Technical Report

Microsoft SQL Server and NetApp SnapManager for SQL Server on NetApp Storage Best Practices Guide

Abhishek Basu, NetApp

June 2013 | TR-4003

Abstract

This best practice guide is designed to give storage administrators and database administrators the keys to successfully deploy Microsoft® SQL Server® on NetApp® storage.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Executive Summary

Today's business applications are more data-centric than in the past, requiring fast and reliable access to intelligent information structures that are often provided by a high-performance relational database system. Over the last decade, many organizations have selected Microsoft SQL Server to provide just such a back-end datastore for mission-critical business applications. The latest release, Microsoft SQL Server 2012 builds on the strong base of SQL Server 2008 delivering performance, scalability, availability, and security.

SQL Server implementations have become more complex and require more reliability than before. Database service-level agreements (SLAs) require predictable performance, and outages are disruptive and costly for database consumers. The underlying physical storage architectures supporting SQL Server are expected to scale in order to meet the growing capacity, but frequently bottlenecks arise and backup processes get slower as databases grow. SQL Server databases are growing significantly larger to meet the needs of today's organizations, while business requirements dictate shorter backup and restore windows.

Using SQL Server’s built-in streaming backup and recovery solution, restore times require the current backup time plus additional time to play any transaction logs. The larger the database, the harder it is to meet organizational service-level agreements. Failure to meet the organization’s SLA can have a direct and devastating effect on revenue and customer goodwill. It is clear that conventional backup mechanisms simply do not scale, and many organizations have sought new and innovative solutions to solve increasingly stringent business requirements.

1.1 Purpose and Scope

The success or failure of any software or infrastructure and software deployment hinges on making the right choices and configurations up front. The core goal of this report is to provide best practices for deploying and using SnapManager® for SQL Server (SMSQL) with Microsoft SQL Server 2005, 2008, 2008R2, and 2012 with NetApp storage systems and supporting software. Organizations looking to get the most out of their NetApp storage investment with SQL Server will benefit from putting into practice the recommendations included in this report.

At a high level, the following areas are covered:

- New features of SMSQL and complementary NetApp technologies
- Best practices for both database and storage logical and physical layout with SQL Server and NetApp storage systems and software
- Backup and recovery (BR) and disaster recovery (DR) considerations and tactics
- Best practices for installation, configuration, and ongoing use of SnapDrive® software and SMSQL
- Ongoing management and monitoring best practices

Note: This document highlights specific best practices in boxed sections throughout. All best practices are also available in the SMSQL Best Practices Recap section.

1.2 Intended Audience

This report is intended for experienced SQL Server administrators, IT managers, and storage administrators who have reviewed these documents:

- NetApp SnapDrive for Windows Installation and Administration Guide
- SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide
- Data ONTAP System Administration Guide

Readers should ideally have a solid understanding of SQL Server storage architecture and administration as well as SQL Server backup and restore concepts. For more information about Microsoft SQL Server architecture, refer to SQL Server Books Online (BOL).

1.3 Caveats

- Applications using SQL Server as a back end might have specific requirements dictated by the design characteristics of the application and are beyond the scope of this technical report. For application-specific guidelines and recommendations relating to physical and logical database layout, contact your application provider.
- Best practices for managing SQL Server environments focus exclusively on the latest NetApp storage operating system, the Data ONTAP® architecture.

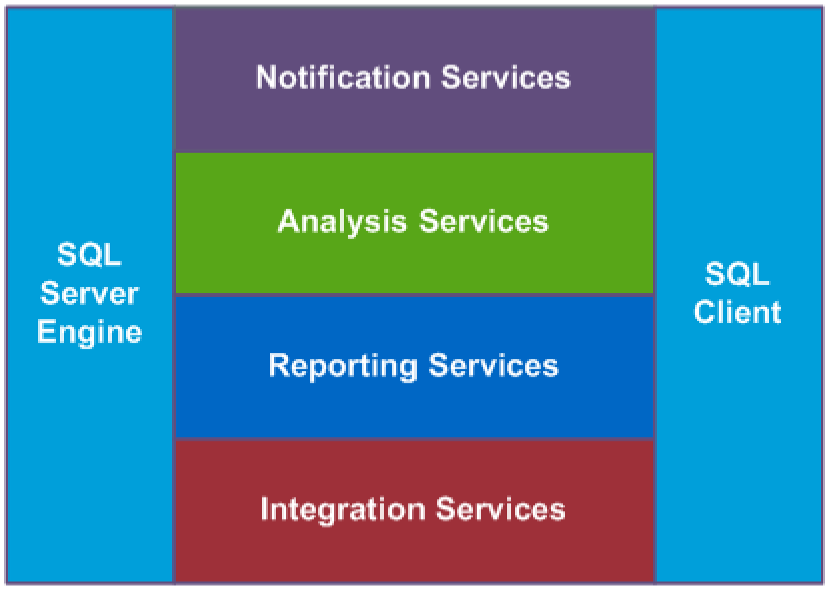
2 SQL Server Architecture and Components

SQL Server is a rich relational database engine that allows flexibility for both the small department and the large enterprise data warehouse. The flexibility of the architecture comes from the layering of components and services within the SQL Server product offering.

2.1 SQL Server Component Model

The SQL Server component model (Figure 1) helps you understand where the moving parts are concentrated when planning for SQL Server. The basic installation of SQL Server resides entirely on the local drive. The component locations can be moved around as needed.

Figure 1) SQL Server model.



The foundation of SQL Server is the database engine. The core engine consists of the initial 150MB needed from the operating system to start as well as the initial portion of memory. This guide does not recommend any particular type of drive, but describes factors you must consider when deploying SQL Server. This is important in helping customers understand that the storage designs for their SQL Server deployments are based on factors beyond the size of the SQL Server environment.

The ability to configure the location of the specific files allows flexibility. When sizing the SQL Server components, the needs of the SQL Server engine, Analysis Services, Reporting Services, and, in some cases, Integration Services must be addressed.

2.2 SQL Server 2012 New Features and Enhancements

This section explores some of the many new features and enhancements in SQL Server 2012.

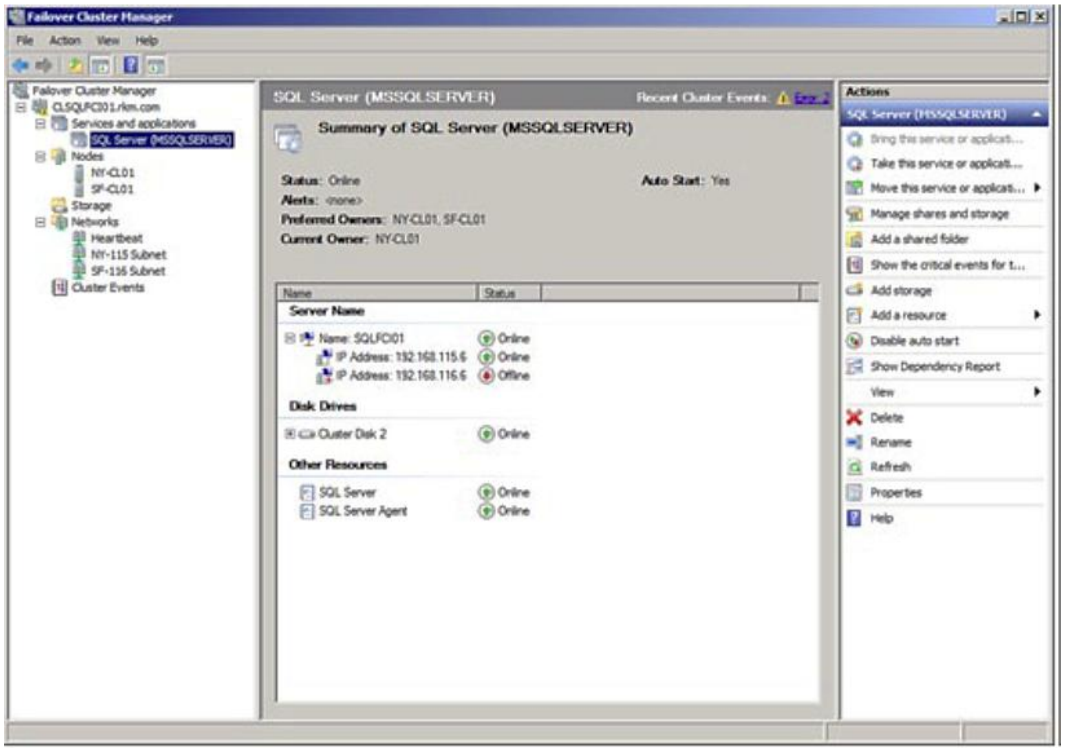
AlwaysOn Availability Groups

The new AlwaysOn Availability Groups capability protects databases and allows multiple databases to fail over as a single unit. Data redundancy and protection are improved because the solution supports up to four secondary replicas. Of these four, up to two can be configured as synchronous secondaries to make sure that the copies are up to date. The secondary replicas can reside within a data center to achieve high availability within a site, or on different data centers for disaster recovery purposes.

AlwaysOn Failover Cluster Instances

AlwaysOn Failover Cluster Instances (FCI) offers superior instance-level protection by using Windows® Server Failover Clustering and shared storage to deliver the following improvements:

- FCI now supports multisubnet failover clusters. These subnets, where the FCI nodes reside, can be located in the same data center or in geographically dispersed sites.
- Local storage can be leveraged for the TempDB database.
- Faster startup and recovery times are achieved after a failover occurs.
- Improved cluster health-detection policies can be leveraged, offering stronger and more flexible failover.



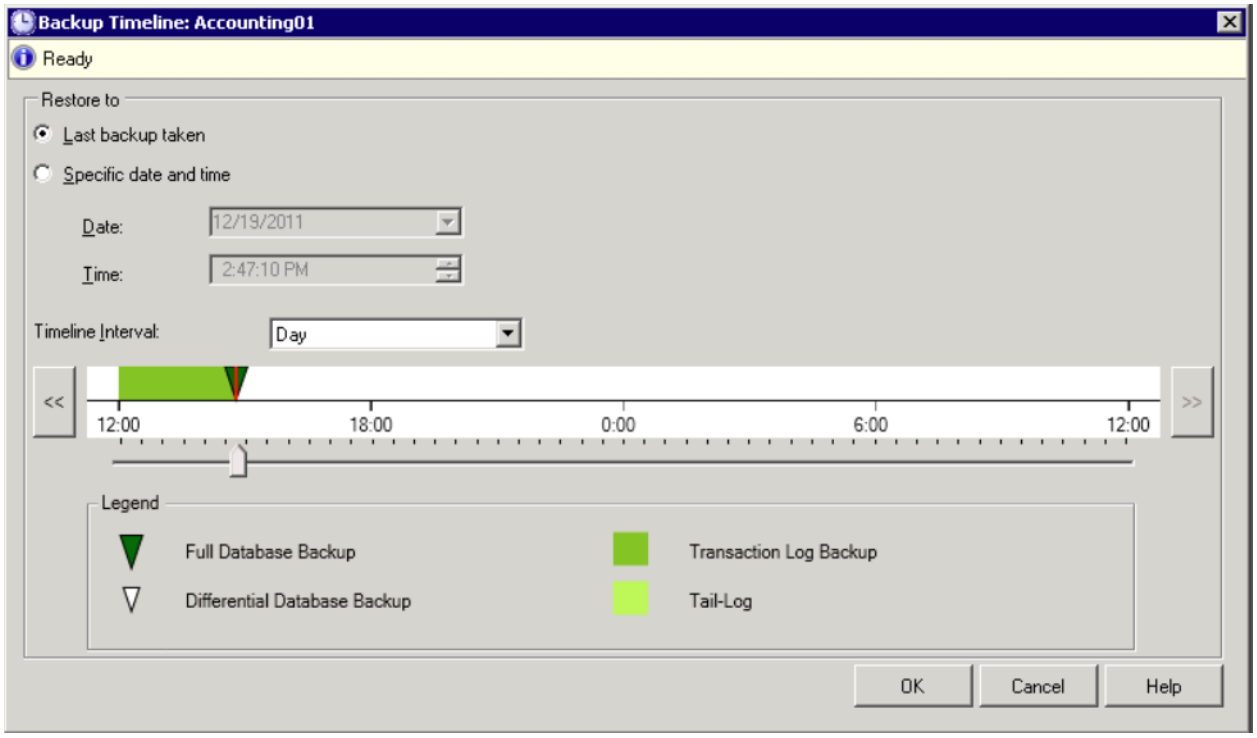
Summary of SQL Server (MSSQLSERVER)

Status: Online
Auto Start: Yes
Alerts: <none>
Preferred Owners: NY-CL01, SF-CL01
Current Owner: NY-CL01

Name	Status
Server Name	
Name: SQLFCI01	Online
IP Address: 192.168.115.6	Online
IP Address: 192.168.116.6	Offline
Disk Drives	
Cluster Disk 2	Online
Other Resources	
SQL Server	Online
SQL Server Agent	Online

Recovery Advisor

A new visual timeline has been introduced in SQL Server Management Studio to simplify the database restore process.


