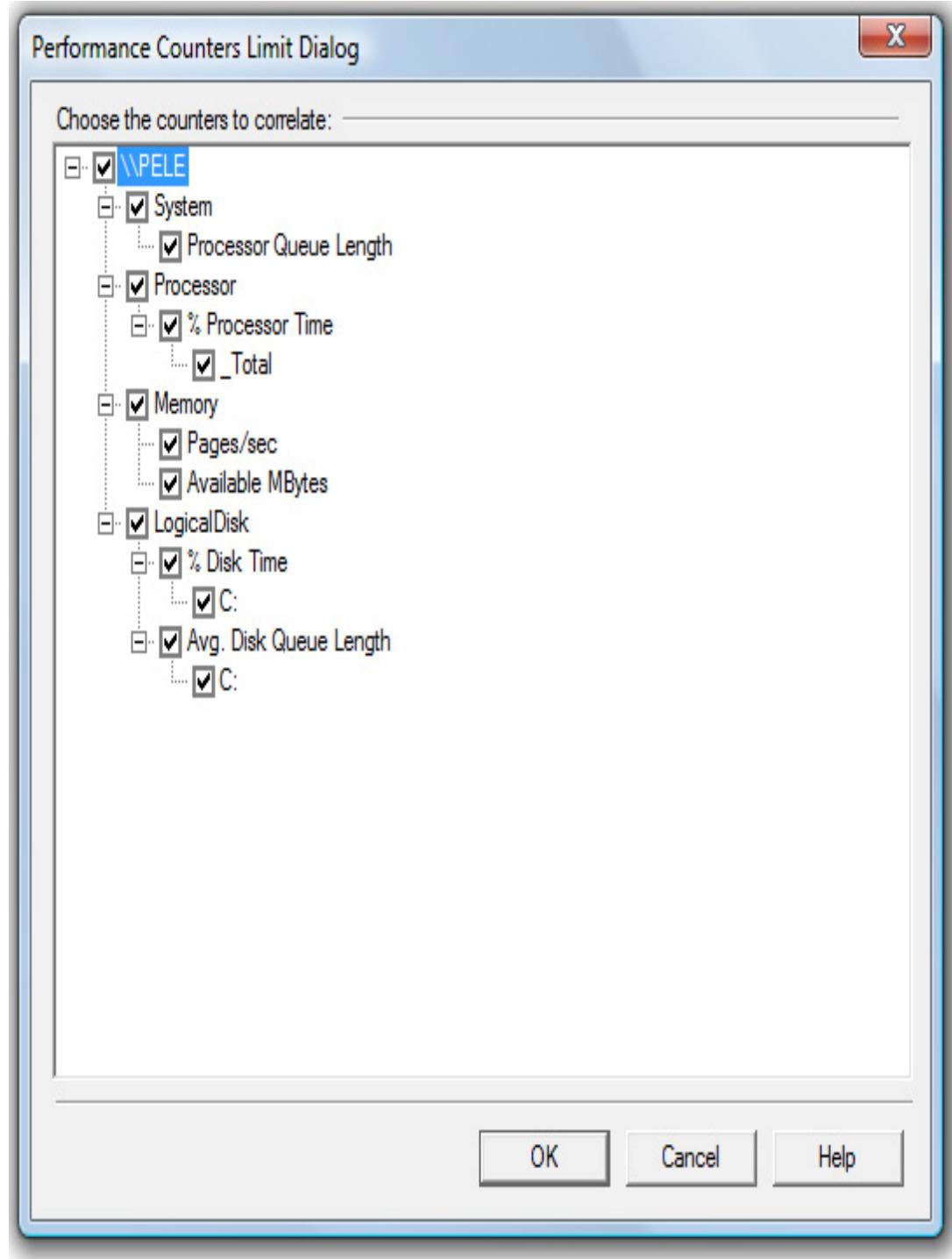


Figure 1-15: You must select the counters you want to correlate with your Profiler data.

The screen shown in Figure 1-15 allows you to select which counters to display as part of the correlation analysis. Ideally, you should include no more than about six, or the analysis screen (shown in Figure 1-17) will get too busy and be hard to read.

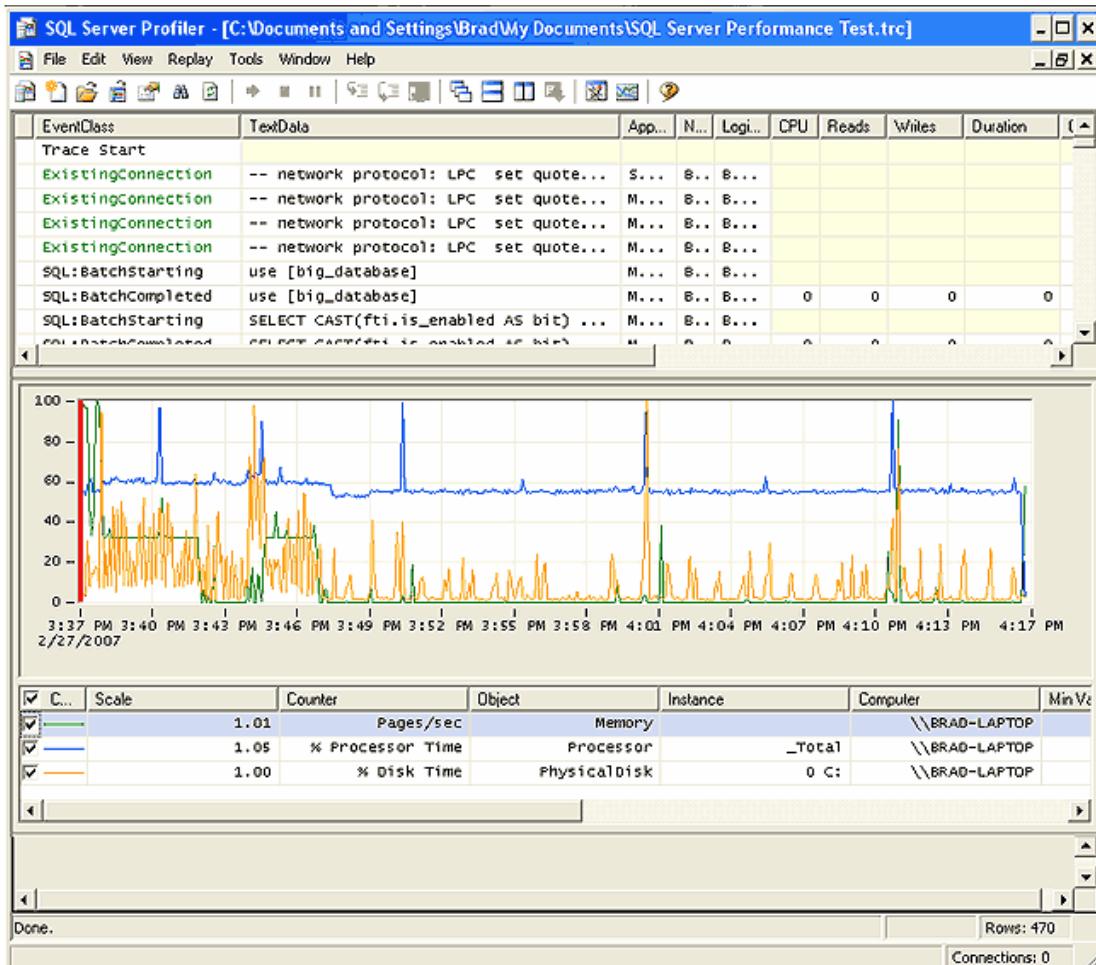
In this example, only six counters were collected, so import them all by clicking on the checkbox next to the server's name, as shown in Figure 1-16:



OK

Cancel

Help

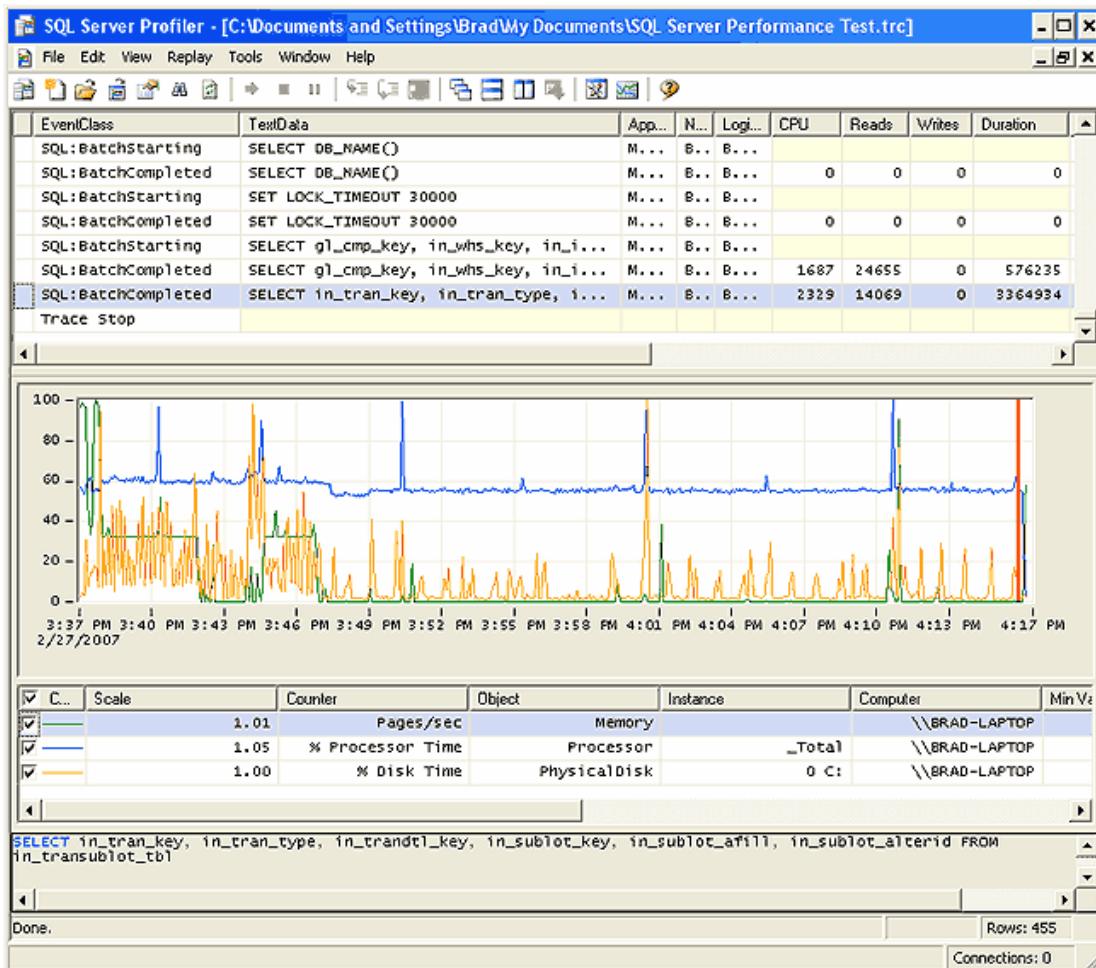


In the above window, you now see a correlation between Profiler Events (at the top of the dialog box) and System Monitor Counter data in the graph (in the middle). Depending on how many events and counters you are viewing, the dialog box above can become very busy. The larger your screen, and the fewer the events and counters displayed, the easier the data will be to view and analyze.

You can view the data two different ways. First, you can click anywhere on the graph and a red line will appear. This indicates the counter activity at a specific point in time. And then at the top, you will see an event highlighted that occurred closest to the time the counter data was collected. This option allows you to see what event (or events) was occurring at a specific point in time. You can then see what query or other activity was going on during this time and how it was affecting server resources.

Second, you can click on any of the Profiler events at the top of the dialog box, and then the red line in the graph will move to the time that particular event took place. This way, you can see how a particular event affected server resources.

For example, in the figure below, notice that I have clicked on an event at the top of the window that had a duration of 3364934 milliseconds. Notice where the red line is. Here, you can see that this particular event, which was very long, was the likely cause that the % Processor Time ran about 60% or so for a very long period.



Fully Understanding What You Are Seeing

After you begin playing around with this correlation window, you may discover some oddities. For example, above, how do I know that this long running query was the cause for the CPU running for 60% for such a long time? Actually, I don't. It is an educated guess. This tool does not have the ability to granularly determine exactly how much server resources were used up by a particular query. But you can make some educated guesses. In my case, I sorted all of the queries captured by Profiler by duration. By doing this, I found out that one of the queries ran for 3364934 milliseconds. Most of the others did not run that long. In the graph, I can see that the % CPU Utilization was high for about this same period of time. Thus, I can conclude that this particular query was the main cause of this excess CPU utilization. Especially when I can see that the CPU utilization went down to almost zero after the query finished executing.

You will also want keep in mind that multiple queries will probably be running at the same time, and that there may not be an exact time correlation between counters and events, as counters are measured over an interval of time you specify, and while events occur at discreet points in time. If you keep all of this in mind, and do a little practicing with the data, I think you will come to find that this correlation capability is very useful indeed. Now, finally, I can fairly easily determine what query (or queries) is causing my server to get stressed out.

Case study 2: How to detect Deadlock in SQL Server by using profiler

Poorly written queries in SQL Server can trigger deadlock in the system. If not common, at times you may need to troubleshoot deadlock issues. We all knows cyclic dependency causes dead lock. When SQL Server find a deadlock, it kill one process (deadlock victim) rolls back the transaction of the deadlock victim, and returns a 1205 error to the application. Database engine choose deadlock victim according to the least cost to rollback. Read more about Deadlock in Books online.

You can trace deadlock using Profiler Deadlock Graph event. This is one of the finest utility since it gives you graphical easy to read display of deadlock.

How to create profiler trace to find deadlock?

- (a) Create a new trace, using a Blank template.
- (b) Add the Deadlock graph event to the trace from the Locks category. You will get an additional tab appears on the Trace Properties window, called Event Extraction Settings.
- (c) Click the Save Deadlock XML Events Separately check box. This causes the deadlock information to be written to a separate file.

If you completed all the above mentioned steps (sl (a) to (c)) , the trace is already running on the server to detect deadlock. No we need to simulate Deadlock situation.

Simulation of Deadlock Scenario

-(a) Open first query analyzer window and run the following script

```
Create table TestDeadLoack (ID int,Name varchar(100))
```

```
Create table DeadLoack (ID int,Name varchar(100))
```

```
Insert TestDeadLoack Select 1,'Madhu'
```

```
Insert TestDeadLoack Select 2,'ABC'
```

```
Insert TestDeadLoack Select 3,'XYZ'
```

```
Insert DeadLoack Select 1,'Madhu'
```

```
Insert DeadLoack Select 2,'ABC'
```

```
Insert DeadLoack Select 3,'XYZ'
```

```
-- Dead lock scenario starts
```

```
Begin Tran
```

```
UPDATE TestDeadLoack SET Name = 'Madhu'
```

The screenshot shows the SQL Server Management Studio interface with two query windows open. The top window, titled 'LHI-115.test - SQLQuery7.sql*', contains the following T-SQL script:

```
1 Use Test --Database for test in my server
2
3 --Creating two tables to simulate Deadlock situation
4 GO
5 Create table TestDeadLoack (ID int,Name varchar(100))
6 GO
7 Create table DeadLoack (ID int,Name varchar(100))
8 GO
9
10 Insert TestDeadLoack Select 1,'Madhu'
11 Insert TestDeadLoack Select 2,'ABC'
12 Insert TestDeadLoack Select 3,'XYZ'
13
14 Insert DeadLoack Select 1,'Madhu'
15 Insert DeadLoack Select 2,'ABC'
16 Insert DeadLoack Select 3,'XYZ'
17
18
19 -- Dead lock scenario starts
20
21 Begin Tran
22 UPDATE TestDeadLoack SET Name = 'Madhu'
23
24
```

-(b) Open another query analyser windows and run the following script

--Step #2 - To be run in second query analyser window

```
Begin tran
```

```
UPDATE DeadLoack SET Name = 'xyz'
```

```
SELECT * FROM TestDeadLoack
```

The screenshot shows the SQL Server Management Studio interface with two query windows open. The bottom window, titled 'LHI-115.test - SQLQuery7.sql*', contains the following T-SQL script:

```
1 Begin tran
2 UPDATE DeadLoack SET Name = 'xyz'
3 SELECT * FROM TestDeadLoack
4
5
```

-(c) Go back to first query analyser window and run the following command

--Step #3 -- To be run in the first query analyser window

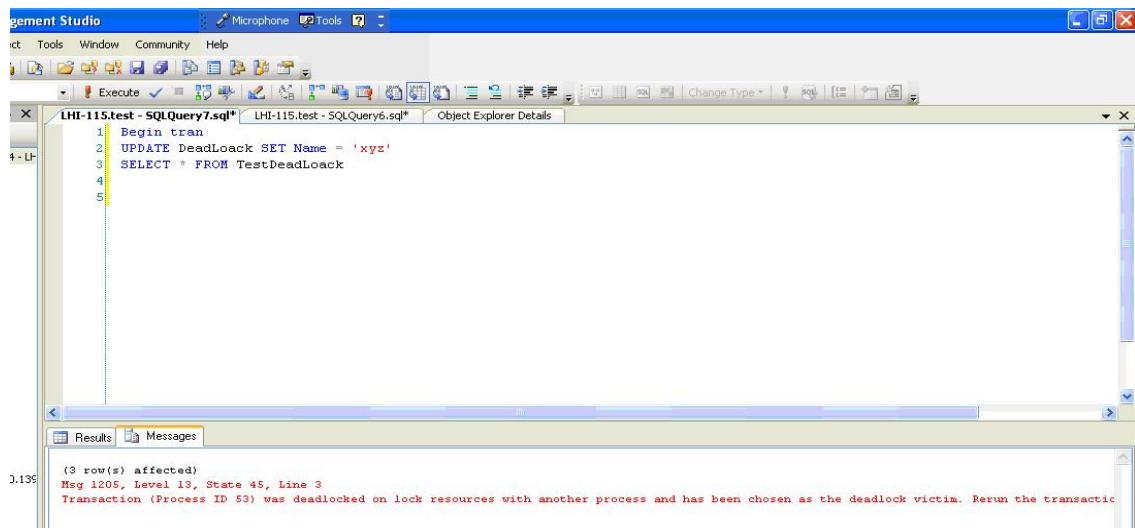
```
Select *From DeadLoack
```

Basically, here cyclic dependency is been created and system has to choose one of the process as deadlock victim. In my case, system made the Step #2 process (query analyzer window 2) as deadlock victim with the following error

(3 row(s) affected)

Msg 1205, Level 13, State 45, Line 3

Transaction (Process ID 53) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction.