Backup Timeline: Accounting01

Ready

Restore to

(●) Last backup taken

( ) Specific date and time

Date: 12/19/2011

Time: 2:47:10 PM

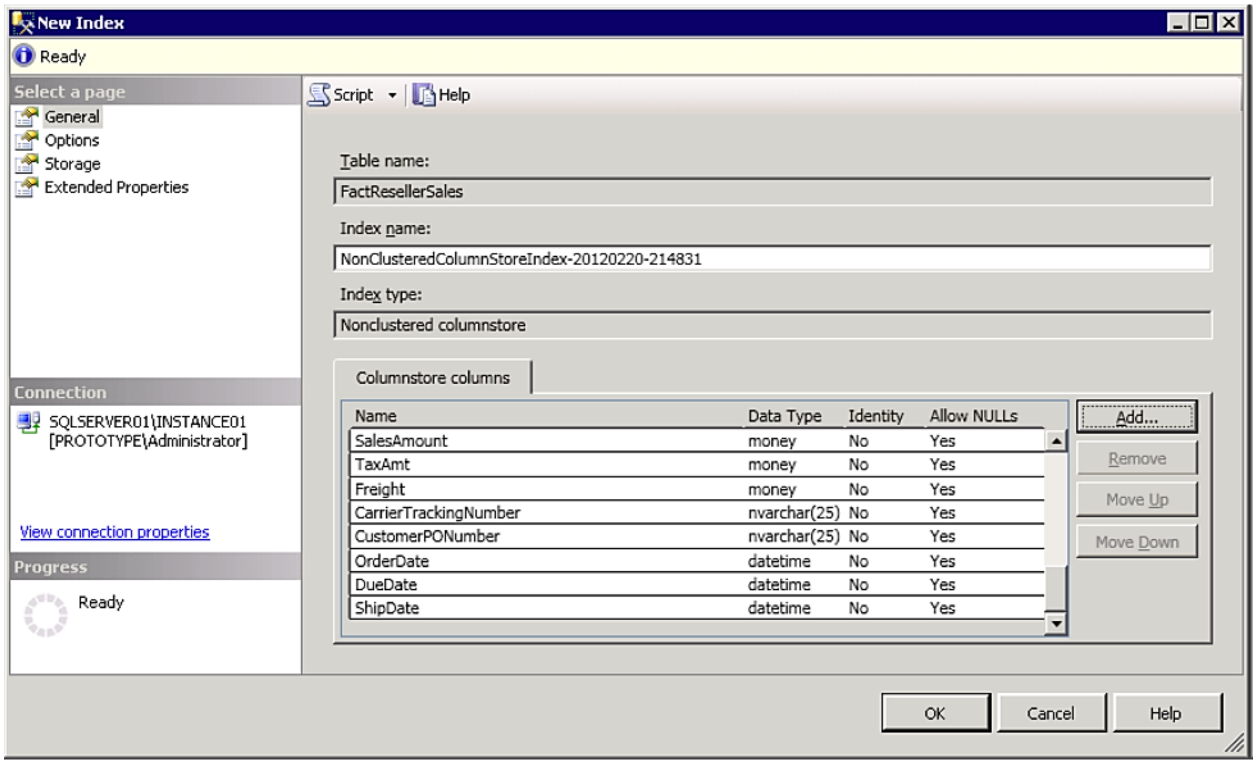
Timeline Interval: Day

Legend

▼	Full Database Backup	🟩	Transaction Log Backup
▽	Differential Database Backup	🟩	Tail-Log

Columnstore Indexes

SQL Server 2012 introduces a new in-memory, columnstore index built into the relational engine. Together with advanced query-processing enhancements, these technologies provide blazing fast performance and improve queries associated with data warehouse workloads.


New Index

Ready

Table name: FactResellerSales

Index name: NonClusteredColumnStoreIndex-20120220-214831

Index type: Nonclustered columnstore

Columnstore columns

Name	Data Type	Identity	Allow NULLs
SalesAmount	money	No	Yes
TaxAmt	money	No	Yes
Freight	money	No	Yes
CarrierTrackingNumber	nvarchar(25)	No	Yes
CustomerPONumber	nvarchar(25)	No	Yes
OrderDate	datetime	No	Yes
DueDate	datetime	No	Yes
ShipDate	datetime	No	Yes

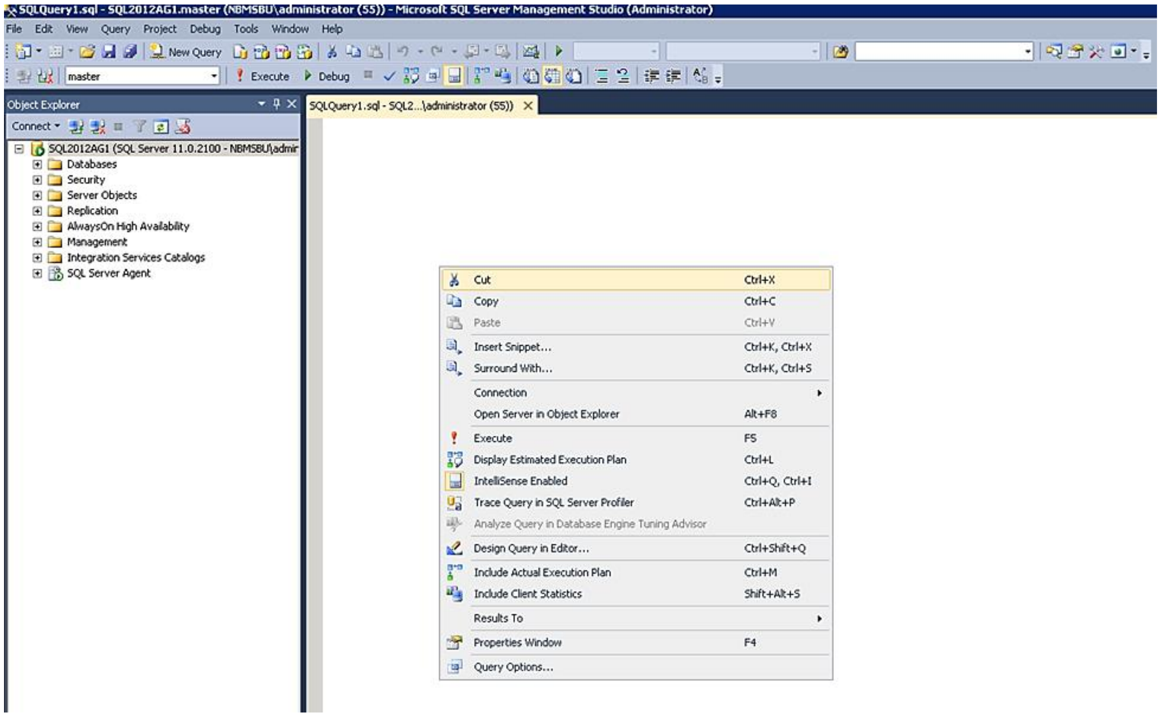
Connection

SQLSERVER01\INSTANCE01
[PROTOTYPE\Administrator]

Ready

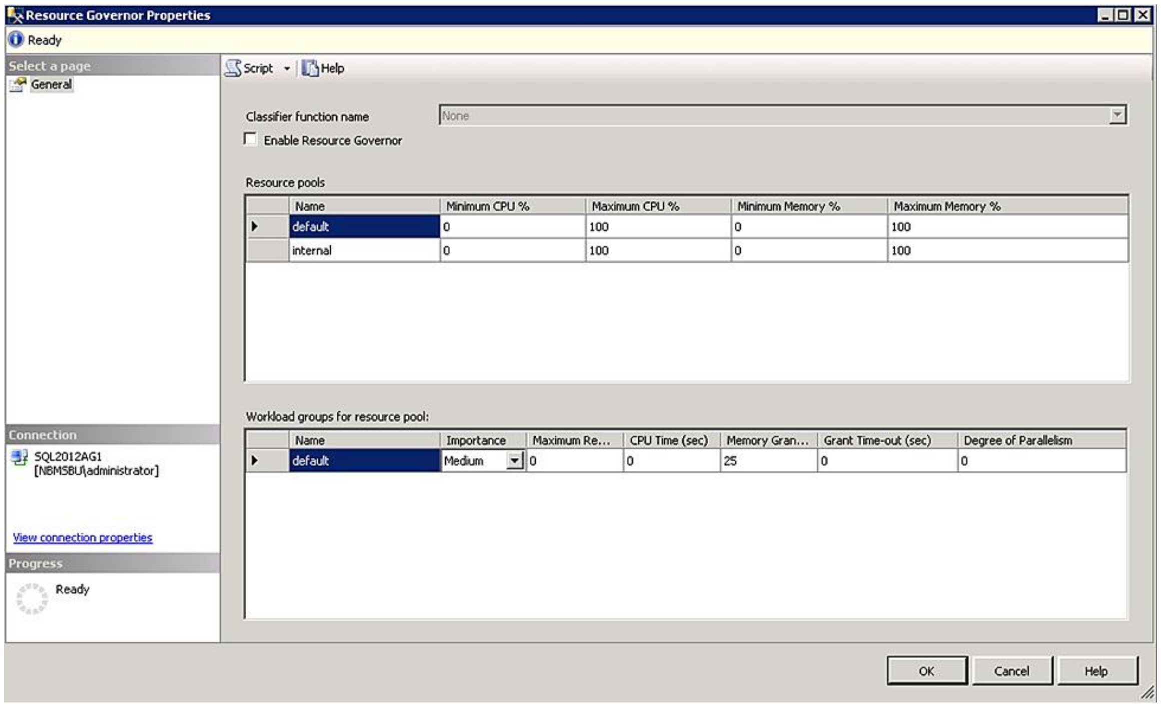
SQL Server Management Studio

IntelliSense, TSQL debugging, and the Insert Snippet menu have been enhanced and are included in the latest version of SQL Server Management Studio.


SQLQuery1.sql - SQL2012AG1.master (NBMSBU\administrator (55)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Debug Tools Window Help
New Query
master
Execute
Debug
Object Explorer
Connect
SQLQuery1.sql - SQL2...\administrator (55))
SQL2012AG1 (SQL Server 11.0.2100 - NBMSBU\admir
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Connection
Open Server in Object Explorer Alt+F8
Execute F5
Display Estimated Execution Plan Ctrl+L
IntelliSense Enabled Ctrl+Q, Ctrl+I
Trace Query in SQL Server Profiler Ctrl+Alt+P
Analyze Query in Database Engine Tuning Advisor
Design Query in Editor... Ctrl+Shift+Q
Include Actual Execution Plan Ctrl+M
Include Client Statistics Shift+Alt+S
Results To
Properties Window F4
Query Options...

Resource Governor Enhancements

Support for larger-scale multi-tenancy can now be achieved on a single instance of SQL Server, because the number of resource pools that Resource Governor supports has 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.



Classifier function name: None

Enable Resource Governor [ ]

SQL2012AG1
[NBMSBU\administrator]

Resource pools

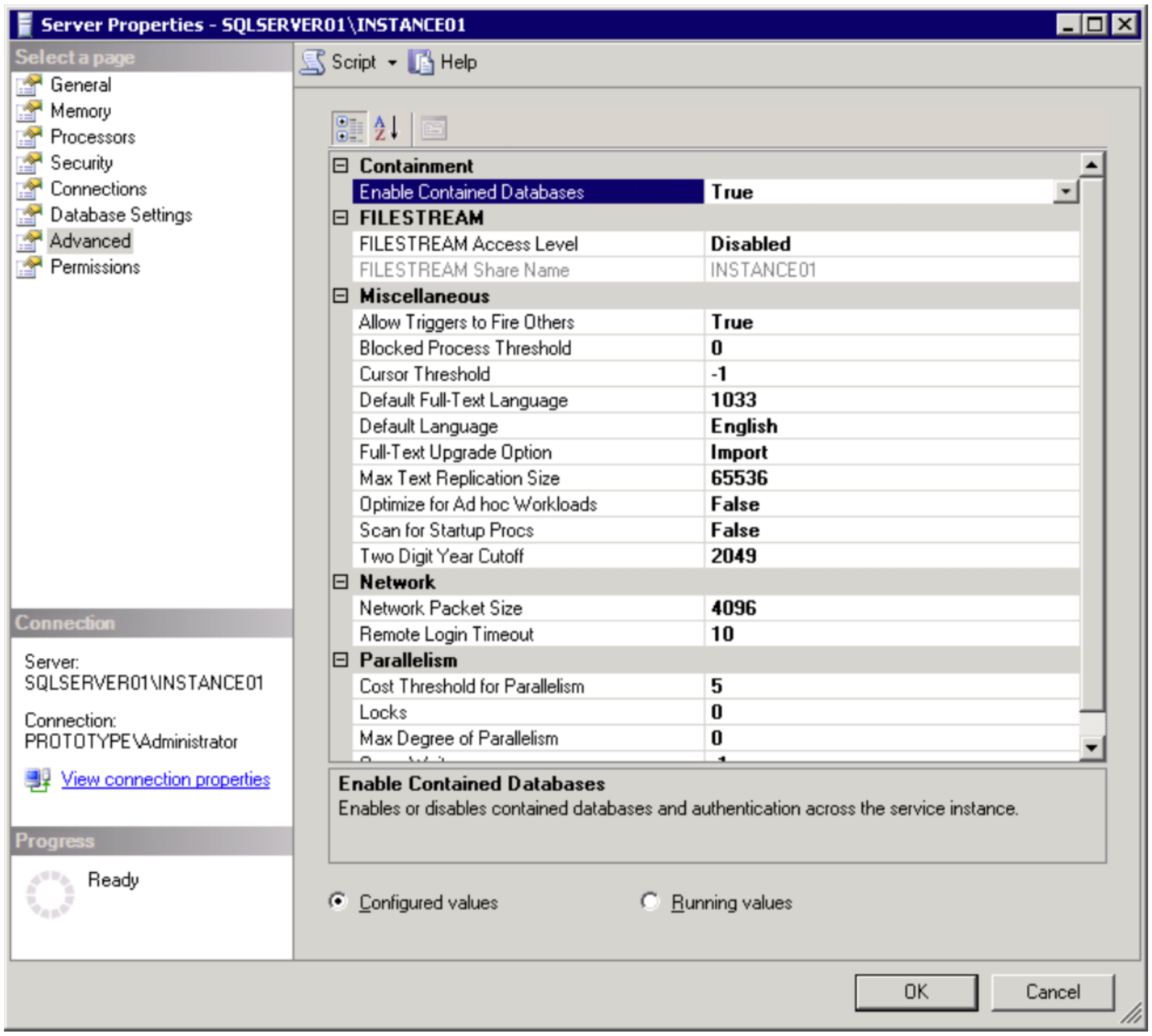
Name	Minimum CPU %	Maximum CPU %	Minimum Memory %	Maximum Memory %
default	0	100	0	100
internal	0	100	0	100

Workload groups for resource pool:

Name	Importance	Maximum Re...	CPU Time (sec)	Memory Gran...	Grant Time-out (sec)	Degree of Parallelism
default	Medium	0	0	25	0	0

Contained Databases

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 portability of user databases among servers because contained databases have no external dependencies.



Server:
SQLSERVER01\INSTANCE01

Connection:
PROTOTYPE\Administrator

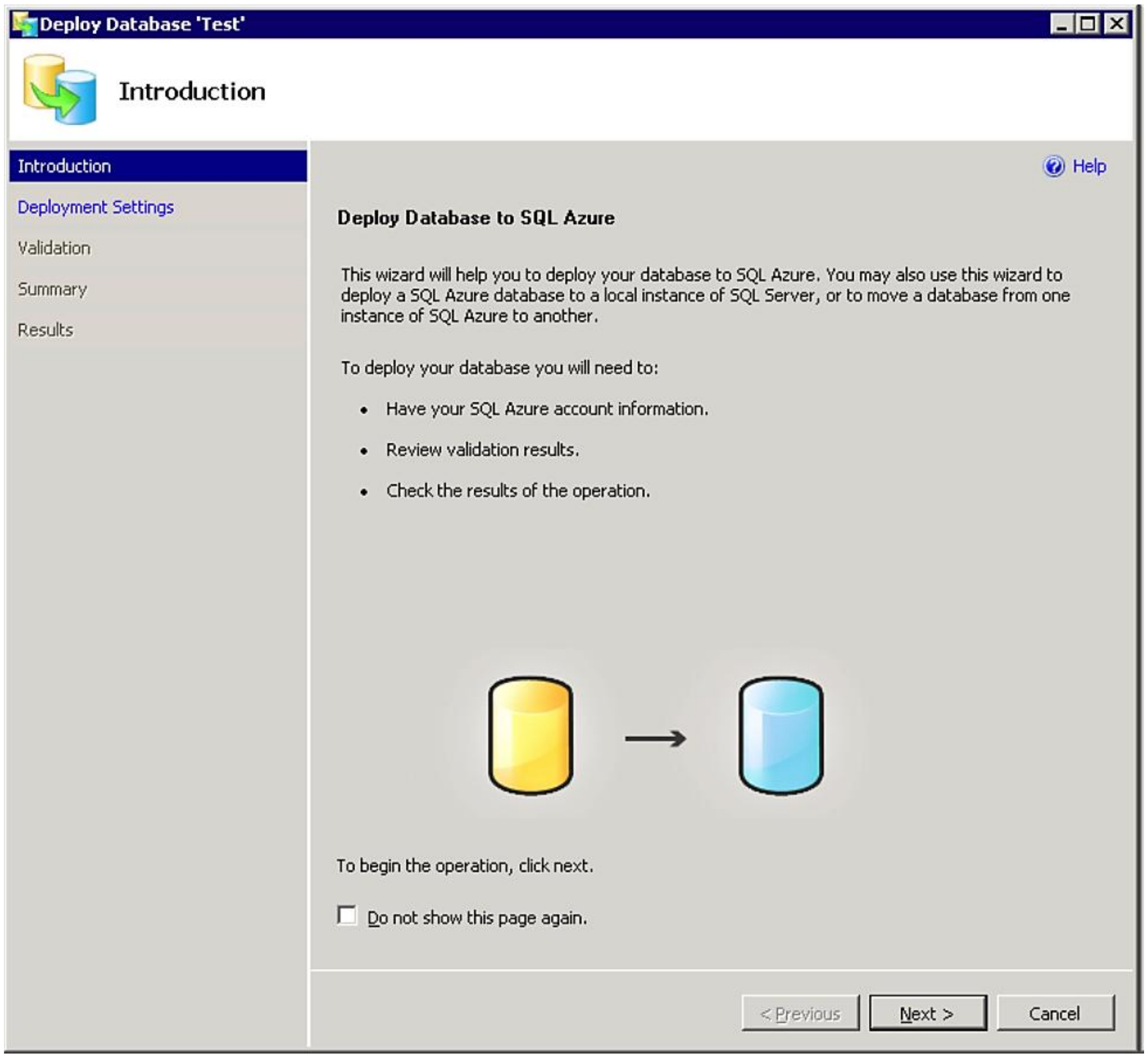
Containment	
Enable Contained Databases	True
FILESTREAM	
FILESTREAM Access Level	Disabled
FILESTREAM Share Name	INSTANCE01
Miscellaneous	
Allow Triggers to Fire Others	True
Blocked Process Threshold	0
Cursor Threshold	-1
Default Full-Text Language	1033
Default Language	English
Full-Text Upgrade Option	Import
Max Text Replication Size	65536
Optimize for Ad hoc Workloads	False
Scan for Startup Procs	False
Two Digit Year Cutoff	2049
Network	
Network Packet Size	4096
Remote Login Timeout	10
Parallelism	
Cost Threshold for Parallelism	5
Locks	0
Max Degree of Parallelism	0

Enable Contained Databases
Enables or disables contained databases and authentication across the service instance.

[✓ Configured values] Running values

Integration with SQL Azure

A new Deploy Database to SQL Azure wizard is integrated into the SQL Server database engine to help organizations deploy an on-premise database to SQL Azure™.


Deploy Database 'Test'

Introduction

Deploy Database to SQL Azure

This wizard will help you to deploy your database to SQL Azure. You may also use this wizard to deploy a SQL Azure database to a local instance of SQL Server, or to move a database from one instance of SQL Azure to another.

To deploy your database you will need to:

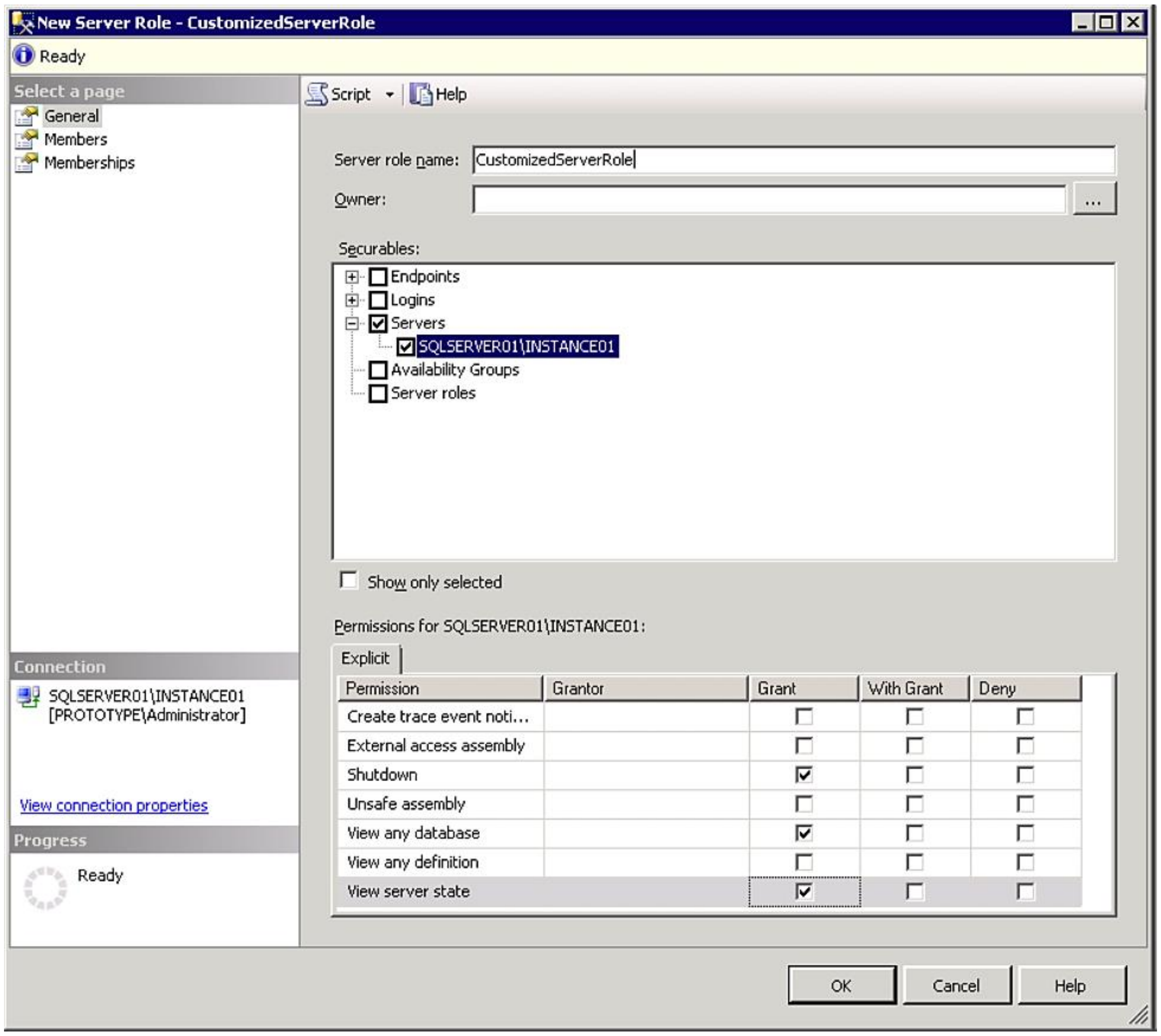
- Have your SQL Azure account information.
- Review validation results.
- Check the results of the operation.

To begin the operation, click next.

☐ Do not show this page again.

User-Defined Server Roles

Role-based security and separation of duties are strategies that organizations must adhere to in order for their systems to comply with internal and external security policies. The main goal of these strategies is to limit system access to authorized users to reduce security threats, compromised security, and operational mistakes, while improving the manageability of users and their privileges. With SQL Server 2012, user-defined roles have been introduced at the server level to increase flexibility and manageability and to facilitate compliance along with improved separation of duties, when administering the server.



Server role name: CustomizedServerRole

Owner:

Securables:
- Endpoints
- Logins
- [✓] Servers
  - [✓] SQLSERVER01\INSTANCE01
- Availability Groups
- Server roles

Show only selected

Connection: SQLSERVER01\INSTANCE01 [PROTOTYPE\Administrator]