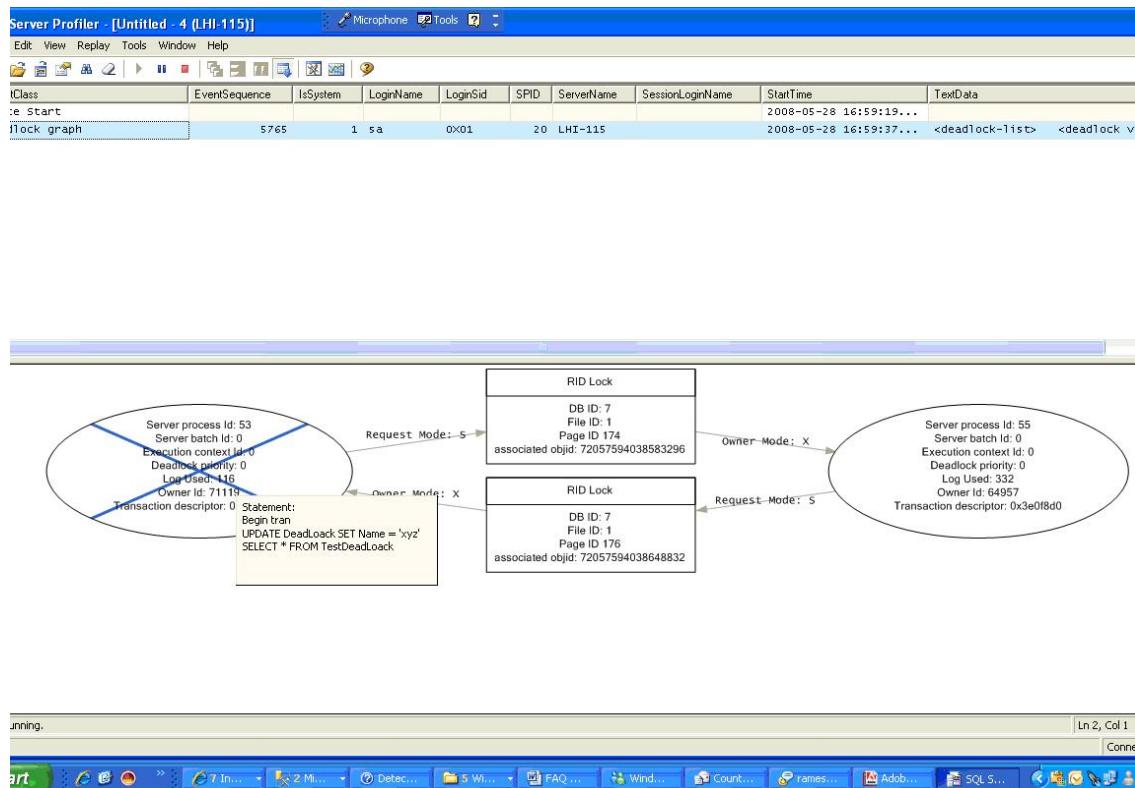
The screenshot shows the Microsoft Management Studio (MMS) interface. A query window titled 'LHI-115.test - SQLQuery7.sql' is open, displaying the following SQL code:

```
1 Begin tran
2 UPDATE DeadLoack SET Name = 'xyz'
3 SELECT * FROM TestDeadlock
4
5
```

The status bar at the bottom of the window shows the following message:

0.135 (3 row(s) affected)
Msg 1205, Level 13, State 45, Line 3
Transaction (Process ID 53) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transactio

In Profiler would show something like this



In the profiler , the Deadlock graph event contains very useful informations about the process and the sql statements caused the deadlock. The oval nodes shows the processes involved in the deadlock. The oval with an X mark across it is the deadlock victim. The other oval where there is no X mark says that process was allowed to complete after the system killed the deadlock victim process and deadlock resolved. IF you point your mouse on these shapes it will give you more information (see figure 4)

You can also export the deadlock event. Steps are as follows

Profiler – File – Export – Extract SQL Server Events – Extract Deadlock events and save to some files

Case study 3: SQL Server: Performance Counters- Thresholds

Performance – Non Disk Counters

Performance – Disk Counters

Performance – SQL Server Counters

To troubleshoot overall Database system performance issue, analyzing performance counters is the best way to start. By collecting performance counters during busy period for few days consistently and analyzing those data would give a better idea about overall system problems regarding Memory, CPU, and/or Disk I/O. Please note, for troubleshooting a particular SQL

problem such as a stored procedure or a piece of T-SQL, it is better to look at the query execution plan and SQL Trace data and identify the need of redesigning a query or table indexes.

Following are some key performance counters to use while assessing a performance issues on SQL Server.

Memory and Disk I/O complements each other. Memory issues on the system could affect disk I/O and vice versa. It is very critical to carefully observe the trend of performance counters data over a long period of time to identify the real problem.

Performance Non-Disk Counters			
Object	Counter	Preferred Value	Description
Memory	Available Mbytes	> 100MB	Available MBytes is the amount of physical memory available to processes running on the computer, in Megabytes. Note that this counter displays the last observed value only. It is not an average.
Memory	Pages Input/Sec	< 10	Higher the value poor the performance. Pages Input/sec is the rate at which pages are read from disk to resolve hard page faults. Hard page faults occur when a process refers to a page in virtual memory that is not in its working set or elsewhere in physical memory, and must be retrieved from disk. See KB 889654.
Memory	Pages/Sec	See Description	Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. Investigate if over 100 pages per second on a system with a slow disk, usually even 500 pages per second on a system with a fast disk subsystem may not be an issue.

			<p>Note:</p> <ul style="list-style-type: none"> Values of >20 pages that appear in many other sources of documentation are out of date. A high value for the Memory: Pages/sec counter does not necessarily indicate memory pressure or a System Monitor reporting error. To gain an accurate reading of your system, you must also monitor other counters (Pages Input/Sec, %Usage, %Usage Peak). See KB 889654.
Paging File	%Usage	< 70%	The amount of the Page File instance in use in percent. See KB 889654.
Paging File	%Usage Peak	< 70%	The peak usage of the Page File instance in percent. See KB 889654.
Process (sqlservr)	%Processor Time	< 80%	% Processor Time is the percentage of elapsed time that all of process threads used the processor to execution instructions. An instruction is the basic unit of execution in a computer, a thread is the object that executes instructions, and a process is the object created when a program is run. Code executed to handle some hardware interrupts and trap conditions are included in this count.
Process (msmdsrv)	%Processor Time	< 80%	For the SSAS. 80% is for a server which is dedicated to SSAS.
Processor	%Privileged Time	< 30% of Total %Processor Time	% Privileged Time is the percentage of elapsed time that the process threads spent executing code in privileged mode.
Processor	%Processor Time	< 80%	% Processor Time is the percentage of elapsed time that the processor

SQL Performance Counters

			spends to execute a non-Idle thread.
System	Processor Queue Length	< 4 per CPU	For standard servers with long Quanta <= 4 per CPU Excellent < 8 per CPU Good < 12 per CPU Fair

Performance Disk Counters

When the data files are places on a SAN ignore the following!! Use the performance tools provided by the SAN vendor instead

Object	Counter	Preferred Value	Description
PhysicalDisk	Avg. Disk Sec/Read	< 8ms	Measure of disk latency. Avg. Disk sec/Read is the average time, in seconds, of a read of data from the disk. More Info: Reads Excellent < 08 Msec (.008 seconds) Good < 12 Msec (.012 seconds) Fair < 20 Msec (.020 seconds) Poor > 20 Msec (.020 seconds)
PhysicalDisk	Avg. Disk sec/Write	< 8ms (non cached) < 1ms (cached)	Measure of disk latency. Avg. Disk sec/Write is the average time, in seconds, of a write of data to the disk. Non cached Writes Excellent < 08 Msec (.008 seconds) Good < 12 Msec (.012 seconds) Fair < 20 Msec (.020 seconds) Poor > 20 Msec (.020 seconds) Cached Writes Only Excellent < 01 Msec (.001 seconds) Good < 02 Msec (.002 seconds) Fair < 04 Msec (.004 seconds) Poor > 04 Msec (.004 seconds)

Object	Counter	Preferred Value	Description
SQLServer:Access Methods	Forwarded Records/sec	< 10 per 100 Batch Requests/Sec	Rows with varchar columns can experience expansion when varchar values are updated with a longer string. In the case where the row cannot fit in the existing page, the row migrates and access to the row will traverse a pointer. This only happens on heaps (tables without clustered indexes). Evaluate clustered index for heap tables. In cases where clustered indexes cannot be used, drop non-clustered indexes, build a clustered index to reorg pages and rows, drop the clustered index, then recreate non-clustered indexes.
SQLServer:Access Methods	Full Scans / sec	(Index Searches/sec)/(Full Scans/sec) > 1000	This counter monitors the number of full scans on base tables or indexes. Values greater than 1 or 2 indicate that we are having table / Index page scans. If we see high CPU then we need to investigate this counter, otherwise if the full scans are on small tables we can ignore this counter. A few of the main causes of high Full Scans/sec are

			<ul style="list-style-type: none"> • Missing indexes • Too many rows requested <p>Queries with missing indexes or too many rows requested will have a large number of logical reads and an increased CPU time.</p>
SQLServer:Access Methods	Index Searches/sec	(Index Searches/sec)/(Full Scans/sec) > 1000	<p>Number of index searches. Index searches are used to start range scans, single index record fetches, and to reposition within an index. Index searches are preferable to index and table scans. For OLTP applications, optimize for more index searches and less scans (preferably, 1 full scan for every 1000 index searches). Index and table scans are expensive I/O operations.</p>
SQLServer:Access Methods	Page Splits/sec	< 20 per 100 Batch Requests/Sec	<p>Number of page splits per second that occur as the result of overflowing index pages. Interesting counter that can lead us to our table / index design. This value needs to be low as possible. If you find out that the number of page splits is high, consider increasing the fillfactor of your indexes. An increased</p>

			<p>fillfactor helps to reduce page splits because there is more room in data pages before it fills up and a page split has to occur.</p> <p>Note that this counter also includes the new page allocations as well and doesn't necessarily pose a problem. The other place we can confirm the page splits that involve data or index rows moves are the fragmented indexes on page splits.</p>
SQL Server:Buffer Manager	Buffer Cache hit ratio	> 90%	<p>This counter indicates how often SQL Server goes to the buffer, not the hard disk, to get data. The higher this ratio, the less often SQL Server has to go to the hard disk to fetch data, and performance overall is boosted. Unlike many of the other counters available for monitoring SQL Server, this counter averages the Buffer Cache Hit Ratio from the time the last instance of SQL Server was restarted. In other words, this counter is not a real-time measurement, but an average of all the days since SQL Server was last restarted. In OLTP applications, this ratio should exceed 90-95%.</p>

			If it doesn't, then you need to add more RAM to your server to increase performance. In OLAP applications, the ratio could be much less because of the nature of how OLAP works. In any case, more RAM should increase the performance of SQL Server OLAP activity.
SQL Server:Buffer Manager	Free list stalls/sec	< 2	Free list stalls/sec is the frequency with which requests for available database pages are suspended because no buffers are available. Free list stall rates of 3 or 4 per second indicate too little SQL memory available.
SQL Server:Buffer Manager	Free pages	> 640	Total number of pages on all free lists.
SQL Server:Buffer Manager	Lazy Writes/Sec	< 20	This counter tracks how many times a second that the Lazy Writer process is moving dirty pages from the buffer to disk in order to free up buffer space. Generally speaking, this should not be a high value, say more than 20 per second or so. Ideally, it should be close to zero. If it is zero, this indicates that your SQL Server's buffer cache is plenty big and SQL Server

			doesn't have to free up dirty pages, instead waiting for this to occur during regular checkpoints. If this value is high, then a need for more memory is indicated.
SQL Server:Buffer Manager	Page Life Expectancy	> 300	This performance monitor counter tells you, on average, how long data pages are staying in the buffer. If this value gets below 300 seconds, this is a potential indication that your SQL Server could use more memory in order to boost performance.
SQLServer:Buffer Manager	Page lookups/sec	(Page lookups/sec) / (Batch Requests/sec) < 100	Number of requests to find a page in the buffer pool. When the ratio of page lookups to batch requests is much greater than 100, this is an indication that while query plans are looking up data in the buffer pool, these plans are inefficient. Identify queries with the highest amount of logical I/O's and tune them.
SQL Server:Buffer Manager	Page reads/sec	< 90	Number of physical database page reads issued. 80 – 90 per second is normal, anything that is above indicates indexing or memory constraint.

SQL Server:Buffer Manager	Page writes/sec	< 90	Number of physical database page writes issued. 80 – 90 per second is normal, anything more we need to check the lazy writer/sec and checkpoint counters, if these counters are also relatively high then, it's memory constraint.
SQLServer:General Statistics	Logins/sec	< 2	> 2 per second indicates that the application is not correctly using connection pooling .
SQLServer:General Statistics	Logouts/sec	< 2	> 2 per second indicates that the application is not correctly using connection pooling.
SQLServer:General Statistics	User Connections	See Description	<p>The number of users currently connected to the SQL Server.</p> <p>Note: It is recommended to review this counter along with "Batch Requests/Sec". A surge in "user connections" may result in a surge of "Batch Requests/Sec". So if there is a disparity (one going up and the other staying flat or going down), then that may be a cause for concern. With a blocking problem, for example,</p>

			you might see user connections, lock waits and lock wait time all increase while batch requests/sec decreases.
SQL Server:Latches	Latch Waits/sec	(Total Latch Wait Time) / (Latch Waits/Sec) < 10	This is the number of latch requests that could not be granted immediately. In other words, these are the amount of latches, in a one second period that had to wait.
SQL Server:Latches	Total Latch Wait Time (ms)	(Total Latch Wait Time) / (Latch Waits/Sec) < 10	This is the total latch wait time (in milliseconds) for latch requests in the last second
SQL Server:Locks	Lock Wait Time (ms)	See Description"	<p>Total wait time (milliseconds) for locks in the last second.</p> <p>Note: For "Lock Wait Time" it is recommended to look beyond the Avg value. Look for any peaks that are close (or exceeds) to a wait of 60 sec. Though this counter counts how many total milliseconds SQL Server is waiting on locks during the last second, but the counter actually records at the end of locking event. So most probably the peaks represent one huge locking event. If those events exceeds more</p>

			than 60seconds then they may have extended blocking and could be an issue. In such cases, thoroughly analyze the blocking script output. Some applications are written for timing out after 60 seconds and that's not acceptable response for those applications.
SQL Server:Locks	Lock Waits/sec	0	This counter reports how many times users waited to acquire a lock over the past second. Note that while you are actually waiting on the lock that this is not reflected in this counter—it gets incremented only when you "wake up" after waiting on the lock. If this value is nonzero then it is an indication that there is at least some level of blocking occurring. If you combine this with the Lock Wait Time counter, you can get some idea of how long the blocking lasted. A zero value for this counter can definitively prove out blocking as a potential cause; a nonzero value will require looking at other information to determine whether it is significant.

SQL Server:Locks	Number of Deadlocks/sec	< 1	The number of lock requests that resulted in a deadlock.
SQLServer:Memory Manager	Total Server Memory(KB)	See Description	The Total Server Memory is the current amount of memory that SQL Server is using. If this counter is still growing the server has not yet reached its steady-state, and it is still trying to populate the cache and get pages loaded into memory. Performance will likely be somewhat slower during this time since more disk I/O is required at this stage. This behavior is normal. Eventually Total Server Memory should approximate Target Server Memory.
SQLServer:SQL Statistics	Batch Requests/Sec	See Description	This counter measures the number of batch requests that SQL Server receives per second, and generally follows in step to how busy your server's CPUs are. Generally speaking, over 1000 batch requests per second indicates a very busy SQL Server, and could mean that if you are not already experiencing a CPU bottleneck, that you may very well soon. Of course, this is a relative number, and

the bigger your hardware, the more batch requests per second SQL Server can handle. From a network bottleneck approach, a typical 100Mbs network card is only able to handle about 3000 batch requests per second. If you have a server that is this busy, you may need to have two or more network cards, or go to a 1Gbs network card.

Note: Sometimes low batch requests/sec can be misleading. If there were a SQL statements/sec counter, this would be a more accurate measure of the amount of SQL Server activity. For example, an application may call only a few stored procedures yet each stored procedure does lot of work. In that case, we will see a low number for batch requests/sec but each stored procedure (one batch) will execute many SQL statements that drive CPU and other resources. As a result, many counter thresholds based on the number of batch requests/sec will seem

			<p>to identify issues because the batch requests on such a server are unusually low for the level of activity on the server.</p> <p>We cannot conclude that a SQL Server is not active simply by looking at only batch requests/sec. Rather, you have to do more investigation before deciding there is no load on the server. If the average number of batch requests/sec is below 5 and other counters (such as SQL Server processor utilization) confirm the absence of significant activity, <i>then there is not enough of a load to make any recommendations or identify issues regarding scalability.</i></p>
SQLServer:SQL Statistics	SQL Compilations/sec	< 10% of the number of Batch Requests/Sec	The number of times per second that SQL Server compilations have occurred. This value needs to be as low as possible. If you see a high value such as over 100, then it's an indication that there are lots of adhoc queries that are running, might cause CPU usage, solution is to re-write these adhoc as stored procedure or use sp_executeSQL.

Case study 4: MAXDOP Settings to Limit Query to Run on Specific CPU

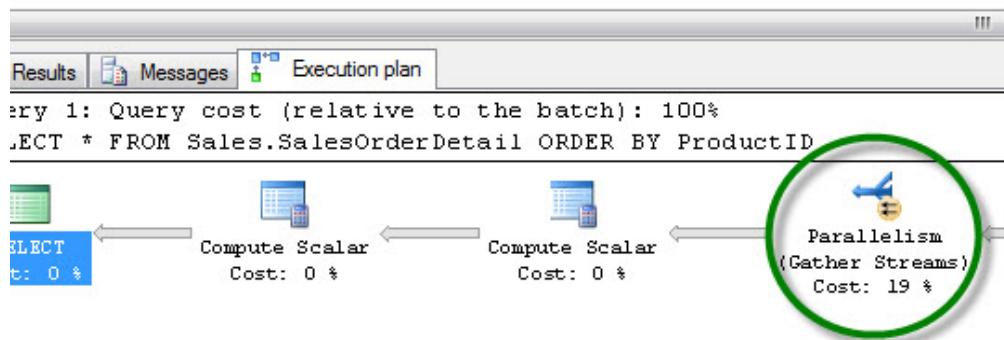
Query Hint MAXDOP – Maximum Degree Of Parallelism can be set to restrict query to run on a certain CPU. Please note that this query cannot restrict or dictate which CPU to be used, but for sure, it restricts the usage of number of CPUs in a single batch.

Let us consider the following example of this query.

The following query usually runs on multicore on a dual core machine (please note it may not be the case with your machine).

```
USE AdventureWorks
GO
SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
GO

USE AdventureWorks
GO
SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
GO
```



Now the same query can be ran on a single core with the usage of MAXDOP query hint. Let us see the query for the same.

```
USE AdventureWorks
GO
SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
OPTION (MAXDOP 1)
GO
```

```

SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
OPTION (MAXDOP 1)
GO

```

Execution plan:

- SELECT Cost: 0 %
- Compute Scalar Cost: 0 %
- Compute Scalar Cost: 0 %
- Compute Scalar Cost: 0 %
- Sort Cost: 90

Execution plan for the query with query hint of maxdop (1) does not have parallelism operator in it. This way we can remove the parallelism using MAXDOP.

However, before playing with this query hint, please make sure that you check your performance using an execution plan. It is quite possible that the performance of Query with MAXDOP as query hint may be quite degraded when compared to the original performance. You should be very careful with this hint.

Let us compare in our case what is the performance difference between the two above queries. The difference between those two queries is only the query hint of MAXDOP.

```

USE AdventureWorks
GO
SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
GO

SELECT *
FROM Sales.SalesOrderDetail
ORDER BY ProductID
OPTION (MAXDOP 1)
GO

```

Results Messages Execution plan

Query 1: Query cost (relative to the batch): 40%

SELECT * FROM Sales.SalesOrderDetail ORDER BY ProductID

```

graph LR
    A[SELECT] --> B[Compute Scalar]
    B --> C[Compute Scalar]
    C --> D[Parallelism<br/>(Gather Streams)]

```

Cost: 0 % Cost: 0 % Cost: 0 % Cost: 19 %

Query 2: Query cost (relative to the batch): 60%

SELECT * FROM Sales.SalesOrderDetail ORDER BY ProductID OPTION (MAXDOP 1)

```

graph LR
    A[SELECT] --> B[Compute Scalar]
    B --> C[Compute Scalar]
    C --> D[Compute Scalar]
    D --> E[Sort]
    E --> F[Clustered Index Scan]
    F --> G[Clustered Index Scan]

```

Cost: 0 % Cost: 0 % Cost: 0 % Cost: 90 % Cost: 100 %

In our example, we got degraded performance as we restricted our query on a single CPU. This is not necessary in the case of all the queries. MAXDOP may improve or reduce performance

Annexure I

Dynamic Management Views

-----Common Language Runtime (CLR):

sys.dm_clr_appdomains: Returns a row for each AppDomain, the unit of isolation for an application running in .NET, running on the server. SQL Server creates one AppDomain per database per owner so that all CLR objects are always executed in the same AppDomain.

sys.dm_clr_loaded_assemblies: Returns a row for each managed user assembly, i.e. managed code DLL files, loaded into the server address space.

sys.dm_clr_properties:

Returns a row for each property of a CLR assembly available to the system, such as the CLR version, current state of the hosted CLR, or the CLR install directory.

sys.dm_clr_tasks:

Returns a row for all currently running CLR tasks.

-----DATABASE :

sys.dm_db_file_space_usage:

Returns space usage information for each file in tempdb.

sys.dm_db_session_space_usage:

Returns the number of pages allocated and deallocated by each session, user or internal, in tempdb.

sys.dm_db_partition_stats:

Returns one row per partition in the database, showing page and row-count information for every partition in the current database, including the space used to store and manage in-row data, LOB data, and row-overflow data for all partitions in a database.

sys.dm_db_task_space_usage:

Returns page allocation and deallocation activity by task for tempdb, excluding IAM pages.

-----TRANSACTION:

sys.dm_tran_active_snapshot_database_transactions:

Returns a virtual table for each active transaction that could potentially generate access row versions. Each transaction it returns has an XSN (transaction sequence number) which is given to any transaction that accesses a version store.

sys.dm_tran_active_transactions:

Returns information about transactions (such as the state of the transaction, when it began, whether it is read-only or not, and so forth) executing within the SQL Server instance.

sys.dm_tran_current_snapshot:

Returns a virtual table of all active XSNs currently running when the current snapshot transaction starts. Returns no rows if the current transaction is not a snapshot transaction.

sys.dm_tran_current_transaction:

Returns a single row that displays the state and snapshot sequence information of a transaction in the current session. Not useful for transactions running in isolation levels other than snapshot isolation level

sys.dm_tran_current_transactions:

Returns information about transactions at the database level, especially transaction log information such as the log sequence number (LSN), log bytes used, and transaction type.

sys.dm_tran_locks:

Returns information about currently active requests to the lock manager, broken into a resource group and a request group. The request status may be active, convert, or may be waiting (wait) and includes details such as the resource on where the lock request wants a log (for the resource group) or the lock request itself (for the request group).

sys.dm_tran_session_transactions:

Returns correlation information for associated transactions and sessions, especially useful for monitoring distributed transactions.

sys.dm_tran_top_version_generators:

Returns a virtual table for the objects that are producing the most versions in the version store, as found in the sys.dm_tran_version_store system view. Use with caution. This is a very resource intensive DMV.

sys.dm_tran_transactions_snapshot:

Returns information about active transactions when each snapshot transaction starts. Using this, you can find how many snapshot transaction are currently active and identify any data modifications that are ignored by any given snapshot transaction.

sys.dm_tran_version_store:

Returns each versioned record, both its XSN and its binary version sequence number. Because the version store can be quite large, this DMV can be very resource intensive.

-----EXECUTION & THREAD:**sys.dm_exec_background_job_queue:**

Returns a row for each query processor job that is scheduled for asynchronous, i.e. background, execution. In SQL Server 2005, this will show update statistics jobs.

sys.dm_exec_background_job_queue_stats:

Returns a row that provides aggregate statistics for each query processor job submitted for background execution.

sys.dm_exec_cached_plans:

Similar to syscacheobjects in SQL Server 2000. Returns a row for each query plan that is held in procedure cache, and containing information like the cached query plans, the cached query text, the amount of memory taken by cached plans, and the reuse count of the cached plans.

sys.dm_exec_cached_plan_dependent_objects:

Returns a row for each TSQL execution plan, CLR execution plan, and cursor associated with a plan.

sys.dm_exec_connections:

Returns server-level information about a connection to SQL Server, such as the client network address, client TCP port, and client authorization scheme.

sys.dm_exec_cursors:

Returns information about cursors that are open in one or more databases on the server.

sys.dm_exec_plan_attributes:

Returns one row per plan attribute for a specific plan identified by its plan handle, such as information like the cache key values or the number of current simultaneous executions of the plan.

sys.dm_exec_query_memory_grants:

Returns information about queries that have acquired memory grants or that still require a memory grant to execute. This DMV is useful for determining query timeouts, since only queries that have to wait on a memory grant will appear in this view.

sys.dm_exec_query_optimizer_info:

Returns detailed statistics about the operation of the SQL Server query optimizer, such as the total number of optimizations, the elapsed time value, or sophisticated assessments like comparing the query optimization cost of the current workload to that of a tuned workload. Some counters shown by this DMV are for internal use only.

sys.dm_exec_query_plan:

Returns a given query's Showplan output in XML format, as specified by the plan handle

sys.dm_exec_query_resource_semaphores:

Returns two rows, one for the regular resource semaphore and the other for the small-query resource semafor, information about the current query-resource semaphore status. When used with sys.dm_os_memory_clerks, this DMV provides a complete picture of memory status information about query executions and allows you to determine whether the system can access enough memory..

sys.dm_exec_query_stats:

Returns one row per query statement within a cached plan, detailing aggregated performance statistics for cached query plans. Because the information is aggregated, you may sometimes get better information by rerunning this DMV.

sys.dm_exec_requests:

Returns one row for each request executing within SQL Server, but does not contain information for code that executes outside of SQL Server, such as distributed queries or extended stored procedures.

sys.dm_exec_sessions::

Returns one row per authenticated session, including both active users and internal tasks, running anywhere on SQL Server.

sys.dm_exec_sql_text:

Similar to fn_get_sql in SQL Server 2000, this DMV returns the text of the SQL batch that is identified by the specified sql_handle.

sys.dm_exec_text_query_plan:

Returns Showplan output in text format for a given TSQL batch or a statement within the batch. It's similar sys.dm_exec_query_plan, except that it returns data as text, is not limited in size, and may be specific to an individual statement within a batch.

sys.dm_exec_xml_handles:

Returns information about one or all active handles that opened with the sp_xml_preparedocument system stored procedure.

-----SQL SERVER OS:**sys.dm_os_buffer_descriptors:**

Returns information about all data pages that are in the SQL Server buffer cache, excluding free, stolen, or erroneously read pages. SQL Server 2005 tracks anything put into the buffer cache using buffer descriptors. You can easily scope the results to show one or all database and one or all of the objects within the database.

sys.dm_os_child_instances:

Returns a row showing the state and pipe name of each user instance spawned from the parent server instance.

sys.dm_os_cluster_nodes:

Returns a row for each node in the failover cluster instance configuration or an empty rowset if the server is not configured as in a failover cluster

sys.dm_os_hosts:

Returns a row for each host (such as an OLE DB provider) currently registered in an instance of SQL Server and each resource used by these hosts.

sys.dm_os_latch_stats:

Returns information about all latch waits, organized by latch class. Latches are holds placed on very lightweight and transient system resources, such as an address in memory. This DMV is useful for troubleshooting latch contention. It does not track latch usage where the latch was granted immediately or failed immediately.

sys.dm_os_loaded_modules:

Returns a row for each module loaded into the server.

Sys.dm_os_memory_brokers:

Returns a row for each memory broker that is currently active. Memory brokers are used by SQL Server 2008 to track memory allocations between various SQL Server internal and external components, based on current and projected usage.

sys.dm_os_memory_cache_clock_hands:

Returns the status (suspended or running) of each hand for a specific cache clock. The process used to age items out of cache is called the cache clock and each clock can have one or more processes (called a hand) to sweep the cache clean.

sys.dm_os_memory_cache_counters:

Returns an overview of the cache health, including run-time information about cache entries allocated, the source of memory for the cache entries, and how they are used.

sys.dm_os_memory_cache_entries:

Returns information, such as statistics, for all entries in the various caches. This DMV is useful for linking cache entries back to their associated database objects.

sys.dm_os_memory_cache_hash_tables:

Returns information about each active cache in the instance of SQL Server, such as the type of cache, the number and type of hash buckets, the length of time the hash buckets have been in use, etc.

sys.dm_os_memory_clerks:

Returns all currently active memory clerks within SQL Server. A memory clerk is the primary means for allocating memory to the various users of SQL Server.

sys.dm_os_memory_objects:

Returns all memory objects that are currently allocated by SQL Server. Memory objects are more granular than memory clerks and are used by internal SQL Server processes and components, but not users like memory clerks. This DMV is ideal for analyzing memory use and identifying memory leaks.

sys.dm_os_memory_pools:

Returns a row for each memory pool object store in the SQL Server instance. Memory pool objects are certain homogeneous, equally important, stateless types of data. This information is sometimes useful for identifying bad caching behavior.

Sys.dm_os_nodes:

Returns information about the currently active nodes on the instance. Nodes are created by SQL OS to mimic hardware processor locality and can be altered by SQL OS using soft-NUMA techniques.

Sys.dm_os_process_memory:

Returns information providing a complete picture of the process memory address space in kilobytes, including things like the page_fault_count, amount of virtual address space available and committed, process physical and virtual memory, and so forth.

sys.dm_os_performance_counters:

Returns a row per performance counter, the same counters as in Windows PerfMon, maintained by the server. To calculate a discrete value for the various per-second counters, you must sample two discrete values and then subtract the earlier cumulative per-second value from the later.

sys.dm_osSchedulers:

Returns one row per scheduler for schedulers mapped to an individual processor. Active_worker scheduler threads are those devoted to the regular user-controlled work of the server, such as queries and SSIS jobs, while a variety of system and hidden schedulers may be active on the system at any time. This DMV is very useful for monitoring the thread scheduler and to find runaway tasks.

sys.dm_os_stacks:

Used internally by SQL Server track debug data, such as outstanding allocations, and to validate the logic in a component that assumes that a certain call was made.

sys.dm_os_sys_info

Returns a miscellaneous set of useful information about the computer, such as the hyperthread ratio, the max worker count, and other resources used by and available to SQL Server.

Sys.dm_os_sys_memory

Returns a complete picture of memory at the operating system level, including information about total and available physical memory, total and available page memory, system cache, kernel space and so forth.

sys.dm_os_tasks

Returns one row for each OS task that is active in the instance of SQL Server.

sys.dm_os_threads

Returns a row for each SQLOS threads running under the SQL Server process.

sys.dm_os_virtual_address_dump

Returns information about the region, i.e. the range of pages in the virtual address space, used by a calling process.

sys.dm_os_wait_stats

Returns information about waits encountered by currently executing threads. There are a limited number of reasons why a thread might be forced to wait before it can complete execution. Refer to the SQL Server Books On-Line for more information about all possible wait states. This is an excellent DMV to use to diagnose performance issues with SQL Server and also with specific queries and batches because it records any time anything within SQL Server has to wait.

sys.dm_os_waiting_tasks

Returns information about the wait queue of SQLOS tasks that are waiting on some resource, such as blocking and latch contention.

sys.dm_os_workers

Returns a row detailing information for every worker in the system, what it is doing and what it is waiting to do.

-----INDEX:

sys.dm_db_index_operational_stats:

Returns current low-level I/O, locking, latching, and access method activity for one or all databases, and one or all tables, indexes and partitions within each database.

sys.dm_db_index_physical_stats:

Returns size and fragmentation information for the data pages and index pages for one or all databases, and one or all tables, indexes and partitions within each database. You may also specify a scanning mode

to speed processing. You will usually get multiple rows even if specifying a specific index on a specific table in a specific database. The DMV returns one row for indexes for each level of the B-tree in each partition. One row is returned for heaps for the IN_ROW_DATA allocation unit of each partition. One row is returned for LOB data and two-overflow data, if they exist in each partition.

sys.dm_db_index_usage_stats:

Returns a count of different types of index operations (such as seeks, scans, lookups, and updates) and the last time each operation was performed.

sys.dm_db_missing_index_columns:

Returns information about columns that are missing an index, which can then be used to create indexes on those columns.

sys.dm_db_missing_index_details:

Returns detailed information about missing indexes. Used in conjunction with the other sys.xxx_missing_inde_xxx DMVs and functions.

sys.dm_db_missing_index_group_stats:

Returns summary information about groups of missing indexes. Used in conjunction with the other sys.xxx_missing_inde_xxx DMVs and functions.

sys.dm_db_missing_index_groups:

Returns information showing which specific missing indexes are contained in a specific missing index group. Used in conjunction with the other sys.xxx_missing_inde_xxx DMVs and functions.