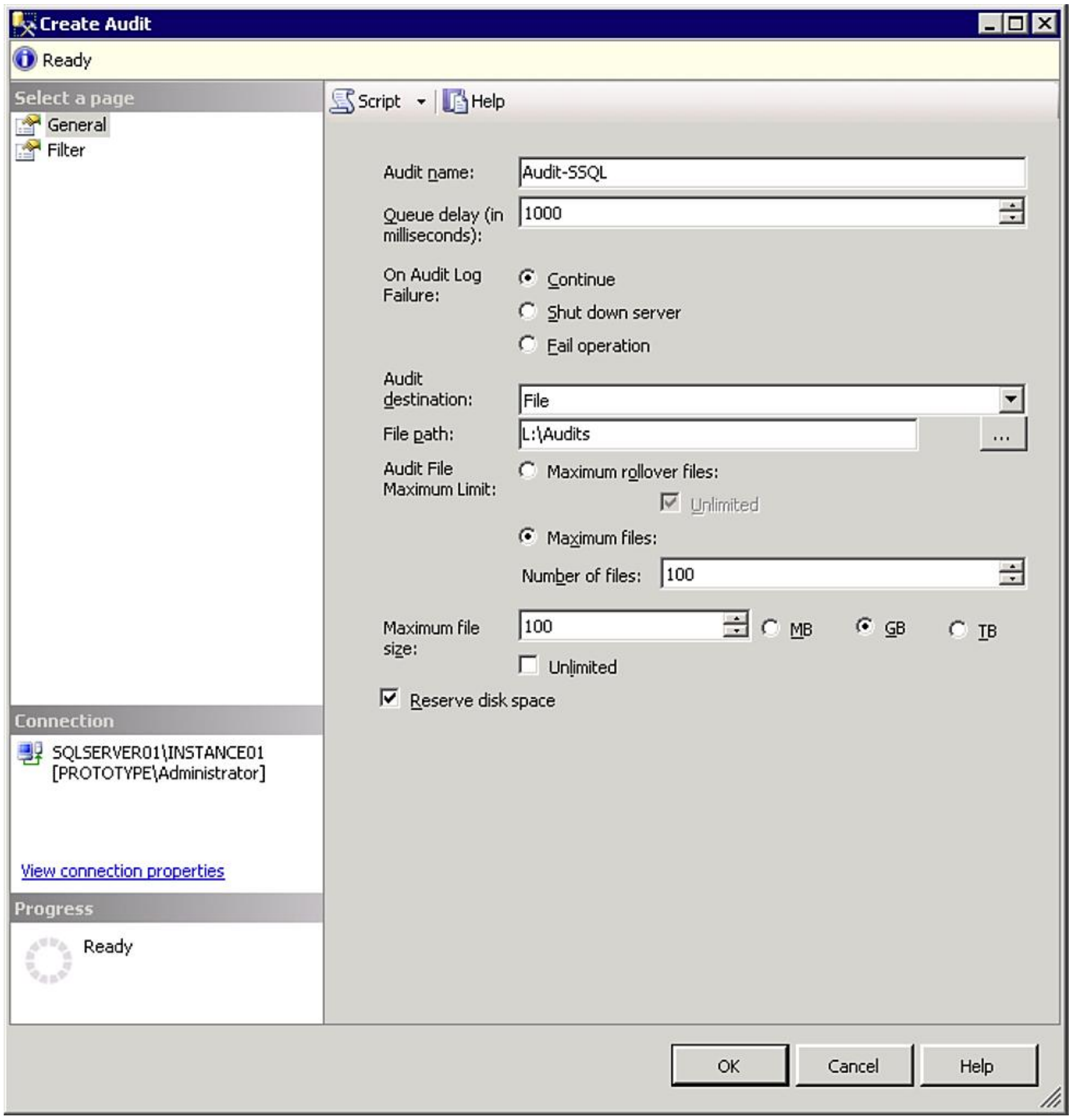
Permissions for SQLSERVER01\INSTANCE01:

Explicit

Permission	Grantor	Grant	With Grant	Deny
Create trace event noti...		☐	☐	☐
External access assembly		☐	☐	☐
Shutdown		✓	☐	☐
Unsafe assembly		☐	☐	☐
View any database		✓	☐	☐
View any definition		☐	☐	☐
View server state		✓	☐	☐

Audit Enhancements

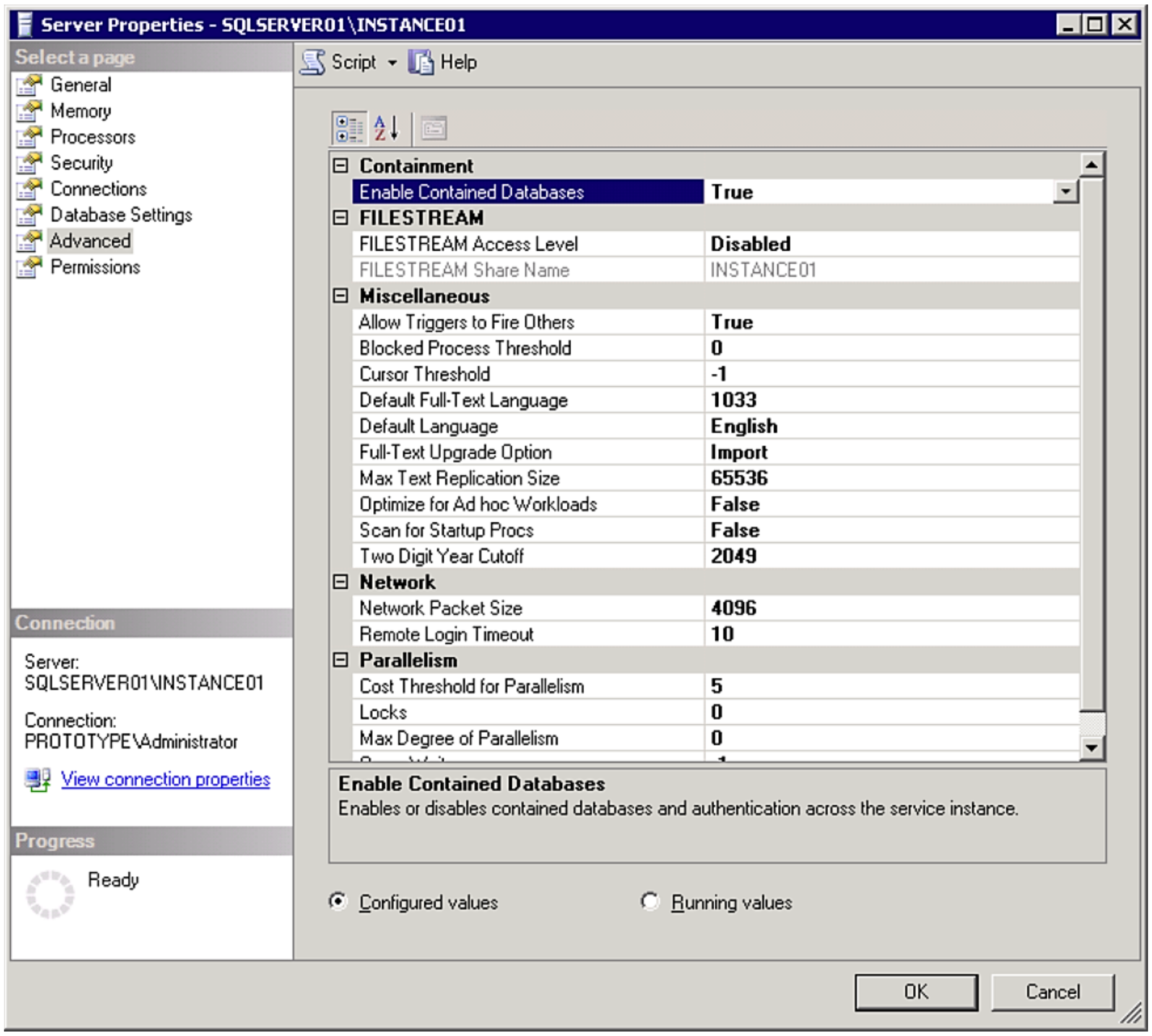
Basic audit functionality is available on all SQL Server 2012 SKUs. There are also enhancements like On Audit Shut Down Server, On Audit Log Failure: Continue, and On Audit Log Failure: Fail Operation. Also, a new user-defined audit event allows users to write custom events into the audit log.



Audit name: Audit-SSQL
Queue delay (in milliseconds): 1000
On Audit Log Failure: [✓ Continue] Shut down server Fail operation
Audit destination: File
File path: L:\Audits
Audit File Maximum Limit: Maximum rollover files: [✓ Unlimited] [✓ Maximum files:]
Number of files: 100
Maximum file size: 100 MB [✓ GB] TB
Unlimited
[✓ Reserve disk space]
SQLSERVER01\INSTANCE01 [PROTOTYPE\Administrator]

Database Authentication Enhancements

SQL Server 2012 addresses authentication and login dependency challenges by introducing Contained Database Authentication to enhance compliance, authorization, and portability of user databases. It allows users to be authenticated directly into a user database without logins that reside in the database engine. SQL Server 2012 also allows authentication without logins for SQL Server users with passwords and Windows authentication without login. This feature is especially important to leverage when implementing AlwaysOn Availability Groups.



Server: SQLSERVER01\INSTANCE01

Connection: PROTOTYPE\Administrator

Containment	
Enable Contained Databases	True
FILESTREAM	
FILESTREAM Access Level	Disabled
FILESTREAM Share Name	INSTANCE01
Miscellaneous	
Allow Triggers to Fire Others	True
Blocked Process Threshold	0
Cursor Threshold	-1
Default Full-Text Language	1033
Default Language	English
Full-Text Upgrade Option	Import
Max Text Replication Size	65536
Optimize for Ad hoc Workloads	False
Scan for Startup Procs	False
Two Digit Year Cutoff	2049
Network	
Network Packet Size	4096
Remote Login Timeout	10
Parallelism	
Cost Threshold for Parallelism	5
Locks	0
Max Degree of Parallelism	0

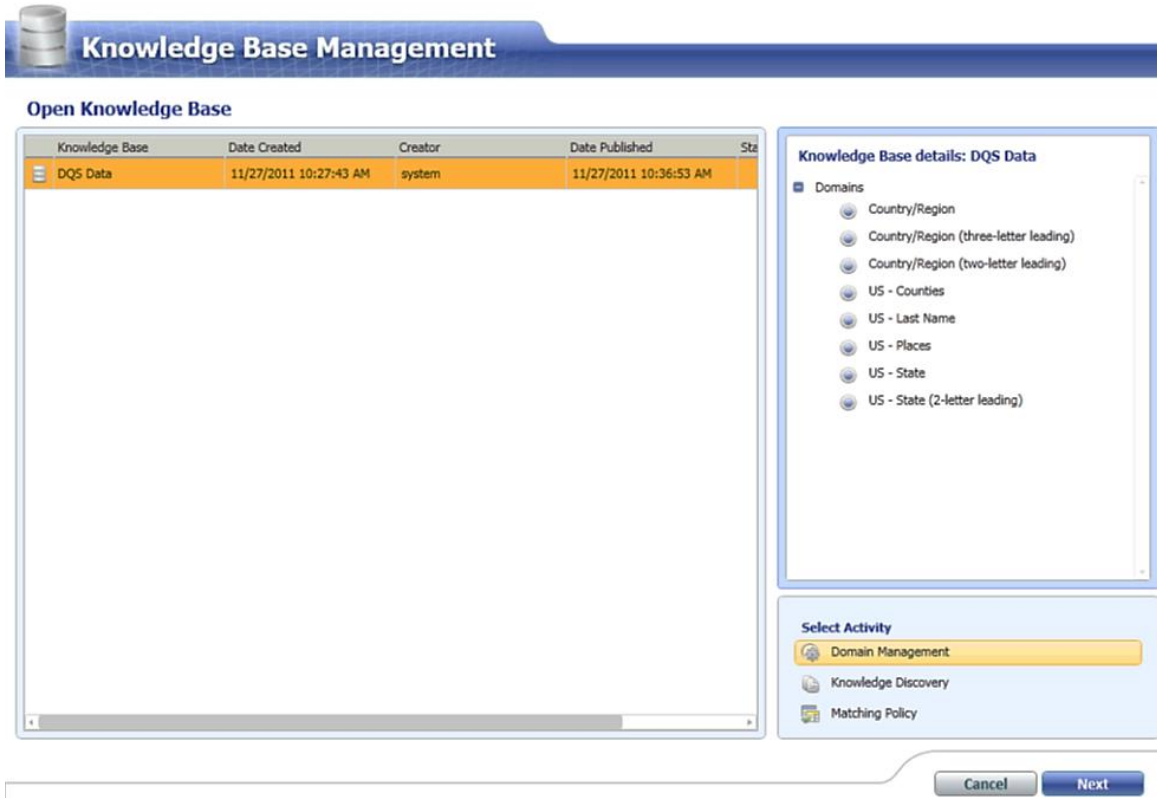
Enable Contained Databases
Enables or disables contained databases and authentication across the service instance.

[✓ Configured values] Running values

Data Quality Services

The Data Quality Services (DQS) feature of Microsoft SQL Server 2012 is a set of technologies that are used to measure and manage data quality through a combination of computer-assisted and manual processes. When your organization has access to high-quality data, your business processes can operate more effectively and managers can rely on this data to improve decision making. By centralizing data quality management, you also reduce the amount of time that people spend reviewing and correcting data.

The Data Quality Server (DQS) is the core component of the architecture that manages the storage of knowledge and executes knowledge-related processes. It consists of a DQS engine and multiple databases stored in a local SQL Server 2012 instance. These databases contain knowledge bases, stored procedures for managing the Data Quality Server and its contents, and data about cleansing, matching, and data-profiling activities. Data Quality Client is the primary user interface for the DQS, which is installed as a standalone application. Business users can use this application to work interactively with data quality projects, such as cleansing and data profiling.



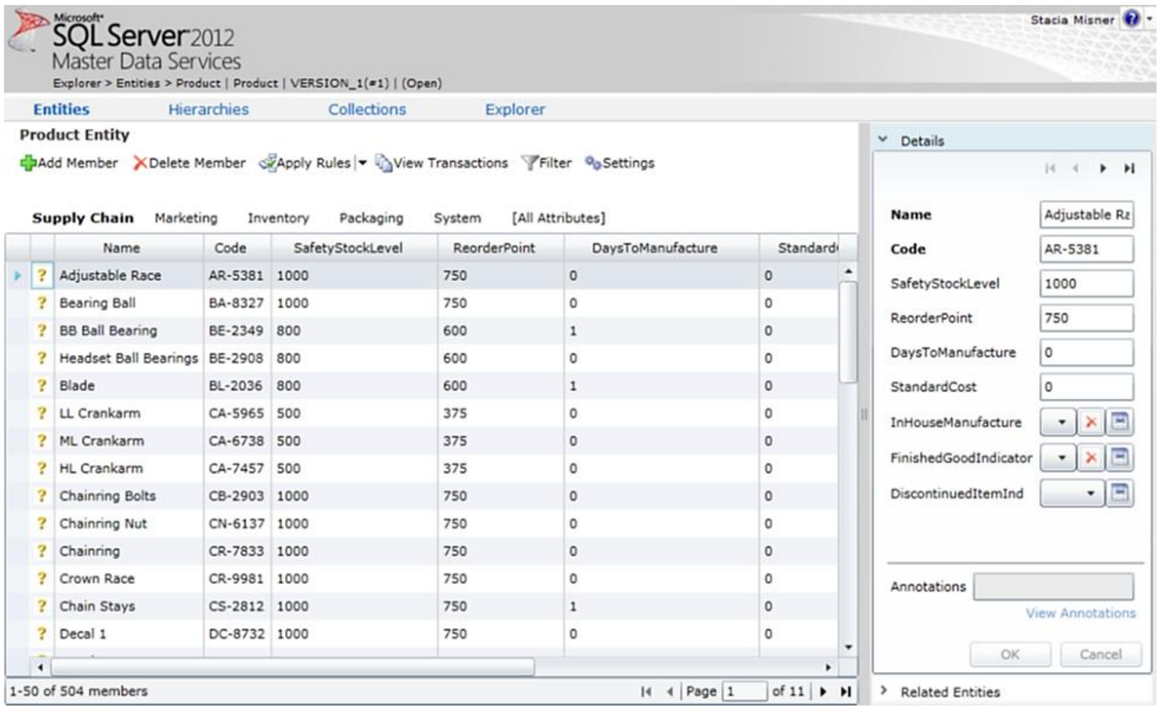
Knowledge Base	Date Created	Creator	Date Published	Sta
DQS Data	11/27/2011 10:27:43 AM	system	11/27/2011 10:36:53 AM	

Knowledge Base details: DQS Data

- Domains
  - Country/Region
  - Country/Region (three-letter leading)
  - Country/Region (two-letter leading)
  - US - Counties
  - US - Last Name
  - US - Places
  - US - State
  - US - State (2-letter leading)

Master Data Services

Master Data Services (MDS) is a feature in SQL Server 2012 setup instead of a separate installation, as it was in SQL Server 2008. Master Data Manager is the Web application that data stewards use to manage master data and administrators use to manage model objects and to configure security. The most extensive addition to MDS in SQL Server 2012 is the new user interface option that enables data stewards and administrators to manage master data inside Microsoft Excel®. In SQL Server 2008 MDS, the model permissions object tree also includes nodes for derived hierarchies, explicit hierarchies, and attribute groups, but these nodes are no longer available in SQL Server 2012 MDS. Instead, derived hierarchies inherit permissions from the model and explicit hierarchies inherit permissions from the associated entity. SQL Server 2012 also offers some new features for SharePoint® integration.



Explorer > Entities > Product | Product | VERSION_1(#1) | (Open)

Product Entity

Name	Code	SafetyStockLevel	ReorderPoint	DaysToManufacture	Standard
Adjustable Race	AR-5381	1000	750	0	0
Bearing Ball	BA-8327	1000	750	0	0
BB Ball Bearing	BE-2349	800	600	1	0
Headset Ball Bearings	BE-2908	800	600	0	0
Blade	BL-2036	800	600	1	0
LL Crankarm	CA-5965	500	375	0	0
ML Crankarm	CA-6738	500	375	0	0
HL Crankarm	CA-7457	500	375	0	0
Chainring Bolts	CB-2903	1000	750	0	0
Chainring Nut	CN-6137	1000	750	0	0
Chainring	CR-7833	1000	750	0	0
Crown Race	CR-9981	1000	750	0	0
Chain Stays	CS-2812	1000	750	1	0
Decal 1	DC-8732	1000	750	0	0

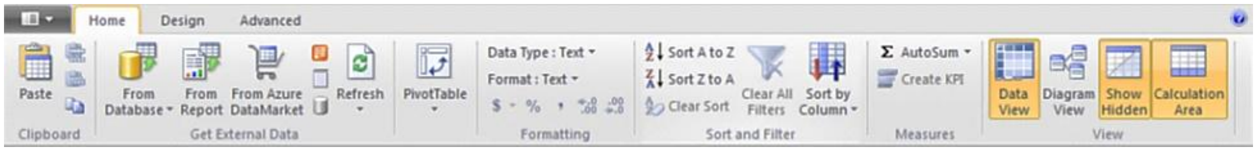
1-50 of 504 members Page 1 of 11

Details

Name	Adjustable Ra
Code	AR-5381
SafetyStockLevel	1000
ReorderPoint	750
DaysToManufacture	0
StandardCost	0
InHouseManufacture	
FinishedGoodIndicator	
DiscontinuedItemInd	
Annotations	

Power Pivot for Excel

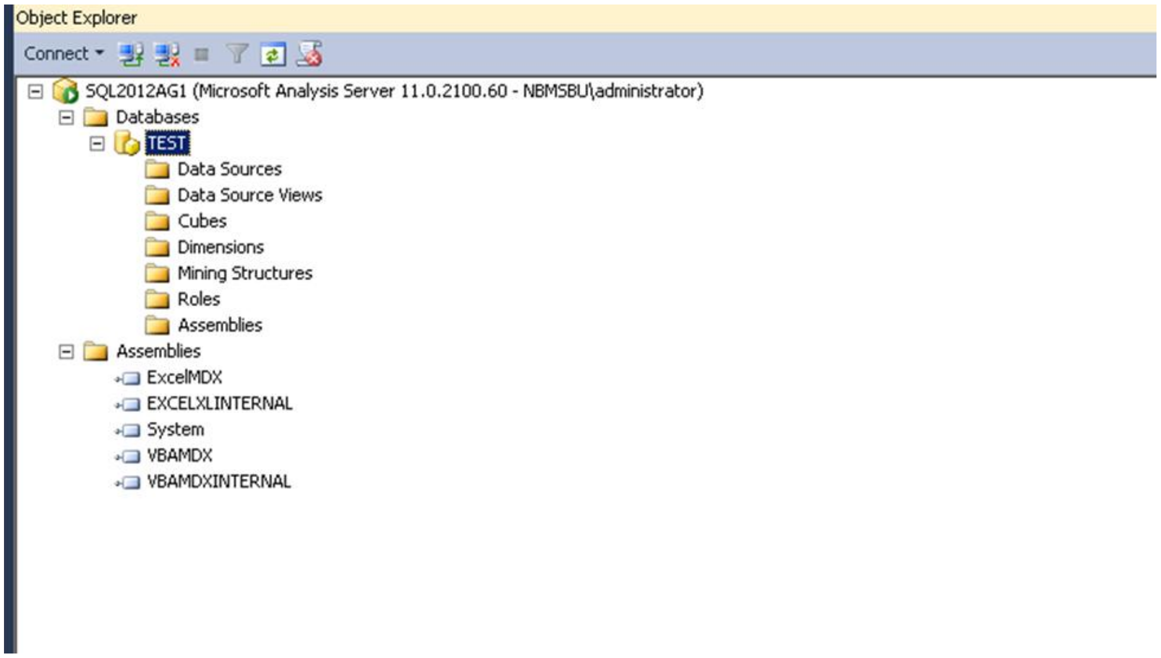
PowerPivot for Excel is a client application that incorporates SQL Server technology into Excel 2010 as an add-in product. The updated version of PowerPivot for Excel that is available as part of the SQL Server 2012 release includes several minor enhancements to improve usability. The key changes to PowerPivot for SharePoint offer a more straightforward installation and configuration process and a wider range of tools for managing the server environment. Now you can add numeric columns as a pivot table value for aggregation, and you can also add numeric columns to rows or to columns as distinct values. DAX is the expression language used to create calculated columns, measures, and key performance indicators for tabular models and PowerPivot models. The current release of SQL Server 2012 extends DAX to include many new statistical, information, logical, filter, and math functions.



Analysis Services Enhancements

In SQL Server 2012, an Analysis Services instance can run in one of the following server modes: multidimensional, tabular, or PowerPivot for SharePoint. Each server mode supports a different type of database by using different storage structures, memory architectures, and engines. Multidimensional mode uses the Analysis Services engine found in SQL Server 2005 and later versions. Both tabular mode and PowerPivot for SharePoint mode use the VertiPaq engine introduced in SQL Server 2008 R2, which compresses data for storage in memory at runtime. However, tabular mode does not depend on SharePoint like PowerPivot for SharePoint does.

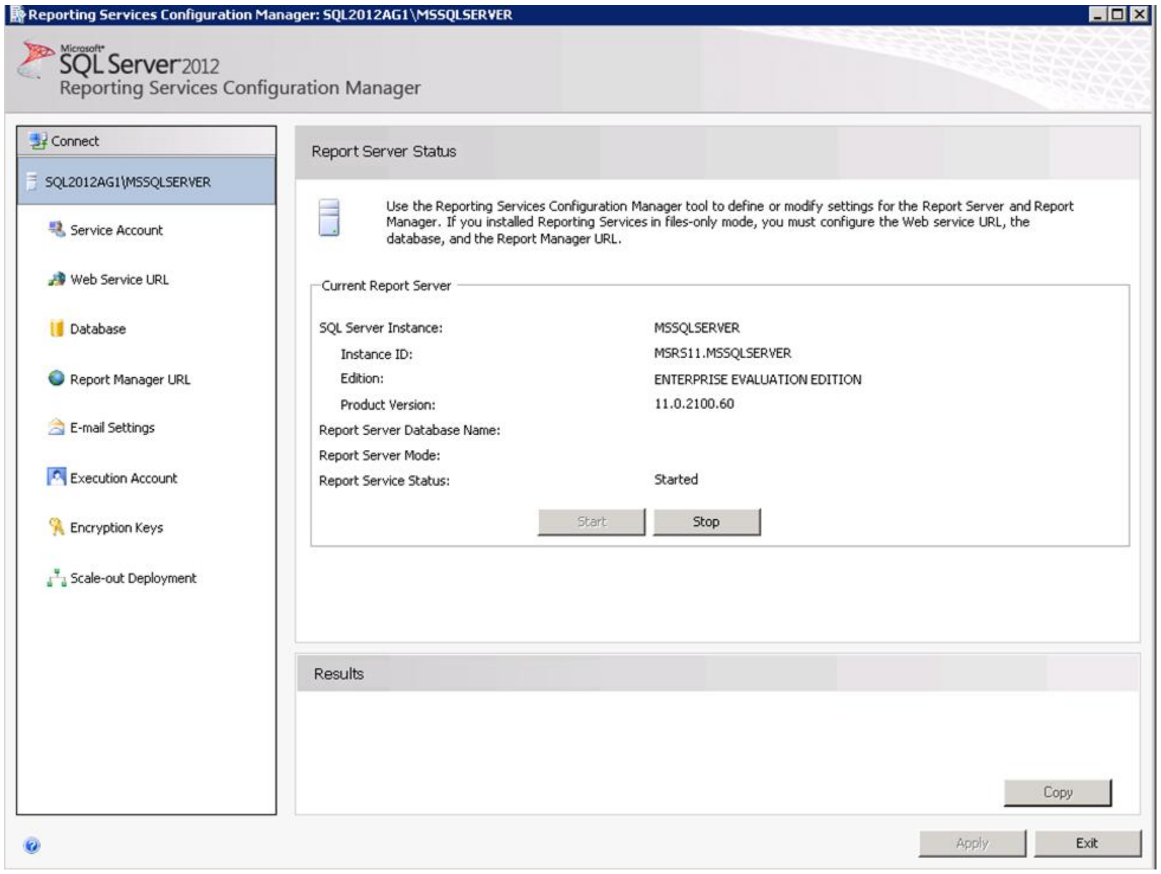
SQL Server Data Tools is the model development tool for multidimensional models, data-mining models, and tabular models. A tabular model is a new type of database structure that Analysis Services supports in SQL Server 2012. Key performance indicators are a special type of measure used to gauge progress toward a goal. At a minimum, each table in a tabular model has one partition, but it is possible to divide a table into multiple partitions in order to manage the reprocessing of each partition separately. The multidimensional online analytical processing engine uses a new type of storage for string data that is more scalable than it was in previous versions of Analysis Services. Specifically, the restriction to a 4-gigabyte maximum file size no longer exists, but you must configure a dimension to use the new storage mode. The BI Semantic Model (BISM) schema in SQL Server 2012 is the successor to the Unified Dimensional Model schema introduced in SQL Server 2005. BISM supports both an entity approach that uses tables and relationships and a multidimensional approach that uses hierarchies and aggregations. This release extends Analysis Management Objects and XMLA to support the management of BISM models.



Reporting Services Enhancements

SQL Server 2012 has some new enhancements for Reporting Services, including Excel 2010 Renderer and Word 2010 Renderer. Also, SharePoint Shared Service Architecture allows users to scale Reporting Services across Web applications and across SharePoint Server 2010 farms with fewer resources than in previous versions. Claims-based authentication is used to control access to Reporting Services reports, and SharePoint is used for backup and recovery processes for Reporting Services content.

Power View is the latest self-service feature available in Reporting Services. This browser-based Silverlight® application requires Reporting Services to run in SharePoint integrated mode using SharePoint Server 2010 Enterprise Edition. It also requires a specific type of data source—either a tabular model deployed to an Analysis Services server or a PowerPivot workbook deployed to a SharePoint document library. Another new feature is Data Alerts, used to create a data alert to e-mail a notification only when specific conditions in the data are true at a scheduled time. New controls like cards and tile are also included.