-----I/O:

sys.dm_io_backup_tapes:

Returns information about tape devices and the status of mount requests for backups.

sys.dm_io_cluster_shared_drives:

Similar to fn_servershareddrives in SQL Server 2000, this DMV returns the drive name of each of the shared drives on a clustered server. Only returns data for clustered instances of SQL Server.

sys.dm_io_pending_io_requests:

Returns one row for each pending I/O request, possibly returning large result sets on very active SQL Servers.

sys.dm_io_virtual_file_stats:

Similar to fn_virtualfilestats in SQL Server 2000, this DMV function returns I/O statistics for data and log files for one or all databases and one or all files within the database(s).

Annexure II

System Stored Procedures

-----DATABASE ENGINE STORED PROCEDURES:

Create and store a user defined message inside the instance.

```
sp_addmessage  
[ @msgnum = ] msg_id ,  
[ @severity = ] severity ,  
[ @msgtext = ] 'msg'  
[ , [ @lang = ] 'language' ]  
[ , [ @with_log = ] 'with_log' ]  
[ , [ @replace = ] 'replace' ]
```

Attach a database file(s) to the instance.

```
sp_attach_db  
[ @dbname= ] 'dbname'  
, [ @filename1= ] 'filename_n' [ ,...16 ]
```

Attach a database file to the instance. This procedure is most useful when no associated log file is available.

```
sp_attach_single_file_db  
[ @dbname= ] 'dbname'  
, [ @physname= ] 'physical_name'
```

Display or alter instance configuration settings for the instance.

```
sp_configure  
[ [ @configname = ] 'option_name'  
[ , [ @configvalue = ] 'value' ] ]
```

Cycle the error log as if the server had been restarted.

```
sp_cycle_errorlog
```

Display list of supported data types for the instance.

```
sp_datatype_info  
[ [ @data_type = ] data_type ]  
[ , [ @ODBCVer = ] odbc_version ]
```

Alters database behaviors to be compatible with previous versions.

sp_dbcmptlevel

```
[ [ @dbname = ] name ]  
[ , [ @new_cmptlevel = ] version ]
```

Display or alter configuration settings for a database.

sp_dboption

```
[ [ @dbname = ] 'database' ]  
[ , [ @optname = ] 'option_name' ]  
[ , [ @optvalue = ] 'value' ]
```

Deletes backup and restore entries in the msdb database for entries older than the specified date.

sp_delete_backuphistory [@oldest_date =] 'oldest_date'

Detach a database from the instance.

sp_detach_db

```
[ @dbname= ] 'dbname'  
[ , [ @skipchecks= ] 'skipchecks' ]  
[ , [ @KeepFulltextIndexFile= ] 'KeepFulltextIndexFile' ]
```

Return information about the specified database object, or all objects.

sp_help [[@objname =] 'name']

Return information about the specified database, or all databases.

sp_helpdb [[@dbname=] 'name']

Return information about the specified backup device, or all backup devices.

sp_helpdevice [[@devname =] 'name']

Return name of the DLL for the specified extended stored procedure, or all extended stored procedures.

sp_helpextendedproc

```
[ [@funcname = ] 'procedure' ]
```

Return information on the specified database filename, or all database files.

sp_helpfile [[@filename =] 'name']

Return information on the specified filegroup, or all filegroups.

sp_helpfilegroup

```
[ [ @filegroupname = ] 'name' ]
```

Return information regarding the indexes on the specified table or view.

sp_helpindex

[@objname =] 'name'

Return information regarding the remote, linked, or replication servers for the current instance.

sp_helpserver

[[@server =] 'server']

[, [@optname =] 'option']

[, [@show_topology =] 'show_topology']

Alter the name of the specified database.

sp_renamedb [@dbname =] 'old_name' , [@newname =] 'new_name'

Return statistical information about the current instance.

sp_monitor

Configure locking options for specified table or index.

sp_indexoption

[@IndexNamePattern =] 'table_or_index_name'

, [@OptionName =] 'option_name'

, [@OptionValue =] 'value'

Return information about locks.

sp_lock

[[@spid1 =] 'spid1'] [,[@spid2 =] 'spid2']

Return space information for the specified object or current database.

sp_spaceused

[[@objname =] 'objname']

[,[@updateusage =] 'updateusage']

Configure options for specified table.

sp_tableoption

[@TableNamePattern =] 'table'

, [@OptionName =] 'option_name'

, [@OptionValue =] 'value'

Execute UPDATE STATISTICS against all tables in the current database.

sp_updatestats

```
[ [ @resample = ] 'resample']
```

-----SECURITY STORED PROCEDURES:

Map a server login to a database user.

sp_addalias

```
[ @loginame = ] 'login' ,  
[ @name_in_db = ] 'alias_user'
```

Create an application role in the current database.

sp_addapprole

```
[ @rolename = ] 'role' ,  
[ @password = ] 'password'
```

Create a group (i.e., role) in the current database.

sp_addgroup

```
[ @grpname = ] 'group'
```

Create new SQL Server login.

sp_addlogin

```
[ @loginame = ] 'login'  
[ , [ @passwd = ] 'password' ]  
[ , [ @defdb = ] 'database' ]  
[ , [ @deflanguage = ] 'language' ]  
[ , [ @sid = ] sid ]  
[ , [ @encryptopt= ] 'encryption_option' ]
```

Create and map a remote login to a local login.

sp_addremotelogin

```
[ @remoteserver = ] 'remoteserver'  
[ , [ @loginame = ] 'login' ]  
[ , [ @remotename = ] 'remote_name' ]
```

Create a role in the current database.

sp_addrole

```
[ @rolename = ] 'role'  
[ , [ @ownername = ] 'owner' ]
```

Add a database user, database role, or server login to the specified role.

sp_addrolemember

```
[ @rolename = ] 'role',  
[ @membername = ] 'security_account'
```

Create a remote server entry for the local instance.

sp_addserver
[@server =] 'server'
[, [@local=] 'local']
[, [@duplicate_ok =] 'duplicate_OK']

Add a server login as a member to the specified fixed server role.

sp_addsrvrolemember
[@loginame=] 'login'
, [@rolename =] 'role'

Add an existing server login as a user in the current database.

sp_adduser
[@loginame =] 'login'
[, [@name_in_db =] 'user']
[, [@grpname =] 'role']

Map an existing database user to a SQL Server login.

sp_change_users_login
[@action =] 'action'
[, [@UserNamePattern =] 'user']
[, [@LoginName =] 'login']
[, [@password =] 'password']

Remove database user from the current database.

sp_dropuser [sp_dropuser] 'user'

Create a server login for a domain user or group.

sp_grantlogin [@loginame=] 'login'

Return list of all roles for the current database.

sp_helpdbfixedrole [[@rolename =] 'role']

Return information for the specified remote or linked server, or all defined remote and linked servers.

sp_helplinkedsrvlogin
[[@rmtsrvname =] 'rmtsrvname']
[, [@locallogin =] 'locallogin']

Return information for the specified server login, or for all server logins.

sp_helplogins [[@LoginNamePattern =] 'login']

Return information for the specified remote server logins, or for all remote server logins, defined on the local server.

```
sp_helpremotelogin
[ [ @remoteserver = ] 'remoteserver' ]
[ , [ @remotename = ] 'remote_name' ]
```

Remove specified database user from the current database.

```
sp_revokedbaccess
[ @name_in_db = ] 'name'
```

Remove domain user or group as a server login.

```
sp_revokelogin
[ @loginame= ] 'login'
```

-----GENERAL EXTENDED STORED PROCEDURES:

Spawn a command shell and execute the command string.

```
xp_cmdshell
{ 'command_string' } [ , no_output ]
```

Log event in the instance error log and the server event log.

```
xp_logevent
{ error_number , 'message' } [ , 'severity' ]
```

Return information for the specified login or all logins.

```
xp_logininfo
[[@acctname =] 'account_name']
[,[@option =] 'all' | 'members']
[,[@privilege=] variable_name OUTPUT]
```

Return version information for the current instance.

```
xp_msver [ optname ]
```

Procedures for All Types of Replication

Procedure	Description
sp_addscriptexec	Posts a Microsoft SQL Server script (.sql file) to all Subscribers of a publication.
sp_adjustpublisheridentityrange	Adjusts the identity range on a publication and reallocates new ranges based on the threshold value on the publication.
sp_changereplicationserverpasswords	Changes stored passwords for the Microsoft Windows account or SQL Server login used by replication agents when connecting to servers in a replication topology. You would normally have to change a password for each individual agent running at a server, even if they all use the same login or account. This stored procedure enables you to change the password for all instances of a given SQL Server login or Windows account used by all replication agents that run at a server.
sp_removedbreplication	Removes all replication objects from a database. This stored procedure is executed at the Publisher on the publication database or at the Subscriber, on the subscription database. When executed at the Publisher on the publication database, an attempt is made to remove objects related to the published database at the Distributor and Subscriber.
sp_removedistpublisherdbreplication	Removes publishing metadata belonging to a specific publication at the Distributor.
sp_replmonitorhelppublication	Returns current status information for one or more publications at a Publisher.
sp_replmonitorhelppublicationthresholds	Returns the threshold metrics set for a monitored publication.
sp_replmonitorhelppublisher	Returns current status information for one or more Publishers.
sp_replmonitorhelpsubscription	Returns current status information for subscriptions belonging to one or more publications at the Publisher and returns one row for each returned subscription.
sp_table_validation	Either returns row count or checksum information on a table or indexed view, or compares the provided row count or checksum information with the specified table or indexed view.

Procedures for Transactional Replication

Procedure	Description
sp_article_validation	Initiates a data validation request for the specified article.
sp_marksubscriptionvalidation	Marks the current open transaction to be a subscription level validation transaction for the specified Subscriber.
sp_publication_validation	Initiates an article validation request for each article in the specified publication.
sp_browsereplcmds	Returns a result set in a readable version of the replicated commands stored in the distribution database.
sp_helppeerrequests	Returns information on all status requests received by participants in a peer-to-peer replication topology, where these requests were initiated by executing sp_requestpeerresponse at any published database in the topology.
sp_helppeerresponses	Returns all responses to a specific status request received from a participant in a peer-to-peer replication topology, where the request was initiated by executing sp_requestpeerresponse at any published database in the topology.
sp_requestpeerresponse	When executed from a node in a peer-to-peer topology, this procedure requests a response from every other node in the topology.
sp_deletepeerrequesthistory	Deletes history related to a publication status request in a peer-to-peer replication topology.
sp_posttracertoken	This procedure posts a tracer token into the transaction log at the Publisher and begins the process of tracking latency statistics. Information is recorded when the tracer token is written to the transaction log, when it is picked up by the Log Reader Agent, and when it is applied by the Distribution Agent.
sp_helptracertokens	Returns one row for each tracer token that has been inserted into a publication to determine latency.
sp_helptracertokenhistory	Returns detailed latency information for specified tracer tokens, with one row being returned for each Subscriber.

sp_deletetracertokenhistory	Removes tracer token records from the MStracer_tokens and MStracer_history system tables.
sp_replcmds	This procedure is used by the Log Reader Agent. It returns information about the publication database from which it is executed. It allows you to view transactions that currently are not distributed (those transactions remaining in the transaction log that have not been sent to the Distributor).
sp_replcounters	Returns replication statistics about latency, throughput, and transaction count for each published database.
sp_repldone	Updates the record that identifies the last distributed transaction of the server.
sp_replflush	Article definitions are stored in the cache for efficiency. This procedure is used by other replication stored procedures whenever an article definition is modified or dropped.
sp_replshowcmds	Returns the commands for transactions marked for replication in readable format.
sp_repltrans	Returns a result set of all the transactions in the publication database transaction log that are marked for replication but have not been marked as distributed.
sp_setsubscriptionxactseqno	Used to specify the log sequence number (LSN) of the next transaction to be applied by the Distribution Agent at the Subscriber, which enables the agent to skip a failed transaction.
sp_helpsubscriptionerrors	Returns all transactional replication errors for a given subscription.
sp_replmonitorsubscriptionpendingcmds	Returns information on the number of pending commands for a subscription to a transactional publication and an estimate of how much time it takes to process them.
sp_replqueuemonitor	Lists the queue messages for queued updating subscriptions.

Procedures for Merge Replication

Procedure	Description
sp_showpendingchanges	Returns a result set showing an approximate number of changes that are waiting to be replicated.
sp_showrowreplicainfo	Displays information about a row in a table that is being used as an article in merge replication.
sp_enumeratependingsschemachanges	Returns a list of all pending schema changes. This stored procedure can be used with sp_markpendingsschemachange .
sp_markpendingsschemachange	Enables an administrator to skip selected pending schema changes so that they are not replicated.
sp_addtabletocontents	Inserts references into the merge tracking tables for any rows in a source table that are not currently included in the tracking tables.
sp_deletemergeconflictrow	Deletes rows from merge conflict tables.
sp_helpmergearticleconflicts	Returns the articles in the publication that have conflicts.
sp_helpmergeconflictrows	Returns the rows in the specified conflict table.
sp_helpmergedeleteconflictrows	Returns information on data rows that lost delete conflicts.
sp_mergemetadataretentioncleanup	Performs a manual cleanup of metadata in the MSmerge_genhistory , MSmerge_contents and MSmerge_tombstone system tables.
sp_replmonitorhelpmergesession	Returns information on past sessions for a given replication Merge Agent.
sp_replmonitorhelpmergesessiondetail	Returns detailed, article-level information on a specific replication Merge Agent session.
sp_validatemergepublication	Performs a publication-wide validation.
sp_validatemergesubscription	Performs a validation for the specified subscription.

Annexure III

Useful Scripts

To find all active distributed transactions on the SQL Server instance:

```
SELECT *FROM sys.dm_tran_active_transactions  
WHERE transaction_type = 4;
```

To see blocked and blocking transactions on the server:

```
SELECT t.resource_type,  
t.resource_database_id,  
t.resource_associated_entity_id,  
t.request_mode,  
t.request_session_id,  
w.blocking_session_id  
FROM sys.dm_tran_locks as t1  
INNER JOIN sys.dm_os_waiting_tasks AS w  
ON t.lock_owner_address = w.resource_address;
```

To see the percentage of failed background jobs for all executed queries:

```
SELECT CASE ended_count WHEN 0  
THEN 'No jobs ended'  
ELSE CAST((failed_lock_count + failed_giveup_count + failed_other_count) /  
CAST(ended_count AS float) * 100 AS varchar(20)) END AS percent_failed  
FROM sys.dm_exec_background_job_queue_stats;
```

To see the SQL text of all cached plans who have been used at least three times:

```
SELECT usecounts, cacheobjtype, objtype, text  
FROM sys.dm_exec_cached_plans  
CROSS APPLY sys.dm_exec_sql_text(plan_handle)  
WHERE usecounts >= 3  
ORDER BY usecounts DESC;
```

To find any cursors that has been open for more than 24 hours:

```
SELECT name, cursor_id, creation_time, c.session_id, login_name  
FROM sys.dm_exec_cursors(0) AS c  
JOIN sys.dm_exec_sessions AS s  
ON c.session_id = s.session_id  
WHERE DATEDIFF(hh, c.creation_time, GETDATE()) > 24;
```

To find the fraction of optimized queries containing a hint:

```
SELECT (SELECT CAST (occurrence AS float)
       FROM sys.dm_exec_query_optimizer_info WHERE counter = 'hints') /
     (SELECT CAST (occurrence AS float)
       FROM sys.dm_exec_query_optimizer_info WHERE counter = 'hints')
      AS [Fraction Containing Hints];
```

To see the top 5 SQL statements executing on the server by CPU time:

```
SELECT TOP 5 total_worker_time/execution_count AS AVG_CPU_Time,
       SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
       ((CASE qs.statement_end_offset
          WHEN -1 THEN DATALENGTH(st.text)
        ELSE qs.statement_end_offset
       END - qs.statement_start_offset)/2) + 1) AS statement_text
      FROM sys.dm_exec_query_stats AS qs
      CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st
     ORDER BY total_worker_time/execution_count DESC;
```

To find the backup location:

```
select b.database_name,a.physical_device_name from msdb.dbo.backupmediafamily a join
msdb.dbo.backupset b on (a.media_set_id=b.media_set_id)
```

To find the job last run date and next run date—frequency:

```
select a.name,(select top 1 c.run_date from sysjobhistory C Where c.job_id = a.Job_ID Order by
c.run_date Desc) As run_date,b.next_run_date from sysjobs a join sysjobschedules b on
a.job_id=b.job_id where a.enabled=1
```