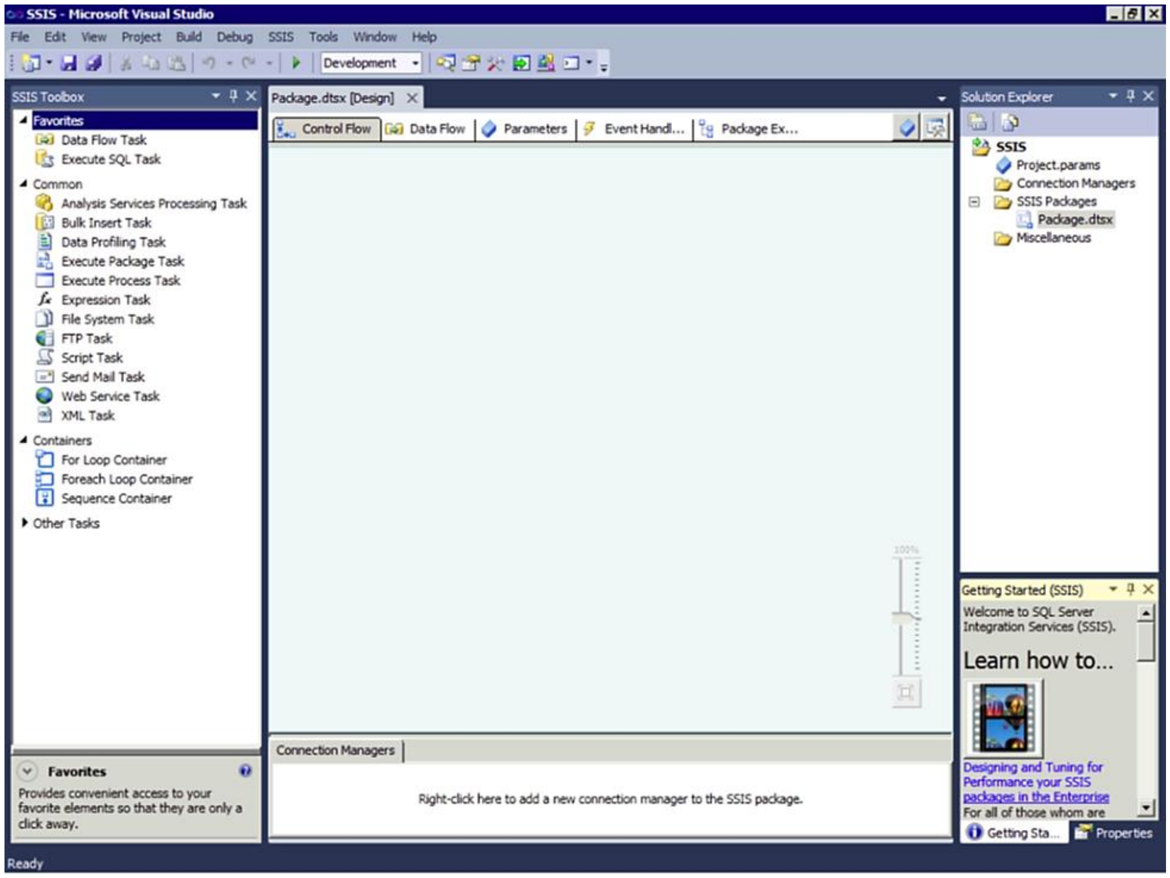
Use the Reporting Services Configuration Manager tool to define or modify settings for the Report Server and Report Manager. If you installed Reporting Services in files-only mode, you must configure the Web service URL, the database, and the Report Manager URL.

Current Report Server

SQL Server Instance:	MSSQLSERVER
Instance ID:	MSRS11.MSSQLSERVER
Edition:	ENTERPRISE EVALUATION EDITION
Product Version:	11.0.2100.60
Report Server Database Name:	
Report Server Mode:	
Report Service Status:	Started

SQL Server Integration Services Enhancements

The first change that you notice as you create a new SQL Server Integration Services (SSIS) project is that Business Intelligence Development Studio (BIDS) is now a Microsoft Visual Studio® 2010 shell called SQL Server Data Tools. The scripting engine in SSIS is an upgrade to Visual Studio Tools for Applications 3.0 and includes support for the Microsoft .NET Framework 4.0. There are three notable updates for the control flow: Expression Task, Execute Package Task, and Change Data Capture Task. Packages and related objects are stored securely in the catalog by using encryption. Only members of the new SQL Server database role, ssis_admin, or members of the existing sysadmin role have permissions to all objects in the catalog.



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.

Online Index Create, Rebuild, and Drop

With SQL Server 2012, indexes containing varchar(max), nvarchar(max), and varbinary(max) columns can now be created, rebuilt, and dropped as online operations.

Startup Options Relocated

A new Startup Parameters tab offers better manageability of the parameters required for startup. To open the Startup Parameters tab, right-click a SQL Server instance name in SQL Server Configuration Manager and select Properties.

Data-Tier Application Enhancements

A data-tier application (DAC) is a single unit of deployment that contains all of the database’s schema, dependent objects, and deployment requirements used by an application. With SQL Server 2012, DAC upgrades are performed in place, compared to the previous side-by-side upgrade process. In addition, 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 previous SQL Server releases.

Startup Options Relocated

A new Startup Parameters tab in SQL Server Configuration Manager offers better manageability of the parameters required for startup. To open the Startup Parameters tab, right-click a SQL Server instance name in SQL Server Configuration Manager and select Properties.

Cryptography Changes

SQL Server 2012 uses the AES encryption algorithm to protect the service master key and the database master key. When creating certificates, the maximum length of private keys imported from an external source is expanded from 3,456 bits to 4,096 bits. You can create certificates from bytes when using the `TRANSACT-SQL CREATE CERTIFICATE` procedure. Use the `FROM BINARY` option to specify the binary description of an ASN-encoded certificate.

New Permissions

New permissions are available for securing and managing authorization to elements within the database. New `GRANT`, `REVOKE`, and `DENY` permissions to a `SEARCH PROPERTY LIST` are available. New `GRANT`, `REVOKE`, and `DENY` permissions to `CREATE SERVER ROLE` and `ALTER ANY SERVER ROLE` are also implemented.

FileTable

FileTable, a new capability in SQL Server 2012, builds on FILESTREAM technology that was introduced with SQL Server 2008. FileTable offers Windows file namespace support and application compatibility with the file data stored in SQL Server. Also, when applications are allowed to integrate storage and data management in SQL Server, fulltext and semantic search are available over unstructured and structured data. FileTable is a new user table that is created in a database in the database engine. FileTable has a fixed schema and contains FILESTREAM and file attributes. Users can define indexes, constraints, and triggers; however, columns and system-defined constraints cannot be altered or dropped.

Statistical Semantic Search

SQL Server 2012 allows organizations to achieve deeper insight into unstructured data stored in the database engine. The FileTable and FILESTREAM feature in Microsoft SQL Server 2012 is a prerequisite. Statistical semantic search extracts pertinent key phrases by using statistics to identify the meaning of documents and similarities between them. This improvement is achieved by providing deep insight into unstructured documents stored in one or many databases that reside in the database engine.

Full-Text Search Enhancements

Full-text search in SQL Server 2012 offers better query performance and scale. It also introduces property-scoped searching functionality, which gives organizations the ability to search properties such as Author and Title without requiring developers to maintain file properties in a separate database.

Extended Events Enhancements

This enhanced user interface helps simplify the management associated with extended events. New extended events for functional and performance troubleshooting were introduced in SQL Server 2012. The new events are `page_allocated`, `page_freed`, and `allocation_failure`.

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 that requires approximately 50% to 60% fewer reboots when patching servers.

2.3 Database Sizing Considerations

There are several considerations to keep in mind when sizing a SQL Server environment. Keep track of the database (system and user databases) and transaction log files, file stream data, database compression (in SQL Server 2008 and 2008R2), and the SnapInfo directory if you are using SnapManager for SQL Server. Following are the additional considerations when configuring your SQL Server environment. For details on how to configure and deploy your SQL Server environment, see section 4 for a description of the physical layout for SQL Server.

- Data files (tables and indexes) are the primary files that are used by the SQL Server storage engine. Each database might have multiple files and be spread across multiple LUNs and volumes.
- Transaction log files operate in a sequential manner. The backup of the transaction log is stored as a copy in the SnapInfo directory.
- A new feature in SQL Server 2008 allows the database to be compressed, giving the database administrator the ability to stretch the growth of the storage system without severely affecting the database. The compression ratio varies depending on the actual data, so testing is important.
- Backing up tempdb is not required, because it is rebuilt when a SQL instance is rebooted. The size of tempdb is affected by features such as online database maintenance, version store, and database query plans.
- Filestream data (SQL Server 2008) allows the extension of the database for large objects such as bitmap images, text files, music files, and video and audio files. These can then be managed through insert, update, delete, and select statements, as well as standard SQL backup procedures.
- SnapInfo—calculating the SnapInfo directory if using SnapManager for SQL Server. This can be accomplished using the following formula:
  SnapInfo_LUN_Size = ((DB_TRANLog_Size × Daily_Chg_Rate × Days_per_TA-Log_Snapshots) + system databases (master, model, MSDB, and so on) + SnapInfo metadata)

When sizing your SQL Server environment, any additional data that can be gathered helps define the profile of the system that you are sizing. The database sizer tool has specific data requirements. Having a good understanding of how the system is expected to perform and/or has performed in the past will help enable the viability of a sizing solution.

2.4 Workload Sizing Guidelines

SQL Server I/O Overview

SQL Server is very sensitive to I/O latency issues due to the concurrent transactional nature of the SQL Server engine. SQL Server is built on a complicated system of row, page, extent, and table locks that provide transactional consistency throughout the SQL Server system. A poor I/O structure (for example, when I/O takes too long) causes resources to be held longer than necessary, resulting in blocking within the system. When this occurs, it frequently is not obvious that the I/O subsystem is the root cause.

SQL Server customers new to NetApp might not understand the differences between RAID-DP® technology and RAID 10. SQL Server performance has generally been centered around I/O. This performance can be improved by either increasing the number of spindles or making the spindles go faster. This is where traditional SQL environments have been slow to adopt RAID 5 or RAID 6. NetApp RAID-DP, a RAID 6 implementation, is a standard feature of Data ONTAP and prevents data loss—without excessive redundancy costs—in the event of a second drive failure.

- **SQL Server reads.** When reading data from the SQL Server, the client first goes to the buffer cache. If the data is not in the buffer cache, SQL Server goes to the I/O subsystem to retrieve the data. The statement does not complete until 100% of the data is read; the user connection or process remains in an I/O wait state until completion.
- **SQL Server writes.** The user writes to the transaction log and the buffer cache. If the data to be modified is not already in the buffer cache, then it must be read into the buffer cache from the I/O

subsystem. The buffer manager enables the transaction log to be written to first, before changes are written to the database. This is known as write-ahead logging. When the user makes the change and the `COMMIT` is executed, a log write occurs showing that the change took place, allowing the `COMMIT` to complete. Once the `COMMIT` is complete, the user process can continue on to the next stage or command without having to wait for the changes to be written to the disk. `ROLLBACK TRANSACTION` follows the same process as the `COMMIT`, but in reverse. The buffer manager moves the data from the cache to the disk. It keeps track of log sequence numbers (LSNs) for each log record.

- **Transaction log.** The SQL Server transaction log is a write-intensive operation that is sequential in nature. The transaction log is used to provide recoverability of data in case of database or instance failure.

Online Transaction Processing

The Online Transaction Processing (OLTP) database system is the SQL Server environment most concerned about getting the greatest number of transactions through the system in the least amount of time. Examples of different types of OLTP systems include Web order systems and manufacturing tracking systems. OLTP systems can have very large volumes of transactions per second, and for the OLTP system it is all about throughput. For these transactions to take place, SQL Server relies on an efficient I/O subsystem. According to the Microsoft SQL Server best practices article written by Mike Ruthruff, an OLTP transaction profile is composed of the following pattern:

- OLTP processing is generally random in nature for both reads and writes issued against data files.
- Read activity (in most cases) is consistent and uses point queries; it does not consist of large time-consuming queries.
- Write activity to the data files occurs during checkpoint operations (frequency is determined by recovery interval settings).
- Log writes are sequential in nature with a varying size that depends on the nature of the workload (sector aligned up to 60KB).
- Log reads are sequential in nature (sector aligned up to 120KB).

Decision Support System

The SQL Server best practices article describes the nature of the decision support system (DSS) as follows:

- Reads and writes tend to be sequential in nature and are generally the result of table or index scans and bulk insert operations.
- I/O size varies but is generally larger than 8KB. Read-ahead is any multiple of 8KB up to 256KB (1024KB for Enterprise edition). Bulk load operations are any multiple of 8KB up to 128KB.

Mixed

In mixed environments you must take the blended approach for managing the I/O for the SQL Server. The reads or writes work out to an average, which is closer to a ratio of 40% through 60% to 65% through 35%. Arranging the reads and the writes according to the proper mix of your environment defines the performance profile for the database system. Even though mixed environments vary, a 75/25 mix for reads and writes is a good place to start. Here, you are looking for the balance between reporting and online transactions.

Table 1) Read/write percentages.

Database System Type	Read Percentage	Write Percentage
DSS	80%	20%

Database System Type	Read Percentage	Write Percentage
OLTP	66%	33%
Mixed	75%	25%

2.5 Approximating Data Sizes

Estimating I/OS

Estimating the number of I/Os required for a system is crucial when sizing a database. This exercise helps the administrator understand how to keep the database instance performing within acceptable limits. You must estimate I/Os when you are not able to get the actual physical I/O numbers for the system. This is typically the case in new systems that are in the process of being constructed. Following are the formulas for estimating I/Os.

To estimate I/Os for a new database system without access to the system:

1. Estimate the number of transactions for a given period size.
2. Multiply the number of transactions by the 0.85 saturation rate and then divide that by the number of seconds in a day.
   The seconds in a day are determined by the hours of operation for the database. If the database operates in a 24-hour environment, the number is 86,400. The formula for estimating the number of I/Os is:

   Total I/Os = (estimated number of transactions × 0.85)/seconds in a day
   For example, if there are 40,000 transactions on a system that operates 24 hours per day, the formula is as follows:
   (40,000 × 0.85)/86,400 = 0.3935 IOPS
3. After determining the I/Os required, determine what kind of system you plan to deploy. If deploying an OLTP system, determine the read and write I/Os by multiplying the number of I/Os by the percentage of reads or writes. Table 1 lists the I/O percentages for each type of system.
   The formula for I/Os in megabytes is ((number of transactions × 0.85)/seconds in a day) × type % = I/O megabytes.
   For example, to determine the reads and writes for a DSS system, the formula is as follows:
   ((40,000 × 0.85)/86,400) × 0.80) = 0.3148MB reads
   ((40,000 × 0.85)/86,400) × 0.20) = 0.0787MB writes

Gathering Statistics

When sizing for an existing database environment, understanding the type of workload and interpreting the statistical data are helpful. The database sizer tool does not determine the type of system based on the data uploaded. By comparing the physical reads and writes to the numbers from Table 1, you can estimate the type of system you are evaluating. It is important to gather statistics during periods of peak stress on the system. PerfMon allows you to see the high-water marks for the time frame in which you monitor the system. For more information, see the Perfmon section in the Appendixes. Enter the highest points of the reads and writes values in the database sizer tool.

2.6 Building Growth and Capacity Models

Capacity Planning

According to the “Microsoft SQL Server 2005 Administrator's Companion,” the fundamentals of sizing and capacity planning are based on queuing theory. Queuing theory has a few terms that you should understand: service time and wait time, which is also known as queue time. The key formula is:

Response time = service time + queue time

Queuing theory has two key components according the “Microsoft SQL Server 2005 Administrator's Companion”:

- The chance of queuing increases exponentially as you near capacity of the resource.
- The response time is equal to the sum of the service time plus the queue time. (See Chapter 6 [pages 108–111] in the “Microsoft SQL Server 2005 Administrator's Companion.”)

Monitoring the I/O subsystem with PerfMon is an excellent way to develop a behavior profile for the I/O subsystem of the SQL Server. Physical disk counters are preferred for understanding your I/O.

- **Disk reads/sec.** Read IOPS for the disk drive or drives
- **Disk transfers/sec.** Total read plus write IOPS for each disk drive
- **Disk writes/sec.** Write IOPS for each disk drive
- **Avg. disk sec/read.** Read latency or average time for each read operation
- **Avg. disk sec/write.** Write latency or average write time for each operation
- **Average disk queue length.** Average of both read and write requests that are in the queue

For a well-tuned I/O subsystem, the read and write latency should be within 5 to 10 ms. If the latency is 20 ms, the system might be experiencing an I/O issue. Latency exceeding 30 ms is in the unacceptable range. As all values are relative, the effect of measured latency depends on the type of operations taking place on your storage system and what they mean to your customers.

Questions to ask the customer about capacity planning:

- What is the size of the database or databases?
- How many users will be accessing the databases?
- When are the peak times for database activity?
- What types of activity will be taking place on each database? OLTP, DSS, batch jobs, misc.?
- What is the rate of growth of the data?
- What is the rate of growth of users?
- Is this a dedicated SQL solution, or is it a multipurpose solution?
- What are the needs for high availability, scalability, and disaster recovery?
- Will the databases be replicated as part of the new strategy?

Growth Models

Growth models allow administrators to develop projections for the growth that databases will potentially see. The only growth model that is 100% accurate is the historical model, but this model only tells you where you have been and not where you are going. Hence, administrators must take that historical data and plug it into the model that best fits their requirements.

The limitation with any growth model is the amount of data that you must plug into the model. The more data about the system that you can gather, the better your projection will be. This technical report examines two different models: the linear growth model and the geometric growth model (Hotek and Makin).

Linear Growth Model

The linear growth model builds a straight line model for the future. It is based on the amount of growth in the system. As with all models, the need to determine the amount of growth that the system has historically seen over a given period of time is still an important aspect of system management.

The formula for the linear growth model is as follows:

Future usage = current usage + (growth amount × number of periods)

Using the formula, determine the total size of the system as it currently stands. One procedure for gathering this information is by using sp_helpdb. This gives you the totals for each database. Next, take the sum of disk space used by all databases. The growth amount is found by tracking the size over a period of time. Take this growth amount and apply it for the number of months looking into the future.

For example, for a system that is currently 400GB in size with a growth amount of 50GB per month over a three-month period (Table 2), the formula is as follows:

Estimated future usage = 400GB + (50GB × 3) = 550GB in 3 months

Table 2) Example 1.

Parameter	Metric
Current usage	400GB
Growth amount	50GB/month
Number of periods	3 months

The conclusions drawn from this exercise provide an idea of how much the capacity of the database system is expected to grow in the three-month time frame. This allowance for future growth gives an educated estimate of the sizing needs for a particular database.

Geometric Growth Models

Exploring the geometric growth model allows finer tuning of changes in the system. One way to look at the growth rate is as a percentage of growth over a period of time. The sampling of that time frame for total database size then compares the differences from the last sampling period. If there is no historical sizing data, then you can look at the difference between the backup sizes for each database over the history of backups in the msdb database. As long as the databases in the system have been backed up regularly, you can estimate the growth rate over a given sample period.

To estimate the growth, insert the current usage and growth rate into the following formula along with the number of months for the projection for a geometric view of the rate of growth.

Future usage = current usage × (1 + growth rate/current usage) $^{\text{number of periods}}$

For example, applying the values from Table 2, the formula is as follows:

Estimated future usage = 400 × (1 + 50/400)^3 = 569.53GB after 3 months

3 NetApp Simplifies SQL Server Data Management

Database managers in today’s business environment are looking for better ways to manage their database assets. Better management of database systems gives organizations the ability to meet their SLAs and reduce costs. Meeting SLAs through better management allows higher efficiency and better use of database assets. This report discusses high availability, database mirroring, SnapMirror® technology, and SQL Server replication.

3.1 High Availability

NetApp storage systems are uniquely scalable and highly available with the ability to scale from small database systems to very large data warehouses. Fabric-attached storage (FAS) systems offer a rich set of features with the inherent redundancy of robust hardware components. Clustered FAS systems are available for even higher levels of availability. Innovations like double parity RAID (RAID-DP) enhance availability by protecting against secondary drive failures that might occur during RAID reconstructs. For more information about RAID-DP, refer to TR-3298: NetApp Data Protection: Double-Parity RAID for Enhanced Data Protection with RAID-DP. The SnapDrive family of products offers multipath I/O (MPIO) solutions for both iSCSI and Fibre Channel protocols, providing redundant paths from hosts to NetApp FAS systems. For more information about MPIO, visit the Microsoft MPIO site.

What does database mirroring protect you from versus Microsoft clustering services, and when do you use it rather than SnapMirror? Database mirroring is great for protecting against computer room or server rack failures. It requires redundant storage and like systems whereas Microsoft clustering protects a database from server failure but not from an I/O subsystem failure. SnapMirror plays a great part in moving over very long distances and allows you to move over the entire SQL Server instance but not individual databases.

These capabilities enhance the key features that allow highly available systems. When applying these concepts to Microsoft SQL Server there are several approaches that can be considered. The considerations that this best practice guide discusses are database mirroring, SnapMirror replication, SQL Server replication, and Microsoft Clustering Services. This guide also covers some considerations for business continuance and disaster recovery.

Database Mirroring

Database mirroring has been a part of Microsoft SQL Server since 2005. It is part of both SQL Server 2005 and 2008. Database mirroring is one of the key tools used by SQL Server to achieve high availability. It is one of those tools that not only provides high availability, it provides a level of data protection from catastrophic system failure as well. Management of a mirrored database requires planning on the client and infrastructure side to deal with the unique issues related to mirroring. Database mirroring requires the same storage requirements for both sides of the mirror. Additional considerations involve the database client. If the client is not mirroring aware, then failover to the mirror requires some type of intervention by the client in switching from the primary database to the mirror.

When maximizing SQL Server performance between mirrors, it is important to determine that both the primary and the target have similar LUN structures. An additional consideration that must be made for database mirroring is the need for the proper bandwidth between the different sites. Database mirroring comes in two flavors: “Full Safety On” and “Full Safety Off,” which are synchronous and asynchronous, respectively. Full Safety On, or synchronous, works as a two-phased commit from the client to the primary database to the target database, requiring a fast connection between the primary database and the target so that what is committed to the client is written on both the primary and the mirror. Full Safety Off, or asynchronous, is more of a store-and-forward approach to committed transactions. This can result in lost transactions under certain circumstances.

To use SnapManager for SQL Server with database mirrors requires that you have it installed on both the primary database as well as the target. The mirrored database will appear to be offline or in recovery mode. This does not allow a consistent Snapshot™ copy until the database mirror set is failed over to the target site, bringing the target database online and open for transactions. If you need to recover, then you must recover from the Snapshot copies that were made on what was initially the primary site.

Database mirroring can also be used as a migration strategy from one platform to another. Migrating databases from one platform to another has the potential for data outages while the move occurs. By mirroring the database you can move from a non NetApp FAS system to a NetApp FAS system, while minimizing the effect on the production database. Then, at a later time, the failover can take place moving the database clients onto the new system.

SQL Server and SnapMirror

The use of SnapMirror elevates the ability of businesses to recover from major data center disasters and outages. The time to look at SnapMirror, both synchronous and asynchronous, is when you need geographical DR. The best practice for SnapMirror with SQL Server is defined in TR-3604: NetApp Disaster Recovery Solution for Microsoft SQL Server.

Many organizations have requirements for mirroring SQL Server databases to geographically dispersed locations for data protection and business continuance purposes. Like Snapshot technology, SnapMirror works at the volume level, so proper grouping of databases by the high-availability index (discussed in further detail in the Consolidation Methodologies section of the Appendixes) can be particularly useful. Locating one database per FlexVol® volume provides the most granular control over SnapMirror frequency as well as the most efficient use of bandwidth between the SnapMirror source and destination. This is because each database is sent as an independent unit without the additional baggage of other files, such as tempdb, or databases that may not share the same requirements. While sharing LUNs and volumes across databases is sometimes required because of drive letter limitations, every effort should be made to group databases of like requirements when specific mirroring and backup policies are implemented.

Note: Data ONTAP 7G can support up to 200 FlexVol volumes per storage system and 100 for clustered configurations. Also, drive letter constraints in MSCS configurations may require combining log and data into a single LUN or FlexVol volume.

SQL Server Replication

The need to replicate a smaller subset of the database using SQL Server replication is the way to accomplish moving smaller sets of data to multiple locations. When sizing for replication it is important to remember that the size of the distribution database is determined by the retention policy for the replication publications. The default is 72 hours when all transactions that take place on replicated tables are stored in the distribution database. In order to properly size the distribution database you must have the delta change rate on the tables that are being replicated for those 72 hours. The retention period can be changed by altering the article publication.

SQL Server Clustering

The use of a SQL Server cluster is a proven method to protect against host failures. The use of a cluster allows quick failover on a SQL Server instance level. The cluster alone is not protection against disk failure. To overcome this challenge you can use SnapMirror to build a fast way to provide a redundant storage failover with a simple script that uses simple cluster and SnapDrive Command Line Interface (SDCLI) commands to manage both failover of the cluster to additional nodes and failover of the cluster storage components to a SnapMirror location. A sample of such a script is provided by John Jones, a NetApp solution architect.