Finding Last Backup Time for All Database:

```
SELECT sdb.Name AS DatabaseName, COALESCE(CONVERT(VARCHAR(12),
MAX(bus.backup_finish_date), 101),'-') AS LastBackUpTime FROM sys.sysdatabases sdb LEFT OUTER
JOIN msdb.dbo.backupset bus ON bus.database_name = sdb.name GROUP BY sdb.Name
```

To Get All the Information of Database :

```
SELECT database_id, CONVERT(VARCHAR(25), DB.name) AS dbName, CONVERT(VARCHAR(10),
DATABASEPROPERTYEX(name, 'status') AS [Status], state_desc, (SELECT COUNT(1) FROM
sys.master_files WHERE DB_NAME(database_id) = DB.name AND type_desc = 'rows') AS DataFiles,
(SELECT SUM((size*8)/1024) FROM sys.master_files WHERE DB_NAME(database_id) = DB.name AND
type_desc = 'rows') AS [Data MB], (SELECT COUNT(1) FROM sys.master_files WHERE
DB_NAME(database_id) = DB.name AND type_desc = 'log') AS LogFiles, (SELECT SUM((size*8)/1024)
FROM sys.master_files WHERE DB_NAME(database_id) = DB.name AND type_desc = 'log') AS [Log MB],
user_access_desc AS [User access], recovery_model_desc AS [Recovery model], CASE
compatibility_level
```

```

WHEN 90 THEN '90 (SQL Server 2005)'

WHEN 100 THEN '100 (SQL Server 2008)'

WHEN 110 THEN '110 (SQL Server 2012)'

END AS [compatibility level], CONVERT(VARCHAR(20), create_date, 103) + ' ' +
CONVERT(VARCHAR(20), create_date, 108) AS [Creation date], -- last backup ISNULL((SELECT TOP 1
CASE TYPE WHEN 'D' THEN 'Full' WHEN 'I' THEN 'Differential' WHEN 'L' THEN 'Transaction log' END + ' - ' +
LTRIM(ISNULL(STR(ABS(DATEDIFF(DAY, GETDATE(), Backup_finish_date)))) + ' days ago', 'NEVER'))
+ ' - ' + CONVERT(VARCHAR(20), backup_start_date, 103) + ' ' + CONVERT(VARCHAR(20),
backup_start_date, 108) + ' - ' + CONVERT(VARCHAR(20), backup_finish_date, 103) + ' ' +
CONVERT(VARCHAR(20), backup_finish_date, 108) + ' ' (' + CAST(DATEDIFF(second,
BK.backup_start_date, BK.backup_finish_date) AS VARCHAR(4)) + ' ' + 'seconds') FROM
msdb..backupset BK WHERE BK.database_name = DB.name ORDER BY backup_set_id DESC,'-) AS
[Last backup], CASE WHEN is_fulltext_enabled = 1 THEN 'Fulltext enabled' ELSE " END AS [fulltext],
CASE WHEN is_auto_close_on = 1 THEN 'autoclose' ELSE " END AS [autoclose],
page_verify_option_desc AS [page verify option], CASE WHEN is_read_only = 1 THEN 'read only' ELSE "
END AS [read only], CASE WHEN is_auto_shrink_on = 1 THEN 'autoshrink' ELSE " END AS [autoshrink],
CASE WHEN is_auto_create_stats_on = 1 THEN 'auto create statistics' ELSE " END AS [auto create
statistics], CASE WHEN is_auto_update_stats_on = 1 THEN 'auto update statistics' ELSE " END AS [auto
update statistics], CASE WHEN is_in_standby = 1 THEN 'standby' ELSE " END AS [standby], CASE
WHEN is_cleanly_shutdown = 1 THEN 'cleanly shutdown' ELSE " END AS [cleanly shutdown] FROM
sys.databases DB ORDER BY dbName, [Last backup] DESC, NAME

```

To determine the index size:

```

DECLARE @OBJECT_NAME VARCHAR(255) = 'HumanResources.Shift';

DECLARE @temp TABLE ( indexID BIGINT, objectId BIGINT, index_name NVARCHAR(MAX),
used_page_count BIGINT, pages BIGINT )

--Insert into temp table

INSERT INTO @temp

SELECT P.index_id, P.OBJECT_ID , I.name, SUM (used_page_count), SUM (CASE WHEN (p.index_id < 2)
THEN (in_row_data_page_count + lob_used_page_count + row_overflow_used_page_count) ELSE
lob_used_page_count + row_overflow_used_page_count END ) FROM sys.dm_db_partition_stats P
INNER JOIN sys.indexes I ON I.index_id = P.index_id AND I.OBJECT_ID = P.OBJECT_ID WHERE
p.OBJECT_ID = OBJECT_ID(@OBJECT_NAME) GROUP BY P.index_id, I.Name, P.OBJECT_ID; SELECT
index_name INDEX_NAME, LTRIM (STR ((CASE WHEN used_page_count > pages THEN
(used_page_count - pages) ELSE 0 END) * 8, 15, 0) + ' KB') INDEX_SIZE FROM @temp T

```

GO

Table and Index information:

```
SELECT object_name(i.object_id) as objectName, i.[name] as indexName, sum(a.total_pages) as totalPages, sum(a.used_pages) as usedPages, sum(a.data_pages) as dataPages, (sum(a.total_pages) * 8 ) / 1024 as totalSpaceMB, (sum(a.used_pages) * 8 ) / 1024 as usedSpaceMB, (sum(a.data_pages) * 8 ) / 1024 as dataSpaceMB FROM sys.indexes I INNER JOIN sys.partitions p ON i.object_id = p.object_id AND i.index_id = p.index_id INNER JOIN sys.allocation_units a ON p.partition_id = a.container_id GROUP BY i.object_id, i.index_id, i.[name] ORDER BY sum(a.total_pages) DESC, object_name(i.object_id)
```

GO

Currently Running Queries:

It helps you find which queries are currently running and displays the SQL text for each currently running query. This is useful when you try to determine what is currently running in terms of T-SQL code, and not just SPIDs / session_ids.

```
select r.session_id ,r.status ,substring(qt.text,r.statement_start_offset/2,  
(case when r.statement_end_offset = -1 then len(convert(nvarchar(max), qt.text)) * 2  
else r.statement_end_offset end - r.statement_start_offset)/2) as query_text ,qt.dbid, qt.objectid  
,r.cpu_time ,r.total_elapsed_time ,r.reads ,r.writes ,r.logical_reads ,r.scheduler_id  
from sys.dm_exec_requests as r  
cross apply sys.dm_exec_sql_text(sql_handle) as qt  
inner join sys.dm_exec_sessions as es on r.session_id = es.session_id  
where es.is_user_process = 1  
order by r.cpu_time desc
```

Who Is Waiting?

```
Select * , 1 as sample , getdate() as sample_time into #waiting_tasks from sys.dm_os_waiting_tasks  
waitfor delay '00:00:10'  
insert #waiting_tasks select * , 2 , getdate() from sys.dm_os_waiting_tasks  
-- figure out the deltas
```

```
select w1.session_id , w1.exec_context_id ,w2.wait_duration_ms - w1.wait_duration_ms as d_wait_duration , w1.wait_type , w2.wait_type , datediff(ms, w1.sample_time, w2.sample_time) as
```

```

interval_ms from #waiting_tasks as w1 inner join #waiting_tasks as w2 on w1.session_id = w2.session_id and w1.exec_context_id = w2.exec_context_id where w1.sample = 1 and w2.sample = 2
order by 3 desc

-- select * from #waiting_tasks

drop table #waiting_tasks

```

Wait Stats

Samples the wait stats to see what has changed over the sample period:

```

select *, 1 as sample, getdate() as sample_time into #wait_stats from sys.dm_os_wait_stats
waitfor delay '00:00:30'
insert #wait_stats select *, 2, getdate() from sys.dm_os_wait_stats
-- figure out the deltas
select w2.wait_type ,w2.waiting_tasks_count - w1.waiting_tasks_count as d_wtc , w2.wait_time_ms - w1.wait_time_ms as d_wtm , cast((w2.wait_time_ms - w1.wait_time_ms) as float) / cast((w2.waiting_tasks_count - w1.waiting_tasks_count) as float) as avg_wtm ,
datediff(ms, w1.sample_time, w2.sample_time) as interval
from #wait_stats as w1 inner join #wait_stats as w2 on w1.wait_type = w2.wait_type
where w1.sample = 1 and w2.sample = 2 and w2.wait_time_ms - w1.wait_time_ms > 0
and w2.waiting_tasks_count - w1.waiting_tasks_count > 0 order by 3 desc
drop table #wait_stats

```

Viewing the Locking Information

```

SELECT l.resource_type, l.resource_associated_entity_id ,OBJECT_NAME(sp.OBJECT_ID) AS ObjectName
,l.request_status, l.request_mode,request_session_id ,l.resource_description FROM sys.dm_tran_locks l
LEFT JOIN sys.partitions sp ON sp.hobt_id = l.resource_associated_entity_id
WHERE l.resource_database_id = DB_ID()

```

Viewing Blocking Information

```

SELECT      t1.resource_type      ,t1.resource_database_id      ,t1.resource_associated_entity_id
,OBJECT_NAME(sp.OBJECT_ID)    AS      ObjectName      ,t1.request_mode      ,t1.request_session_id
SQL Server 2012 DBA          593          MindQ

```

```
,t2.blocking_session_id FROM sys.dm_tran_locks as t1 JOIN sys.dm_os_waiting_tasks as t2 ON t1.lock_owner_address = t2.resource_address LEFT JOIN sys.partitions sp ON sp.hobt_id = t1.resource_associated_entity_id
```

View Queries Waiting for Memory Grants

It indicates the queries waiting for memory grants. SQL Server analyzes a query and determines how much memory it needs based on the estimated plan. If memory is not available at that time, the query is suspended until the memory required is available. If a query is waiting for a memory grant, an entry shows up in the DMV sys.dm_exec_query_memory_grants:

```
SELECT es.session_id AS SPID ,es.login_name ,es.host_name ,es.program_name, es.status AS Session_Status ,mg.requested_memory_kb ,DATEDIFF(mi, mg.request_time , GETDATE()) AS [WaitingSince-InMins] FROM sys.dm_exec_query_memory_grants mg  
JOIN sys.dm_exec_sessions es ON es.session_id = mg.session_id  
WHERE mg.grant_time IS NULL
```

Connected User Information

It tells you which users are connected, and how many sessions each of them has open:

```
SELECT login_name , count(session_id) as session_count FROM sys.dm_exec_sessions  
GROUP BY login_name
```

Query Plan and Query Text for Currently Running Queries

Use the following query to find out the query plan in XML and the query text for the currently running batch for a particular session. Make sure that you use a grid to output the result in SQL Server Management Studio. When you get the result, you can click the link for the XML plan, which opens an XML editor inside Management Studio. If you want to look at the graphical query plan from this XML plan, click on the link to the XML plan, and it opens in a new window in SSMS.

```
SELECT er.session_id ,es.login_name ,er.request_id ,er.start_time,QueryPlan_XML = (SELECT query_plan  
FROM  
sys.dm_exec_query_plan(er.plan_handle)) ,SQLText = (SELECT Text FROM  
sys.dm_exec_sql_text(er.sql_handle))  
FROM sys.dm_exec_requests er JOIN sys.dm_exec_sessions es ON er.session_id = es.session_id  
WHERE es.is_user_process = 1 ORDER BY er.start_time ASC
```

Memory Usage

It indicates the memory used, in KB, by each internal SQL Server component:

```
SELECT name ,type ,SUM(single_pages_kb + multi_pages_kb) AS MemoryUsedInKB
```

```
FROM sys.dm_os_memory_clerks GROUP BY name, type  
ORDER BY SUM(single_pages_kb + multi_pages_kb) DESC
```

Buffer Pool Memory Usage

It lists out all the objects within the buffer pool, along with the amount of space used by each. This is a great way to see who uses the Buffer Pool:

```
SELECT count(*)AS cached_pages_count ,name ,index_id FROM sys.dm_os_buffer_descriptors AS bd  
INNER JOIN  
( SELECT object_name(object_id) AS name ,index_id ,allocation_unit_id FROM sys.allocation_units AS au  
INNER JOIN sys.partitions AS p ON au.container_id = p.hobt_id AND (au.type = 1 OR au.type = 3)  
UNION ALL  
SELECT object_name(object_id) AS name ,index_id, allocation_unit_id  
FROM sys.allocation_units AS au  
INNER JOIN sys.partitions AS p  
ON au.container_id = p.partition_id AND au.type = 2 ) AS obj  
ON bd.allocation_unit_id = obj.allocation_unit_id  
WHERE database_id = db_id()  
GROUP BY name, index_id  
ORDER BY cached_pages_count DESC;
```

Annexure IV

Frequently Asked Questions

What is RDBMS?

Relational Data Base Management Systems (RDBMS) are database management systems that maintain data records and indices in tables. Relationships may be created and maintained across and among the data and tables. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage.

What is normalization?

Database normalization is a data design and organization process applied to data structures based on rules that help build relational databases. In relational database design, the process of organizing data to minimize redundancy. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

What are different normalization forms?

1NF: Eliminate Repeating Groups

Make a separate table for each set of related attributes, and give each table a primary key.
Each field contains at most one value from its attribute domain.

2NF: Eliminate Redundant Data

If an attribute depends on only part of a multi-valued key, remove it to a separate table.

3NF: Eliminate Columns Not Dependent On Key

If attributes do not contribute to a description of the key, remove them to a separate table. All attributes must be directly dependent on the primary key

BCNF: Boyce-Codd Normal Form

If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.

4NF: Isolate Independent Multiple Relationships

No table may contain two or more 1:n or n:m relationships that are not directly related.

5NF: Isolate Semantically Related Multiple Relationships

There may be practical constraints on information that justify separating logically related many-to-many relationships.

ONF: Optimal Normal Form

A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.

DKNF: Domain-Key Normal Form

A model free from all modification anomalies.

Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database.

What is Stored Procedure?

A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.

e.g. sp_helpdb, sp_renamedb, sp_depends etc.

What is Trigger?

A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs. Triggers are stored in and managed by the DBMS. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. A trigger cannot be called or executed; the DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

Nested Trigger: A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

What is View?

A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does data in the view, as views are the way to look at part of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.

What is Index?

An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the

indexes, they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.

Clustered indexes define the physical sorting of a database table's rows in the storage media. For this reason, each database table may have only one clustered index.

Non-clustered indexes are created outside of the database table and contain a sorted list of references to the table itself.

What is the difference between clustered and a non-clustered index?

A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.

A nonclustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a nonclustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

What are the different index configurations a table can have?

A table can have one of the following index configurations:

- No indexes
- A clustered index
- A clustered index and many nonclustered indexes
- A nonclustered index
- Many nonclustered indexes

What is the use of DBCC commands?

DBCC stands for database consistency checker. We use these commands to check the consistency of the databases, i.e., maintenance, validation task and status checks. E.g. DBCC CHECKDB - Ensures that tables in the db and the indexes are correctly linked. DBCC CHECKALLOC - To check that all pages in a db are correctly allocated. DBCC CHECKFILEGROUP - Checks all tables file group for any damage.

What is a Linked Server?

Linked Servers is a concept in SQL Server by which we can add other SQL Server to a Group and query both the SQL Server dbs using T-SQL Statements. With a linked server, you can create very clean, easy to follow, SQL statements that allow remote data to be retrieved, joined and combined with local data.

Storped Procedure sp_addlinkedserver, sp_addlinkedsrvlogin will be used add new Linked Server.

What is Collation?

Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case-sensitivity, accent marks, kana character types and character width.

What are different type of Collation Sensitivity?

Case sensitivity

A and a, B and b, etc.

Accent sensitivity

a and á, o and ó, etc.

Kana Sensitivity

When Japanese kana characters Hiragana and Katakana are treated differently, it is called Kana sensitive.

Width sensitivity

When a single-byte character (half-width) and the same character when represented as a double-byte character (full-width) are treated differently then it is width sensitive.

What's the difference between a primary key and a unique key?

Both primary key and unique enforce uniqueness of the column on which they are defined. But by default primary key creates a clustered index on the column, where unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.

What is a NOLOCK?

Using the NOLOCK query optimiser hint is generally considered good practice in order to improve concurrency on a busy system. When the NOLOCK hint is included in a SELECT statement, no locks are taken when data is read. The result is a Dirty Read, which means that another process could be updating the data at the exact time you are reading it. There are no guarantees that your query will retrieve the most recent data. The advantage to performance is that your reading of data will not block updates from taking place, and updates will not block your reading of data. SELECT statements take Shared (Read) locks. This means that multiple SELECT statements are allowed simultaneous access, but other processes are blocked from modifying the data. The updates will queue until all the reads have completed, and reads requested after the update will wait for the updates to complete. The result to your system is delay(blocking).

What is difference between DELETE & TRUNCATE commands?

Delete command removes the rows from a table based on the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table and there will be no data in the table after we run the truncate command.

TRUNCATE

TRUNCATE is faster and uses fewer system and transaction log resources than DELETE.

TRUNCATE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log. TRUNCATE removes all rows from a table, but the table structure and its columns, constraints, indexes and so on remain. The counter used by an identity for new rows is reset to the seed for the column.

You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint. Because TRUNCATE TABLE is not logged, it cannot activate a trigger. TRUNCATE can not be Rolled back using logs.

TRUNCATE is DDL Command.

TRUNCATE Resets identity of the table.

DELETE

DELETE removes rows one at a time and records an entry in the transaction log for each deleted row.

If you want to retain the identity counter, use DELETE instead. If you want to remove table definition and its data, use the DROP TABLE statement.

DELETE Can be used with or without a WHERE clause

DELETE Activates Triggers.

DELETE Can be Rolled back using logs.

DELETE is DML Command.

DELETE does not reset identity of the table.

When is the use of UPDATE_STATISTICS command?

This command is basically used when a large processing of data has occurred. If a large amount of deletions any modification or Bulk Copy into the tables has occurred, it has to update the indexes to take these changes into account. UPDATE_STATISTICS updates the indexes on these tables accordingly.

What is SQL Profiler?

SQL Profiler is a graphical tool that allows system administrators to monitor events in an instance of Microsoft SQL Server. You can capture and save data about each event to a file or SQL Server table to analyze later. For example, you can monitor a production environment to see which stored procedures are hampering performance by executing too slowly.

Use SQL Profiler to monitor only the events in which you are interested. If traces are becoming too large, you can filter them based on the information you want, so that only a subset of the event data is collected. Monitoring too many events adds overhead to the server and the monitoring process and can cause the trace file or trace table to grow very large, especially when the monitoring process takes place over a long period of time.

Which TCP/IP port does SQL Server run on? How can it be changed?

SQL Server runs on port 1433. It can be changed from the Network Utility TCP/IP properties –> Port number.both on client and the server.

What are the authentication modes in SQL Server? How can it be changed? Windows mode and mixed mode (SQL & Windows).

To change authentication mode in SQL Server click Start, Programs, Microsoft SQL Server and click SQL Enterprise Manager to run SQL Enterprise Manager from the Microsoft SQL Server program group. Select the server then from the Tools menu select SQL Server Configuration Properties, and choose the Security page.

Where are SQL server users names and passwords are stored in sql server?
They get stored in master db in the syslogins table.

Which command using Query Analyzer will give you the version of SQL server and operating system?

```
SELECT SERVERPROPERTY('productversion'), SERVERPROPERTY ('productlevel'),
```

What is SQL server agent?

SQL Server agent plays an important role in the day-to-day tasks of a database administrator (DBA). It is often overlooked as one of the main tools for SQL Server management. Its purpose is to ease the implementation of tasks for the DBA, with its full-function scheduling engine, which allows you to schedule your own jobs and scripts.

What is log shipping?

Log shipping is the process of automating the backup of database and transaction log files on a production SQL server, and then restoring them onto a standby server. Enterprise Editions only supports log shipping. In log shipping the transactional log file from one server is automatically updated into the backup database on the other server. If one server fails, the other server will have the same db can be used this as the Disaster Recovery plan. The key feature of log shipping is that is will automatically backup transaction logs throughout the day and automatically restore them on the standby server at defined interval.

What command do we use to rename a db?

sp_renamedb 'oldname' , 'newname'

If someone is using db it will not accept sp_renamedb. In that case first bring db to single user using sp_dboptions. Use sp_renamedb to rename database. Use sp_dboptions to bring database to multi user mode.

What is sp_configure commands and set commands?

Use sp_configure to display or change server-level settings. To change database-level settings, use ALTER DATABASE. To change settings that affect only the current user session, use the SET statement.

What are the different types of replication? Explain.

The SQL Server -supported replication types are as follows:

- Transactional
- Snapshot
- Merge

Snapshot replication distributes data exactly as it appears at a specific moment in time and does not monitor for updates to the data. Snapshot replication is best used as a method for replicating data that changes infrequently or where the most up-to-date values (low latency) are not a requirement. When synchronization occurs, the entire snapshot is generated and sent to Subscribers.

Transactional replication, an initial snapshot of data is applied at Subscribers, and then when data modifications are made at the Publisher, the individual transactions are captured and propagated to Subscribers.

Merge replication is the process of distributing data from Publisher to Subscribers, allowing the Publisher and Subscribers to make updates while connected or disconnected, and then merging the updates between sites when they are connected.

What are the OS services that the SQL Server installation adds?

MS SQL SERVER SERVICE, SQL AGENT SERVICE, DTC (Distribution transac co-ordinator)

What are three SQL keywords used to change or set someone's permissions?
GRANT, DENY, and REVOKE.

What does it mean to have quoted_identifier on? What are the implications of having it off?

When SET QUOTED_IDENTIFIER is ON, identifiers can be delimited by double quotation marks, and literals must be delimited by single quotation marks. When SET QUOTED_IDENTIFIER is OFF, identifiers cannot be quoted and must follow all Transact-SQL rules for identifiers.

How to rebuild Master Database?

Shutdown Microsoft SQL Server 2000, and then run Rebuildm.exe. This is located in the Program Files\Microsoft SQL Server\80\Tools\Binn directory.

In the Rebuild Master dialog box, click Browse.

In the Browse for Folder dialog box, select the \Data folder on the SQL Server 2000 compact disc or in the shared network directory from which SQL Server 2000 was installed, and then click OK.

Click Settings. In the Collation Settings dialog box, verify or change settings used for the master database and all other databases.

Initially, the default collation settings are shown, but these may not match the collation selected during setup. You can select the same settings used during setup or select new collation settings. When done, click OK.

In the Rebuild Master dialog box, click Rebuild to start the process.

The Rebuild Master utility reinstalls the master database.

To continue, you may need to stop a server that is running.

Source: [http://msdn2.microsoft.com/en-us/library/aa197950\(SQL.80\).aspx](http://msdn2.microsoft.com/en-us/library/aa197950(SQL.80).aspx)

What are the basic functions for master, msdb, model, tempdb and resource databases?

The master database holds information for all databases located on the SQL Server instance and is the glue that holds the engine together. Because SQL Server cannot start without a functioning master database, you must administer this database with care.

The msdb database stores information regarding database backups, SQL Agent information, DTS packages, SQL Server jobs, and some replication information such as for log shipping.

The tempdb holds temporary objects such as global and local temporary tables and stored procedures.

The model is essentially a template database used in the creation of any new user database created in the instance.

The Resource Database is a read-only database that contains all the system objects that are included with SQL Server. SQL Server system objects, such as sys.objects, are physically persisted in the Resource database, but they logically appear in the sys schema of every database. The Resource database does not contain user data or user metadata.

What is data integrity? Explain constraints?

Data integrity is an important feature in SQL Server. When used properly, it ensures that data is accurate, correct, and valid. It also acts as a trap for otherwise undetectable bugs within applications.

A PRIMARY KEY constraint is a unique identifier for a row within a database table. Every table should have a primary key constraint to uniquely identify each row and only one primary key constraint can be created for each table. The primary key constraints are used to enforce entity integrity.

A UNIQUE constraint enforces the uniqueness of the values in a set of columns, so no duplicate values are entered. The unique key constraints are used to enforce entity integrity as the primary key constraints.

A FOREIGN KEY constraint prevents any actions that would destroy links between tables with the corresponding data values. A foreign key in one table points to a primary key in another table. Foreign keys prevent actions that would leave rows with foreign key values when there are no primary keys with that value. The foreign key constraints are used to enforce referential integrity.

A CHECK constraint is used to limit the values that can be placed in a column. The check constraints are used to enforce domain integrity.

A NOT NULL constraint enforces that the column will not accept null values. The not null constraints are used to enforce domain integrity, as the check constraints.

What is a table called, if it does not have neither Cluster nor Non-cluster Index? What is it used for?

Unindexed table or Heap. Microsoft Press Books and Book On Line (BOL) refers it as Heap. A heap is a table that does not have a clustered index and, therefore, the pages are not linked by pointers. The IAM pages are the only structures that link the pages in a table together.

Unindexed tables are good for fast storing of data. Many times it is better to drop all indexes from table and than do bulk of inserts and to restore those indexes after that.

What is BCP? When does it used?

BulkCopy is a tool used to copy huge amount of data from tables and views. BCP does not copy the structures same as source to destination.

How do you load large data to the SQL server database?

BulkCopy is a tool used to copy huge amount of data from tables. BULK INSERT command helps to Imports a data file into a database table or view in a user-specified format.

Can SQL Servers linked to other servers like Oracle?

SQL Server can be linked to any server provided it has OLE-DB provider from Microsoft to allow a link. E.g. Oracle has a OLE-DB provider for oracle that Microsoft provides to add it as linked server to SQL Server group.

How to know which index a table is using?

```
SELECT table_name,index_name FROM user_constraints
```

How to copy the tables, schema and views from one SQL server to another?

Microsoft SQL Server 2000 Data Transformation Services (DTS) is a set of graphical tools and programmable objects that lets user extract, transform, and consolidate data from disparate sources into single or multiple destinations.

What is an execution plan? When would you use it? How would you view the execution plan?

An execution plan is basically a road map that graphically or textually shows the data retrieval methods chosen by the SQL Server query optimizer for a stored procedure or ad-hoc query and is a very useful tool for a developer to understand the performance characteristics of a query or stored procedure since the plan is the one that SQL Server will place in its cache and use to execute the stored procedure or query. From within Query Analyzer is an option called "Show Execution Plan" (located on the Query drop-down menu). If this option is turned on it will display query execution plan in separate window when query is ran again.

What is Data Compression?

In SQL SERVER 2008 Data Compression comes in two flavors:

Row Compression

Page Compression

Row Compression

Row compression changes the format of physical storage of data. It minimizes the metadata (column information, length, offsets etc) associated with each record. Numeric data types and fixed length strings are stored in variable-length storage format, just like Varchar. (.)

Page Compression

Page compression allows common data to be shared between rows for a given page. It uses the following techniques to compress data:

Row compression.

Prefix Compression. For every column in a page duplicate prefixes are identified. These prefixes are saved in compression information headers (CI) which resides after page header. A

reference number is assigned to these prefixes and that reference number is replaced wherever those prefixes are being used.

Dictionary Compression.

Dictionary compression searches for duplicate values throughout the page and stores them in CI. The main difference between prefix and dictionary compression is that prefix is only restricted to one column while dictionary is applicable to the complete page.

What is Filestream?

Filestream allows you to store large objects in the file system and have these files integrated within the database. It enables SQL Server based applications to store unstructured data such as documents, images, audios, videos etc. in the file system. FILESTREAM basically integrates the SQL Server Database Engine with New Technology File System (NTFS); it basically stores the data in varbinary (max) data type. Using this data type, the unstructured data is stored in the NTFS file system and the SQL Server Database Engine manages the link between the Filestream column and the actual file located in the NTFS. Using Transact SQL statements users can insert, update, delete and select the data stored in FILESTREAM enabled tables.

What is Dirty Read?

A dirty read occurs when two operations say, read and write occurs together giving the incorrect or unedited data. Suppose, A has changed a row, but has not committed the changes. B reads the uncommitted data but his view of the data may be wrong so that is Dirty Read.

What is SQLCMD?

sqlcmd is enhanced version of the isql and osql and it provides way more functionality than other two options. In other words sqlcmd is better replacement of isql (which will be deprecated eventually) and osql (not included in SQL Server 2005 RTM). sqlcmd can work two modes - i) BATCH and ii) interactive modes.

UNION ALL

The UNION ALL command is equal to the UNION command, except that UNION ALL selects all values.

The difference between Union and Union all is that Union all will not eliminate duplicate rows, instead it just pulls all rows from all tables fitting your query specifics and combines them into a table. (.)

What is B-Tree?

The database server uses a B-tree structure to organize index information. B-Tree generally has following types of index pages or nodes:

root node: A root node contains node pointers to branch nodes which can be only one.

branch nodes: A branch node contains pointers to leaf nodes or other branch nodes which can be two or more.

leaf nodes: A leaf node contains index items and horizontal pointers to other leaf nodes which can be many.

What is Service Broker?

Service Broker is a message-queuing technology in SQL Server that allows developers to integrate SQL Server fully into distributed applications. Service Broker is a feature which provides facility to SQL Server to send an asynchronous, transactional message. It allows a database to send a message to another database without waiting for the response, so the application will continue to function if the remote database is temporarily unavailable.

What is Policy Management?

Policy Management in SQL SERVER 2008 allows you to define and enforce policies for configuring and managing SQL Server across the enterprise. Policy-Based Management is configured in SQL Server Management Studio (SSMS). Navigate to the Object Explorer and expand the Management node and the Policy Management node; you will see the Policies, Conditions, and Facets nodes. (.)

What is Replication and Database Mirroring?

Database mirroring can be used with replication to provide availability for the publication database. Database mirroring involves two copies of a single database that typically reside on different computers. At any given time, only one copy of the database is currently available to clients which are known as the principal database. Updates made by clients to the principal database are applied on the other copy of the database, known as the mirror database. Mirroring involves applying the transaction log from every insertion, update, or deletion made on the principal database onto the mirror database.

What are Sparse Columns?

A sparse column is another tool used to reduce the amount of physical storage used in a database. They are the ordinary columns that have an optimized storage for null values. Sparse columns reduce the space requirements for null values at the cost of more overhead to retrieve nonnull values.

What does TOP Operator Do?

The TOP operator is used to specify the number of rows to be returned by a query. The TOP operator has new addition in SQL SERVER 2008 that it accepts variables as well as literal values and can be used with INSERT, UPDATE, and DELETE statements.

What is CTE?

CTE is an abbreviation Common Table Expression. A Common Table Expression (CTE) is an expression that can be thought of as a temporary result set which is defined within the execution of a single SQL statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query.

Which are new data types introduced in SQL SERVER 2008?

The GEOMETRY Type: The GEOMETRY data type is a system .NET common language runtime (CLR) data type in SQL Server. This type represents data in a two-dimensional Euclidean coordinate system.

The GEOGRAPHY Type: The GEOGRAPHY datatype's functions are the same as with GEOMETRY. The difference between the two is that when you specify GEOGRAPHY, you are usually specifying points in terms of latitude and longitude.

New Date and Time Datatypes: SQL Server 2008 introduces four new datatypes related to date and time: DATE, TIME, DATETIMEOFFSET, and DATETIME2.

DATE: The new DATE type just stores the date itself. It is based on the Gregorian calendar and handles years from 1 to 9999.

TIME: The new TIME (n) type stores time with a range of 00:00:00.0000000 through 23:59:59.9999999. The precision is allowed with this type. TIME supports seconds down to 100 nanoseconds. The n in TIME (n) defines this level of fractional second precision, from 0 to 7 digits of precision.

The DATETIMEOFFSET Type: DATETIMEOFFSET (n) is the time-zone-aware version of a datetime datatype. The name will appear less odd when you consider what it really is: a date + a time + a time-zone offset. The offset is based on how far behind or ahead you are from Coordinated Universal Time (UTC) time.

The DATETIME2 Type: It is an extension of the datetime type in earlier versions of SQL Server. This new datatype has a date range covering dates from January 1 of year 1 through December 31 of year 9999. This is a definite improvement over the 1753 lower boundary of the datetime datatype. DATETIME2 not only includes the larger date range, but also has a timestamp and the same fractional precision that TIME type provides

What is RAID and what are different types of RAID configurations?

RAID stands for Redundant Array of Inexpensive Disks, used to provide fault tolerance to database servers. There are six RAID levels 0 through 5 offering different levels of performance, fault tolerance. MSDN has some information about RAID levels and for detailed information, check out the RAID advisory board's homepage

How to restart SQL Server in single user mode? How to start SQL Server in minimal configuration mode?

SQL Server can be started from command line, using the SQLSERVR.EXE. This EXE has some very important parameters with which a DBA should be familiar with. -m is used for starting SQL Server in single user mode and -f is used to start the SQL Server in minimal configuration mode. Check out SQL Server books online for more parameters and their explanations.

What is blocking and how would you troubleshoot it?

Blocking happens when one connection from an application holds a lock and a second connection requires a conflicting lock type. This forces the second connection to wait, blocked on the first. Read up the following topics in SQL Server books online: Understanding and avoiding blocking, Coding efficient transactions. Explain CREATE DATABASE syntax Many of us are used to creating databases from the Enterprise Manager or by just issuing the command: CREATE DATABASE MyDB.

What is a deadlock and what is a live lock? How will you go about resolving deadlocks?

Deadlock is a situation when two processes, each having a lock on one piece of data, attempt to acquire a lock on the other's piece. Each process would wait indefinitely for the other to release the lock, unless one of the user processes is terminated. SQL Server detects deadlocks and terminates one user's process. A livelock is one, where a request for an exclusive lock is repeatedly denied because a series of overlapping shared locks keeps interfering. SQL Server detects the situation after four denials and refuses further shared locks. A livelock also occurs when read transactions monopolize a table or page, forcing a write transaction to wait indefinitely. Check out SET DEADLOCK_PRIORITY and "Minimizing Deadlocks" in SQL Server books online. Also check out the article Q169960 from Microsoft knowledge base.

What are the steps you will take to improve performance of a poor performing query?

This is a very open ended question and there could be a lot of reasons behind the poor performance of a query. But some general issues that you could talk about would be: No indexes, table scans, missing or out of date statistics, blocking, excess recompilations of stored procedures, procedures and triggers without SET NOCOUNT ON, poorly written query with unnecessarily complicated joins, too much normalization, excess usage of cursors and temporary tables. Some of the tools/ways that help you troubleshooting performance problems are: SET SHOWPLAN_ALL ON, SET SHOWPLAN_TEXT ON, SET STATISTICS IO ON, SQL Server Profiler, Windows NT /2000 Performance monitor, Graphical execution plan in Query Analyzer.

Download the white paper on performance tuning SQL Server from Microsoft web site. Don't forget to check out sql-server-performance.com

As a part of your job, what are the DBCC commands that you commonly use for database maintenance?

DBCC CHECKDB, DBCC CHECKTABLE, DBCC CHECKCATALOG, DBCC CHECKALLOC, DBCC SHOWCONTIG, DBCC SHRINKDATABASE, DBCC SHRINKFILE etc. But there are a whole load of DBCC commands which are very useful for DBAs

What are statistics, under what circumstances they go out of date, how do you update them?

Statistics determine the selectivity of the indexes. If an indexed column has unique values then the selectivity of that index is more, as opposed to an index with non-unique values. Query optimizer uses these indexes in determining whether to choose an index or not while executing a query. Some situations under which you should update statistics: 1) If there is significant change in the key values in the index 2) If a large amount of data in an indexed column has been added, changed, or removed (that is, if the distribution of key values has changed), or the table has been truncated using the TRUNCATE TABLE statement and then repopulated 3) Database is upgraded from a previous version. Look up SQL Server books online for the following commands: UPDATE STATISTICS, STATS_DATE, DBCC SHOW_STATISTICS, CREATE STATISTICS, DROP STATISTICS, sp_autostats, sp_createstats, sp_updatestats

What are the different ways of moving data/databases between servers and databases in SQL Server?

There are lots of options available, you have to choose your option depending upon your requirements. Some of the options you have are: BACKUP/RESTORE, dettaching and attaching databases, replication, DTS, BCP, logshipping, INSERT...SELECT, SELECT...INTO, creating INSERT scripts to generate data.

How to determine the service pack currently installed on SQL Server?

The global variable @@Version stores the build number of the sqlservr.exe, which is used to determine the service pack installed. To know more about this process visit SQL Server service packs and versions.

What's the maximum size of a row?

8060 bytes. Don't be surprised with questions like 'what is the maximum number of columns per table'. 1024 columns per table. Check out SQL Server books online for the page titled: "Maximum Capacity Specifications". Explain Active/Active and Active/Passive cluster configurations. Hopefully you have experience setting up cluster servers. But if you don't, at least be familiar with the way clustering works and the two clustering configurations Active/Active and Active/Passive. SQL Server books online has enough information on this topic and there is a good white paper available on Microsoft site. Explain the architecture of SQL Server. This is a very important question and you better be able to answer it if consider yourself a DBA. SQL Server books online is the best place to read about SQL Server architecture. Read up the chapter dedicated to SQL Server Architecture.

What is a Schema in SQL Server 2005? Explain how to create a new Schema in a Database.

A schema is used to create database objects. It can be created using CREATE SCHEMA statement. The objects created can be moved between schemas. Multiple database users can share a single default schema.

CREATE SCHEMA sample;

Table creation

```
Create table sample.sampleinfo
{
id int primary key,
name varchar(20)
}
```

What are Page Splits?

When there is not enough room on a page for a new row, a Server splits the page, allocates a new page, and moves some rows to the new page.

What is SQL Server Agent?

SQL Server Agent is a Microsoft Windows service that executes scheduled administrative tasks called jobs. SQL Server Agent uses SQL Server to store job information. Jobs contain one or more job steps. We generally schedule the backups on the production databases using the SQL server agent. In SQL Server 2005 we have roles created for using SQL Server agents.

- SQLAgentUserRole
- SQLAgentReaderRole
- SQLAgentOperatorRole

SQL Server Agent for SQL Server 2005 provides a more robust security design than earlier versions of SQL Server. This improved design gives system administrators the flexibility they need to manage their Agent service.

Telephonic Interview Approach:

The phone interview. It has tales of bringing normally rationale people to a terrified state. I have even heard of a DBA that was so worried about a SQL Server phone interview that 'they just happened to be in the office park' where the company was located and actually wanted the interview face to face. The reality is, just about all organizations that I work with have a phone interview as a right of first passage in the process. The employer wants to quickly determine if the DBA candidate could be qualified for the position from a technical perspective and if they will fit into the team. As a DBA, what sorts of things should you be on the lookout for during a phone interview? What do you think the employer is expecting? Is this the technical interview or not? Should you try to avoid the phone interview all together and just 'pop-in' for a face to face interview?

Solution

Let's address the last question first, that being should you just 'pop-in' to the office rather than having a phone interview? Phone interviews are setup for a reason. They are intended as a simple means to determine if someone is worth going through the entire interview process. Some organizations have a fairly structured process and follow it closely while other organizations really conduct interviews over lunch or based on a personal contact's network. You need to be the judge and assess the situation for yourself and make the call. In the story I was told about, the results were not positive from either the employer or DBA perspective. So keep that in mind.

With that behind us, let's get into the employer and DBA views of the phone interview as well as some potential questions you should be ready to answer.

Employer's Perspective

In some respects, employers handle phone interviews in such a manner that they can use the same questions to assess the skills of the candidates as a means to compare and contrast their skills to determine a candidate ranking. By this I mean candidates in terms of best to worst skill as well as who should and should not progress to the next step in the process, the on-site interview.

From an employer's perspective, they are trying to determine a few different items during the phone interview:

- Communication skills
- Personality
- Technical experience and background
- Leadership qualities
- How they could fit into the team

DBA Perspective

Here are some thoughts from a DBA perspective when it comes to a phone interview:

- First, be ready for the phone interview and expect it as a portion of the interview process.
- Prepare for the phone interview just like you would the on-site interview. Remember if you do not make a good impression with the phone interview that the on-site interview may not be a reality.
- Remember the phone interview is a rite of passage, so first impressions can mean a great deal. Make sure your first impressions are what you want them to be. Simple items like stuttering, stumbling over your words, smoking during the call, chewing gum, etc. may turn off the interviewer quickly.
- Next, figure out your 30 second elevator pitch and make sure you outline your most important experience and skills as well as how you are going to help the organization.
- Just like with your resume, be sure you do not lie. If you do not know the answer to a question, just say you do not know.
- Be prepared for technical questions from either a technical or non-technical interviewer. Be sure to respond in a way that they can understand the response. The interviewer may be looking for just buzz words or may be not. A good technical interview, from a knowledgeable DBA, can really dig into the details to make sure you truly understand the technology.
- As much as the interview process is about the employer selecting the right employee, also keep in mind that the candidate should select the right organization for themselves. As such, use the phone interview as a means to learn about the organization. If you are given the opportunity, ask the questions you have prepared.
- The topic of salary and compensation may be discussed. Be prepared for the question. Historically, the response has been to push off the salary figures to as late in the process as possible. As an employer and employee, I disagree. I think it only makes sense to state a range to make sure one party does not have different expectations than the other. If the figures are not even close, it might make sense for either party to stop the process rather than spending a significant amount of time only to be disappointed at the end. Just something to consider.

Phone Interview Questions

Although no two phone interviews are the same, below outlines some potential questions to keep in mind as you prepare for a SQL Server DBA phone interview:

Can you explain your skill set?

- Employers look for the following:
 - DBA (Maintenance, Security, Upgrades, Performance Tuning, etc.)
 - Database developer (T-SQL, DTS, SSIS, Analysis Services, Reporting Services, Crystal Reports, etc.)

- Communication skills (oral and written)
- DBA's
 - This is your 30 second elevator pitch outlining your technical expertise and how you can benefit the organization

Can you explain the environments you have worked in related to the following items:

- SQL Server versions
- SQL Server technologies
 - Relational engine, Reporting Services, Analysis Services, Integration Services
- Number of SQL Servers
- Number of instances
- Number of databases
- Range of size of databases
- Number of DBAs
- Number of Developers
- Hardware specs (CPU's, memory, 64 bit, SANs)

What are the tasks that you perform on a daily basis and how have you automated them?

- For example, daily checks could include:
 - Check for failed processes
 - Research errors
 - Validate disk space is not low
 - Validate none of the databases are offline or corrupt
 - Perform database maintenance as available to do so
- For example, automation could include:
 - Setup custom scripts to query for particular issues and email the team
 - Write error messages centrally in the application and review that data
 - Setup Operators and Alerts on SQL Server Agent Jobs for automated job notification

What is your experience with third party applications and why would you use them?

- Experience
 - Backup tools

- Performance tools
 - Code or data synchronization
 - Disaster recovery\high availability
- Why
 - Need to improve upon the functionality that SQL Server offers natively
 - Save time, save money, better information or notification

How do you identify and correct a SQL Server performance issue?

- Identification - Use native tools like Profiler, Perfmon, system stored procedures, dynamic management views, custom stored procedures or third party tools
- Analysis - Analyze the data to determine the core problems
- Testing - Test the various options to ensure they perform better and do not cause worse performance in other portions of the application
- Knowledge sharing - Share your experience with the team to ensure they understand the problem and solution, so the issue does not occur again

What are some of the new T-SQL commands with SQL Server 2005 that you have used and what value do they offer?

- ROW_NUMBER - Means to page through a result set and only return the needed portion of the result set
- EXCEPT - The final result set where data exists in the first dataset and not in the second dataset
- INTERSECT - The final result set where values in both of the tables match
- PIVOT\UNPIVOT - Expression to flip rows to columns or vice versa
- Synonyms - Alias to an object (table, stored procedure, function, view) to maintain the original object and refer to the new object as well
- NOTE - Many more commands do exist, this is an abbreviated list.

What are the dynamic management views and what value do they offer?

The DMV's are a set of system views new to SQL Server 2005 to gain insights into particular portions of the engine

Here are some of the DMV's and the associated value:

- sys.dm_exec_query_stats and sys.dm_exec_sql_text - Buffered code in SQL Server
- sys.dm_os_buffer_descriptors

- sys.dm_tran_locks - Locking and blocking
- sys.dm_os_wait_stats - Wait stats
- sys.dm_exec_requests and sys.dm_exec_sessions - Percentage complete for a process

What is the process to upgrade from DTS to SSIS packages?

- You can follow the steps of the migration wizard but you may need to manually upgrade portions of the package that were not upgraded by the wizard
- For script related tasks, these should be upgraded to new native components or VB.NET code

What are some of the features of SQL Server 2008 that you are looking into and why are they of interest?

- Change Tracking
- Plan Guides
- SQL Data Collector
- Data Auditing
- Data compression
- NOTE - Many more new features do exist, this is an abbreviated list.

Keep in mind that these questions are primarily related to the relational engine, so a BI DBA would have a whole different set of questions. In addition, the more you know about the organization and role should guide you down a path for the types of questions you should be prepared for during the phone interview.

What is normalization? Explain different levels of normalization?

Check out the article Q100139 from Microsoft knowledge base and of course, there's much more information available in the net. It'll be a good idea to get a hold of any RDBMS fundamentals text book, especially the one by C. J. Date. Most of the times, it will be okay if you can explain till third normal form.

What is denormalization and when would you go for it?

As the name indicates, denormalization is the reverse process of normalization. It's the controlled introduction of redundancy in to the database design. It helps improve the query performance as the number of joins could be reduced.

How do you implement one-to-one, one-to-many and many-to-many relationships while designing tables?

One-to-One relationship can be implemented as a single table and rarely as two tables with primary and foreign key relationships. One-to-Many relationships are implemented by splitting the data into two tables with primary key and foreign key relationships. Many-to-Many relationships are implemented using a junction table with the keys from both the tables forming the composite primary key of the junction table. It will be a good idea to read up a database designing fundamentals text book.

What's the difference between a primary key and a unique key?

Both primary key and unique enforce uniqueness of the column on which they are defined. But by default primary key creates a clustered index on the column, where unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.

What are user defined datatypes and when you should go for them?

User defined datatypes let you extend the base SQL Server datatypes by providing a descriptive name, and format to the database. Take for example, in your database, there is a column called Flight_Num which appears in many tables. In all these tables it should be varchar(8). In this case you could create a user defined datatype called Flight_num_type of varchar(8) and use it across all your tables. See sp_addtype, sp_droptype in books online.

What is bit datatype and what's the information that can be stored inside a bit column?

Bit datatype is used to store boolean information like 1 or 0 (true or false). Until SQL Server 6.5 bit datatype could hold either a 1 or 0 and there was no support for NULL. But from SQL Server 7.0 onwards, bit datatype can represent a third state, which is NULL.

Define candidate key, alternate key, composite key.

A candidate key is one that can identify each row of a table uniquely. Generally a candidate key becomes the primary key of the table. If the table has more than one candidate key, one of them will become the primary key, and the rest are called alternate keys. A key formed by combining at least two or more columns is called composite key.

What are defaults? Is there a column to which a default can't be bound?

A default is a value that will be used by a column, if no value is supplied to that column while inserting data. IDENTITY columns and timestamp columns can't have defaults bound to them. See CREATE DEFAULT in books online.

What is a transaction and what are ACID properties?

A transaction is a logical unit of work in which, all the steps must be performed or none. ACID stands for Atomicity, Consistency, Isolation, Durability. These are the properties of a transaction. For more information and explanation of these properties, see SQL Server books online or any RDBMS fundamentals text book. Explain different isolation levels An isolation level determines the degree of isolation of data between concurrent transactions. The default SQL Server isolation level is Read Committed. Here are the other isolation levels (in the ascending order of isolation): Read Uncommitted, Read Committed, Repeatable Read, Serializable. See SQL Server books online for an explanation of the isolation levels. Be sure to read about SET TRANSACTION ISOLATION LEVEL, which lets you customize the isolation level at the connection level. Read Committed - A transaction operating at the Read Committed level cannot see changes made by other transactions until those transactions are committed. At this level of isolation, dirty reads are not possible but nonrepeatable reads and phantoms are possible. Read Uncommitted - A transaction operating at the Read Uncommitted level can see uncommitted changes made by other transactions. At this level of isolation, dirty reads, nonrepeatable reads, and phantoms are all possible. Repeatable Read - A transaction operating at the Repeatable Read level is guaranteed not to see any changes made by other transactions in values it has already read. At this level of isolation, dirty reads and nonrepeatable reads are not possible but phantoms are possible. Serializable - A transaction operating at the Serializable level guarantees that all concurrent transactions interact only in ways that produce the same effect as if each transaction were entirely executed one after the other. At this isolation level, dirty reads, nonrepeatable reads, and phantoms are not possible.

What's the maximum size of a row?

8060 bytes. Don't be surprised with questions like 'what is the maximum number of columns per table'. 1024 columns per table. Check out SQL Server books online for the page titled: "Maximum Capacity Specifications". Explain Active/Active and Active/Passive cluster configurations Hopefully you have experience setting up cluster servers. But if you don't, at least be familiar with the way clustering works and the two clustering configurations Active/Active and Active/Passive. SQL Server books online has enough information on this topic and there is a good white paper available on Microsoft site. Explain the architecture of SQL Server This is a very important question and you better be able to answer it if consider yourself a DBA. SQL Server books online is the best place to read about SQL Server architecture. Read up the chapter dedicated to SQL Server Architecture.

What is lock escalation?

Lock escalation is the process of converting a lot of low level locks (like row locks, page locks) into higher level locks (like table locks). Every lock is a memory structure too many locks would mean, more memory being occupied by locks. To prevent this from happening, SQL Server escalates the many fine-grain locks to fewer coarse-grain locks. Lock escalation threshold was definable in SQL Server 6.5, but from SQL Server 7.0 onwards it's dynamically managed by SQL Server.

What's the difference between DELETE TABLE and TRUNCATE TABLE commands?

DELETE TABLE is a logged operation, so the deletion of each row gets logged in the transaction log, which makes it slow. TRUNCATE TABLE also deletes all the rows in a table, but it won't log the deletion of each row, instead it logs the deallocation of the data pages of the table, which makes it faster. Of course, TRUNCATE TABLE can be rolled back. TRUNCATE TABLE is functionally identical to DELETE statement with no WHERE clause: both remove all rows in the table. But TRUNCATE TABLE is faster and uses fewer system and transaction log resources than DELETE. The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row. TRUNCATE TABLE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log. TRUNCATE TABLE removes all rows from a table, but the table structure and its columns, constraints, indexes and so on remain. The counter used by an identity for new rows is reset to the seed for the column. If you want to retain the identity counter, use DELETE instead. If you want to remove table definition and its data, use the DROP TABLE statement. You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint; instead, use DELETE statement without a WHERE clause. Because TRUNCATE TABLE is not logged, it cannot activate a trigger. TRUNCATE TABLE may not be used on tables participating in an indexed view

Explain the storage models of OLAP

Check out MOLAP, ROLAP and HOLAP in SQL Server books online for more information.

What are constraints? Explain different types of constraints.

Constraints enable the RDBMS enforce the integrity of the database automatically, without needing you to create triggers, rule or defaults. Types of constraints: NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY. For an explanation of these constraints see books online for the pages titled: "Constraints" and "CREATE TABLE", "ALTER TABLE"

What is an index? What are the types of indexes? How many clustered indexes can be created on a table? I create a separate index on each column of a table. What are the advantages and disadvantages of this approach?

Indexes in SQL Server are similar to the indexes in books. They help SQL Server retrieve the data quicker. Indexes are of two types. Clustered indexes and non-clustered indexes. When you create a clustered index on a table, all the rows in the table are stored in the order of the clustered index key. So, there can be only one clustered index per table. Non-clustered indexes have their own storage separate from the table data storage. Non-clustered indexes are stored as B-tree structures (so do clustered indexes), with the leaf level nodes having the index key and its row locator. The row located could be the RID or the Clustered index key, depending upon the absence or presence of clustered index on the table. If you create an index on each column of a table, it improves the query performance, as the query optimizer can choose from all the existing indexes to come up with an efficient execution plan. At the same time, data modification operations (such as INSERT, UPDATE, DELETE) will become slow, as every time data changes in the table, all the indexes need to be updated. Another disadvantage is that, indexes need disk space, the more indexes you have, more disk space is used.

What is RAID and what are different types of RAID configurations?

RAID stands for Redundant Array of Inexpensive Disks, used to provide fault tolerance to database servers. There are six RAID levels 0 through 5 offering different levels of performance, fault tolerance. MSDN has some information about RAID levels and for detailed information, check out the RAID advisory board's homepage

What are the steps you will take to improve performance of a poor performing query?

This is a very open ended question and there could be a lot of reasons behind the poor performance of a query. But some general issues that you could talk about would be: No indexes, table scans, missing or out of date statistics, blocking, excess recompilations of stored procedures, procedures and triggers without SET NOCOUNT ON, poorly written query with unnecessarily complicated joins, too much normalization, excess usage of cursors and temporary tables. Some of the tools/ways that help you troubleshooting performance problems are: SET SHOWPLAN_ALL ON, SET SHOWPLAN_TEXT ON, SET STATISTICS IO ON, SQL Server Profiler, Windows NT /2000 Performance monitor, Graphical execution plan in Query Analyzer. Download the white paper on performance tuning SQL Server from Microsoft web site. Don't forget to check out sql-server-performance.com

What are the steps you will take, if you are tasked with securing an SQL Server?

Again this is another open ended question. Here are some things you could talk about: Preferring NT authentication, using server, database and application roles to control access to the data, securing the physical database files using NTFS permissions, using an unguessable SA password, restricting physical access to the SQL Server, renaming the Administrator account on the SQL Server computer, disabling the Guest account, enabling auditing, using multiprotocol encryption, setting up SSL, setting up firewalls, isolating SQL Server from the web server etc. Read the white paper on SQL Server security from Microsoft website. Also check out MySQL Server security best practices

What is a deadlock and what is a live lock? How will you go about resolving deadlocks?

Deadlock is a situation when two processes, each having a lock on one piece of data, attempt to acquire a lock on the other's piece. Each process would wait indefinitely for the other to release the lock, unless one of the user processes is terminated. SQL Server detects deadlocks and terminates one user's process. A livelock is one, where a request for an exclusive lock is repeatedly denied because a series of overlapping shared locks keeps interfering. SQL Server detects the situation after four denials and refuses further shared locks. A livelock also occurs when read transactions monopolize a table or page, forcing a write transaction to wait indefinitely. Check out SET DEADLOCK_PRIORITY and "Minimizing Deadlocks" in SQL Server books online. Also check out the article Q169960 from Microsoft knowledge base.

What is blocking and how would you troubleshoot it?

Blocking happens when one connection from an application holds a lock and a second connection requires a conflicting lock type. This forces the second connection to wait, blocked on the first. Read up the following topics in SQL Server books online: Understanding and avoiding blocking, Coding efficient transactions. Explain CREATE DATABASE syntax Many of us are used to creating databases from the Enterprise Manager or by just issuing the command: CREATE DATABASE MyDB.

How to restart SQL Server in single user mode? How to start SQL Server in minimal configuration mode?

SQL Server can be started from command line, using the SQLSERVR.EXE. This EXE has some very important parameters with which a DBA should be familiar with. -m is used for starting SQL Server in single user mode and -f is used to start the SQL Server in minimal configuration mode. Check out SQL Server books online for more parameters and their explanations.

As a part of your job, what are the DBCC commands that you commonly use for database maintenance?

DBCC CHECKDB, DBCC CHECKTABLE, DBCC CHECKCATALOG, DBCC CHECKALLOC, DBCC SHOWCONTIG, DBCC SHRINKDATABASE, DBCC SHRINKFILE etc. But there are a whole load of DBCC commands which are very useful for DBAs. Check out SQL Server books online for more information.

What are statistics, under what circumstances they go out of date, how do you update them?

Statistics determine the selectivity of the indexes. If an indexed column has unique values then the selectivity of that index is more, as opposed to an index with non-unique values. Query optimizer uses these indexes in determining whether to choose an index or not while executing a query. Some situations under which you should update statistics: 1) If there is significant change in the key values in the index 2) If a large amount of data in an indexed column has been added, changed, or removed (that is, if the distribution of key values has changed), or the table has been truncated using the TRUNCATE TABLE statement and then repopulated 3) Database is upgraded from a previous version. Look up SQL Server books online for the following commands: UPDATE STATISTICS, STATS_DATE, DBCC SHOW_STATISTICS, CREATE STATISTICS, DROP STATISTICS, sp_autostats, sp_createstats, sp_updatestats

Explain different types of BACKUPs available in SQL Server? Given a particular scenario, how would you go about choosing a backup plan?

Types of backups you can create in SQL Server 7.0+ are Full database backup, differential database backup, transaction log backup, filegroup backup. Check out the BACKUP and RESTORE commands in SQL Server books online. Be prepared to write the commands in your interview. Books online also has information on detailed backup/restore architecture and when one should go for a particular kind of backup.

What is database replication? What are the different types of replication you can set up in SQL Server?

Replication is the process of copying/moving data between databases on the same or different servers. SQL Server supports the following types of replication scenarios:

- Snapshot
- Transactional replication (with immediate updating subscribers, with queued updating subscribers)
- Merge replication

 See SQL Server books online for indepth coverage on replication. Be prepared to explain how different replication agents function, what are the main system tables used in replication etc.

How to determine the service pack currently installed on SQL Server?

The global variable @@Version stores the build number of the sqlservr.exe, which is used to determine the service pack installed. To know more about this process visit SQL Server service packs and versions.

What is a join and explain different types of joins.

Joins are used in queries to explain how different tables are related. Joins also let you select data from a table depending upon data from another table. Types of joins: INNER JOINS, OUTER JOINS, CROSS JOINS. OUTER JOINS are further classified as LEFT OUTER JOINS, RIGHT OUTER JOINS and FULL OUTER JOINS. For more information see pages from books online titled: "Join Fundamentals" and "Using Joins".

Can you have a nested transaction?

Yes, very much. Check out BEGIN TRAN, COMMIT, ROLLBACK, SAVE TRAN and @@TRANCOUNT

What is an extended stored procedure? Can you instantiate a COM object by using T-SQL?

An extended stored procedure is a function within a DLL (written in a programming language like C, C++ using Open Data Services (ODS) API) that can be called from T-SQL, just the way we call normal stored procedures using the EXEC statement. See books online to learn how to create extended stored procedures and how to add them to SQL Server. Yes, you can instantiate a COM (written in languages like VB, VC++) object from T-SQL by using sp_OACreate stored procedure. Also see books online for sp_OAMethod, sp_OAGetProperty, sp_OASetProperty, sp_OADestroy. For an example of creating a COM object in VB and calling it from T-SQL, see 'My code library' section of this site.

What is the system function to get the current user's user id?

USER_ID(). Also check out other system functions like USER_NAME(), SYSTEM_USER, SESSION_USER, CURRENT_USER, USER, SUSER_SID(), HOST_NAME().

What are triggers? How many triggers you can have on a table? How to invoke a trigger on demand?

Triggers are special kind of stored procedures that get executed automatically when an INSERT, UPDATE or DELETE operation takes place on a table. In SQL Server 6.5 you could

define only 3 triggers per table, one for INSERT, one for UPDATE and one for DELETE. From SQL Server 7.0 onwards, this restriction is gone, and you could create multiple triggers per each action. But in 7.0 there's no way to control the order in which the triggers fire. In SQL Server 2000 you could specify which trigger fires first or fires last using sp_settriggerorder. Triggers can't be invoked on demand. They get triggered only when an associated action (INSERT, UPDATE, DELETE) happens on the table on which they are defined. Triggers are generally used to implement business rules, auditing. Triggers can also be used to extend the referential integrity checks, but wherever possible, use constraints for this purpose, instead of triggers, as constraints are much faster. Till SQL Server 7.0, triggers fire only after the data modification operation happens. So in a way, they are called post triggers. But in SQL Server 2000 you could create pre triggers also. Search SQL Server 2000 books online for INSTEAD OF triggers. Also check out books online for 'inserted table', 'deleted table' and COLUMN_UPDATED()

What is a self join? Explain it with an example.

Self join is just like any other join, except that two instances of the same table will be joined in the query. Here is an example: Employees table which contains rows for normal employees as well as managers. So, to find out the managers of all the employees, you need a self join.

```
CREATE TABLE emp ( empid int, mgrid int, empname char(10) )  
INSERT emp SELECT 1,2,'Vyas' INSERT emp SELECT 2,3,'Mohan' INSERT emp SELECT  
3,NULL,'Shobha' INSERT emp SELECT 4,2,'Shridhar' INSERT emp SELECT 5,2,'Sourabh'  
SELECT t1.empname [Employee], t2.empname [Manager] FROM emp t1, emp t2 WHERE  
t1.mgrid = t2.empid Here's an advanced query using a LEFT OUTER JOIN that even returns the  
employees without managers (super bosses)  
SELECT t1.empname [Employee], COALESCE(t2.empname, 'No manager') [Manager] FROM  
emp t1 LEFT OUTER JOIN emp t2 ON t1.mgrid = t2.empid
```

How to read transaction logs?

There is no easy way out if you want to read the transaction logs for whatever reason. The transaction log architecture is proprietary to Microsoft and is not published.

However, there is an undocumented DBCC LOG command that lets you see the records in transaction log. Here is an example:

DBCC LOG (Your_Database_Name, 2)

There is a third party tool called Log Explorer by [Lumigent](#), that helps you read transaction logs and do stuff like recovering data, auditing database etc.

How to reset or reseed the IDENTITY column?

See DBCC CHECKIDENT in SQL Server Books Online.

A quick and dirty way to reset the IDENTITY column would be to run TRUNCATE TABLE command on that table. TRUNCATE TABLE will delete all the rows from the table and reset the IDENTITY column. However, you will not be able to run TRUNCATE TABLE on a table referenced by foreign keys.

How to persist objects, permissions etc. in tempdb?

Tempdb gets recreated every time SQL Server service restarts. So, you will end up losing whatever you store in tempdb. Actually, it's not a good practice to store your own objects in tempdb. But if you must have some of your tables or stored procedures or other objects in tempdb, consider the following two options:

- Create a stored procedure that creates the required objects in tempdb. Mark this stored procedure as a startup stored procedure, so that it runs everytime SQL Server service starts. See sp_procoption in SQL Server Books Online.
- Add the required objects to the model database. Since the model database is used as a template for creating new databases, all new databases will inherit the objects from model database.

How to simulate a deadlock for testing purposes?

In Query Analyzer, run the following statements first:

```
CREATE TABLE t1 (i int)
CREATE TABLE t2 (i int)
```

```
INSERT t1 SELECT 1
INSERT t2 SELECT 9
```

Open a new window (say Window1) in Query Analyzer, paste the following SQL statements:

```
BEGIN TRAN
UPDATE t1 SET i = 11 WHERE i = 1
WAITFOR DELAY '00:00:20'
UPDATE t2 SET i = 99 WHERE i = 9
COMMIT
```

Open another window (say Window2) in Query Analyzer and paste the following code:

```
BEGIN TRAN
UPDATE t2 SET i = 99 WHERE i = 9
WAITFOR DELAY '00:00:20'
UPDATE t1 SET i = 11 WHERE i = 1
```

```
COMMIT
```

Now run the code from Window1, followed by Window2 simultaneously. Briefly after 20 seconds, one of the windows will experience a dead lock!

How to rename an SQL Server computer?

If you are running SQL Server 7.0, after renaming the SQL Server machine, the SQL Server service will fail to start, with an error message "Your installation was corrupted or had been tampered with. To get around this problem, you have to rerun the SQL Server setup. Setup will prompt you to upgrade. After doing so, the necessary SQL Server registry entries will be reset with the new computer name. Now you will be able to start SQL Server. After restarting, use Query Analyzer to run the following commands:

```
EXEC sp_dropserver 'Your_OLD_Computer_Name'  
EXEC sp_addserver 'Your_NEW_Computer_Name', 'local'
```

Restart your SQL Server service. Connect using Query Analyzer and run the following command (It should output the new server name):

```
SELECT @@SERVERNAME
```

If you are running SQL Server 2000, the new name is recognized, the next time SQL Server service starts. You don't have to rerun the setup. However, you have to run the sp_dropserver and sp_addserver stored procedure as shown above.

How to restore single tables from backup in SQL Server

Support for restoring individual tables from backup is discontinued in SQL Server. If you need this functionality, here are some roundabout ways:

- Restore the complete database onto a new database with a different name. Copy the required tables (using T-SQL or DTS) into the actual database and drop the new database that you just created
- You could place the required tables onto specific filegroups and implement filegroup backup and restore. But filegroup backup will not backup the transaction log. So there is a chance of losing some data when you restore the filegroups. See SQL Server Books Online for more information

I have only the .mdf file backup and no SQL Server database backups. Can I get my database back into SQL Server?

Yes. The system stored procedures `sp_attach_db` and `sp_attach_single_file_db` allow you to attach .mdf files to SQL Server. In the absence of the log file (.ldf), SQL Server creates a new log file.

How to add a new column at a specific position (say at the beginning of the table or after the second column) using ALTER TABLE command?

`ALTER TABLE` always adds new columns at the end of the table and will not let you add new columns at a specific position. If you must add a column at a specific position, use Enterprise Manager. In Enterprise Manager, right click on the table, select 'Design Table'. Right click on the desired location and select 'Insert Column'. Mind you, Enterprise Manager drops and recreates the table to add a column at a specific location. So it might take a long time if your table is huge.

How to rename an SQL Server instance?

You cannot rename an instance of an SQL Server. If you must rename an instance, follow these steps:

- Install a new SQL Server instance with the desired name.
- Move your databases from the old instance to the newly created instance.
- Uninstall the old instance of SQL Server.

How to capture/redirect detailed deadlock information into the error logs?

To capture detailed deadlock information into the error logs, enable the trace flags 1204 and 3605 at the session level using the `DBCC TRACEON` command. When you enable these trace flags at the session level, only those deadlocks are captured into the error log, in which this session has participated.

To enable these trace flags at the server level, start your SQL Server from command prompt (`sqlservr.exe`) with `-T1204` and `-T3605` parameters. You could also set these trace flags from Enterprise Manager. (Right click on the server, select 'Properties'. Click on 'Startup parameters...'. Add the parameters `-T1204` and `-T3605` one after another by clicking the 'Add' button.). After setting these trace flags in Enterprise Manager, you must restart your SQL Server service for these trace flags to take effect.

Describe the difference between DATETIME and SMALLDATETIME and describe when you would use each

My answer would first start out with an overview of what the basic differences are - I would keep it high level and then, if they want more detail, I would explore deeper. The minor differences are the DATETIME is 8 bytes and SMALLDATETIME is 4 bytes as well as the fact that the date ranges in DATETIME are larger (1753 - 9999) than SMALLDATETIME (1900-2079).

What happens if you add a new index to large table?

An index can be added when you create a new table. New rows will be indexed as they are inserted into the table. But you can also add a new index to an existing table with the same CREATE INDEX statement. The existing rows will be indexed as part of the CREATE INDEX statement.

If you add a new index to an existing table with a large number of rows. The CREATE INDEX statement could take some time to finish.

How to rebuild all indexes on a single table?

If you have several indexes on a single table and want to rebuild all of them, you may use the "ALTER INDEX ALL ON table_name REBUILD" statement

What is the impact on User sessions when creating indexes?

If you are creating a new index on a table with existing data, all existing rows will be indexed as part of the CREATE INDEX statement. If the table is large, the indexing process could take some time. The impact of this indexing process on other user sessions is based whether SQL server is using the Offline mode or Online mode.

By default, SQL Server performs indexing operations in Offline mode, where table locks are applied for the duration of the index operation. An offline index operation that creates, rebuilds, or drops a clustered index, or rebuilds or drops a nonclustered index, acquires a Schema modification (Sch-M) lock on the table. This prevents all user access to the underlying table for the duration of the operation. An offline index operation that creates a nonclustered index acquires a Shared (S) lock on the table. This prevents updates to the underlying table but allows read operations, such as SELECT statements.

SQL Server Enterprise Edition supports indexing operations in Online mode, where other user sessions will not be impacted.

How to change the name of a database user?

If you want to change the name of an existing database user, you can use the "ALTER USER" statement

```
ALTER USER <username> WITH NAME= <new name>
```

What are page splits?

Pages are contained in extent. Every extent will have around eight data pages. But all the eight data pages are not created at once; they are created depending on data demand. So when a page becomes full it creates a new page, this process is called as Page Split.

// -- 0 -- //