Failover

```
ECHO.
Cluster res "SQL Server" /offline
Cluster res "Disk E:\" /offline
Cluster res "Disk F:\" /offline
Cluster res "Disk E:\" /delete
Cluster res "Disk F:\" /delete
ECHO
ECHO Connecting to DR Storage...........please stand-by
SDCLI disk connect -p r200:/vol/sqlonlinedatadr/data1 -d e -dtype shared -I w2k8-node1 iqn.1991-
05.com.microsoft:w2k8-node1.demo.local w2k8-node2 iqn.1991-05.com.microsoft:w2k8-node2.demo.local
w2k8-node3 iqn.1991-05.com.microsoft:w2k8-node3.demo.local w2k8-node4 iqn.1991-
05.com.microsoft:w2k8-node4.demo.local -e "SQL1"
SDCLI disk connect -p r200:/vol/sqlonlinelogdr/log1 -d f -dtype shared -I w2k8-node1 iqn.1991-
05.com.microsoft:w2k8-node1.demo.local w2k8-node2 iqn.1991-05.com.microsoft:w2k8-node2.demo.local
w2k8-node3 iqn.1991-05.com.microsoft:w2k8-node3.demo.local w2k8-node4 iqn.1991-
05.com.microsoft:w2k8-node4.demo.local -e "SQL1"
Cls
ECHO.
ECHO Bringing SQL Cluster Online....
Cluster res "Disk E:\" /online
Cluster res "Disk F:\" /online
Cluster res "SQL Server" /online
Cluster res "SQL server agent" /online
Color 2f
EcHO **************************************************************************
ECHO.
ECHO                                 !!!IMPORTANT!!!
ECHO.
ECHO                      "The DR storage of the SQLONLINE"
ECHO.
ECHO                 HAS BEEN REPLICATED TO NETAPP STORAGE SYSTEM (R200) AND
ECHO                     SUCCESSFULLY MOUNTED
ECHO.
ECHO
ECHO **************************************************************************|
ECHO.
Pause
```

Failback

```
snapmirror resync -f -S fas3050b:sqlonlinedata1 R200:sqlonlinedatadr
snapmirror resync -f -S fas3050b:sqlonlinelog1 R200:sqlonlinelogdr
sdcli disk disconnect -d e
sdcli disk disconnect -d f
SDCLI disk connect -p fas3050b:/vol/sqlonlinedata1/data1 -d e -dtype shared -I w2k8-node1
iqn.1991-05.com.microsoft:w2k8-node1.demo.local w2k8-node2 iqn.1991-05.com.microsoft:w2k8-
node2.demo.local w2k8-node3 iqn.1991-05.com.microsoft:w2k8-node3.demo.local w2k8-node4 iqn.1991-
05.com.microsoft:w2k8-node4.demo.local -e "SQL1"
SDCLI disk connect -p fas3050b:/vol/sqlonlinelog1/log1 -d f -dtype shared -I w2k8-node1 iqn.1991-
05.com.microsoft:w2k8-node1.demo.local w2k8-node2 iqn.1991-05.com.microsoft:w2k8-node2.demo.local
w2k8-node3 iqn.1991-05.com.microsoft:w2k8-node3.demo.local w2k8-node4 iqn.1991-
05.com.microsoft:w2k8-node4.demo.local -e "SQL1"
Cls
ECHO.
ECHO Bringing SQL Cluster Online....
Cluster res "SQL Server" /offline
Cluster res "SQL Server" /online
Cluster res "Sql Server Agent" /online
```

Best Practice: SQL Server Clustering

Verify that the cluster can fail over and that all the resources are available before you initially establish the SQL Server instances.

3.2 Improved Performance and Asset Utilization

FlexVol technology introduced in Data ONTAP 7G makes it possible to optimize storage utilization by logically partitioning NetApp storage (aggregates) into smaller virtualized volumes (FlexVol volumes). Many SQL Server environments support small databases with very high I/O appetites, which often requires many more disk drives for throughput than required for space. FlexVol volumes make it possible to create a base aggregate container with many disk drives and logically partition the aggregate into smaller virtualized volume structures. By combining the same number of spindles previously sliced up across many conventional volumes, much higher throughput and storage asset utilization can be achieved while still maintaining the benefits of discreet dedicated volumes. FlexVol volumes can be created, expanded, or reduced in mere seconds, regardless of size. For more information about FlexVol volumes as they relate to databases, see TR-3410: Data ONTAP 7G: FlexVol and FlexClone for Microsoft SQL Server and TR-3373: Data ONTAP 7G—The Ideal Platform for Database Applications.

3.3 Accelerated Development and Test with Rapid Cloning

The NetApp solution for rapidly deploying a development and test database solution uses the functionality of NetApp FlexClone® technology to create read-write clones of any database in a SQL Server instance. The solution is outlined in the following documents:

- WP-7078: Development and Test Solution for Microsoft SQL Server
- WP-7014: Rapid Database Development and Deployment

The use of FlexClone volumes allows you to build a SQL Server database on either the existing SQL Server instance or some other instance on a separate host. These databases can be migrated from a FlexClone volume by splitting the clone from the parent. See the "Data ONTAP Storage Management Guide" for the correct syntax for your version of Data ONTAP.

3.4 Fast Backup and Restore with SnapManager for SQL Server

SnapManager for SQL Server allows the activation of NetApp Snapshot copies as a means to protect data in a very timely fashion. The use of Snapshot technology greatly reduces the time it takes to back up large SQL Server instances. This also enables the optimum use of the storage system by tracking only the changing blocks of data.

Installing and Configuring SMSQL

This section discusses the steps needed to install SMSQL on SQL Server 2005 and 2008 hosts.

1. Prerequisites
   The following are the prerequisites for installing SMSQL:
   a. Windows PowerShell™ 1.0.
   b. SnapRestore® data recovery software should be licensed on the storage controller.
   c. Operations Manager 3.7 (only if you want to use the backup archival feature of SMSQL).
   d. Data ONTAP 7.1.3 or above is needed only if you want to use the backup archival feature of SMSQL.
   e. SnapDrive for Windows® 6.0.1 (enable integration with Protection Manager during installation).
   f. If the host system is running SQL Server 2005, Microsoft Data Access Components (MDAC) 2.8 SP1 must be installed. Windows Server® 2003 SP1 and SP2 include MDAC 2.8 SP2.
2. Platform support
   Refer to the platform support documented in the "Installation and Administration Guide." Some of the key highlights for platform support are:
   a. If an SMSQL instance is being used to remotely administer other SMSQL instances, then there is no need to install SnapDrive on that host.
   b. Upgrading SMSQL from older versions invalidates existing scripts based on the older versions due to changes in the CLI commands.
   c. This means that only iSCSI-based LUNs are supported on VMware® and Hyper-V™ guest operating systems.
3. Sizing the SnapInfo volume
   The total size of SnapInfo can be calculated with the following formula:
   SnapInfo_LUN_Size = ((DB_Size × Daily_Change_Rate × NumberOfOnlineSnapShots) + TotalSizeOfSystemDatabases + SnapInfoMetadata)

   The SnapInfoMetadata rate is approximately .01 of the entire SQL Server instance.

   This formula is used for each SQL Server instance. If there are multiple SQL Server instances on the host, then this process is required for each SQL Server instance.

4. Installing SMSQL on a Microsoft cluster
   Apart from the steps listed in the "Installation and Administration Guide" for installing SMSQL on an MSCS, the following points must be noted:
   a. SMSQL is NOT a cluster-aware application.
   b. SMSQL and all its dependencies must be installed on each cluster node.
   c. Each clustered instance must have its own SnapInfo LUN, which should be a shared disk.
   d. Place the SnapInfo LUN resource in its appropriate SQL Server cluster group.

4 Database and Storage Logical and Physical Layout

The proper layout of a SQL Server database is necessary for proper performance and management of the SQL Server infrastructure. TR-3696: Microsoft SQL Server Relational Engine: Storage Fundamentals for NetApp Storage provides guidance for the layout of a SQL Server on NetApp storage. This technical report outlines in detail the fundamental storage layout for optimal performance on NetApp storage systems.

Best Practice: FlexVol Volumes
Use FlexVol volumes when more granular control is required.

When backup requirements are unique to each database, there are two possible paths that can help optimize Snapshot copies and transaction log backups:

- **Most granular control.** Create separate FlexVol volumes for each database for more granular control of backup schedules and policies. The only drawback to this approach is a noticeable increase in administration and setup.
- **Second-most granular control.** Try to group databases with similar backup and recovery requirements on the same FlexVol volume. This can often provide similar benefits with less administration overhead.

Best Practice: Sharing Volumes
Avoid sharing volumes between hosts.

With the exception of Microsoft Cluster Services (MSCS), NetApp does not recommend letting more than a single Windows server create and manage LUNs on the same volume. The main reason for this is that Snapshot copies are based at the volume level and issues can arise with Snapshot copy consistency between servers. Sharing volumes also increases the chances for busy Snapshot copies.

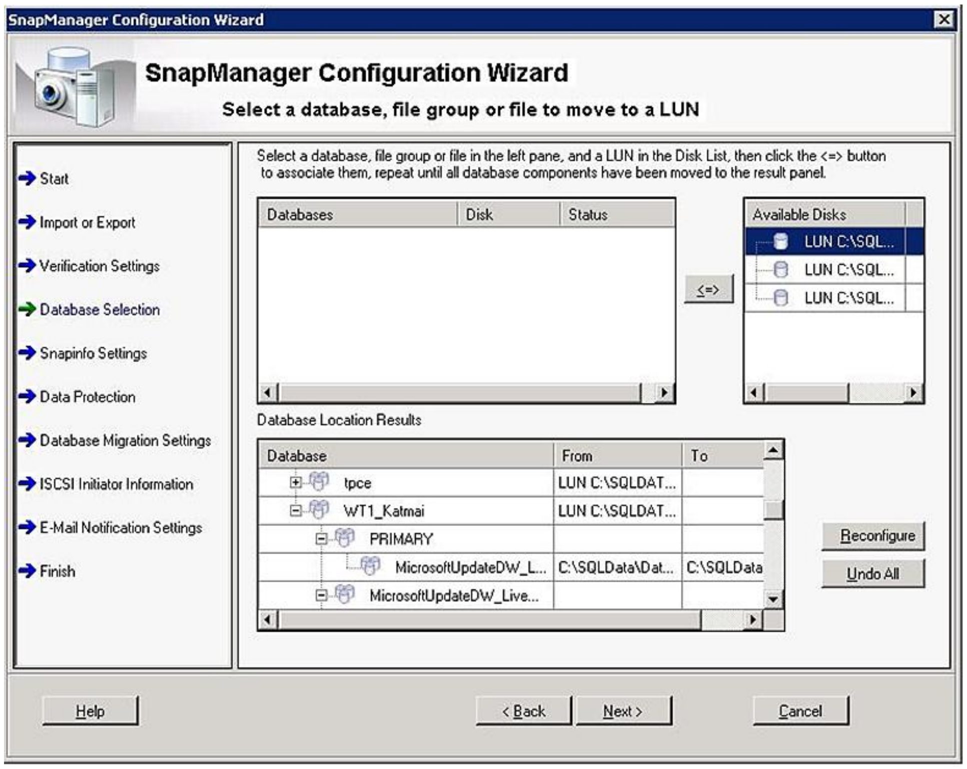
4.1 Migrating from Drive Letters to Volume Mount Points

Existing SMSQL configurations have data LUNs mapped to drive letters. When migrating, if the physical data need not be copied, then it should be migrated to a reference mount point. If the migration is performed on a normal mount point, then data will be physically copied to the new location. The affected databases are taken offline briefly to remap their paths. This process saves time by not having to copy physical data and by minimizing the time that the databases are offline.

When migrating from drive letters to volume mount points, you lose the ability to restore those LUNs from previously taken backups. It is a best practice to take a full backup of all databases both before and after a successful migration. Once the migration is successful, you can discard previous backups. You cannot restore your databases from these backups. Consult with your NetApp Global Services representative (1 [888] 4NETAPP) to assist in planning a migration.

Step 1

A database is residing and attached to LUN M. Add a reference (`C:\Mntp1\db`) to the existing LUN M using SnapDrive.



SnapManager Configuration Wizard

SnapManager Configuration Wizard

Select a database, file group or file to move to a LUN

Select a database, file group or file in the left pane, and a LUN in the Disk List, then click the <=> button to associate them, repeat until all database components have been moved to the result panel.

Databases	Disk	Status

Available Disks
LUN C:\SQL...
LUN C:\SQL...
LUN C:\SQL...

Database Location Results

Database	From	To
tpce	LUN C:\SQLDAT...	
WT1_Katmai	LUN C:\SQLDAT...	
PRIMARY		
MicrosoftUpdateDW_L...	C:\SQLData\Dat...	C:\SQLData
MicrosoftUpdateDW_Live...		

Step 2

Run the SMSQL configuration wizard, highlight the database, and click Reconfigure. The SMSQL configuration wizard lists all references to the same LUN on the Disk List; each of those LUN references has a label that lists any other reference to the same LUN:

- LUN M <`C:\Mntp1\db\`>
- LUN `C:\Mntp1\db\` <M>

Step 3

Select `C:\Mntp1\db` (second reference) and associate it with the database. Click Next to proceed and complete the configuration wizard. The database will now be attached to `C:\Mntp1\db` instead of M.

4.2 Managing Space

Snapshot Copies

It is beyond the scope of this paper to explore Snapshot copies in great detail. However, it is necessary to understand the fundamentals of the unique features of NetApp Snapshot functionality and how it relates to Microsoft SQL Server 2000 and 2005. For more information about NetApp Snapshot technology, refer to TR-3001: A Storage Networking Appliance.

NetApp Snapshot backups occur in a matter of seconds, and typically each copy consumes only the amount of data that has changed since the last copy was created. Thus Snapshot copies consume minimal disk space while providing up to 255 online point-in-time images. The amount of disk space consumed by an individual Snapshot copy is determined by two factors:

- The rate data changes within file systems (in MB/sec or MB/hour, for example)
- The amount of time that elapses between creation of Snapshot copies

The measure of change (in megabytes, gigabytes, and so on) that occurs in the volume between the creation of Snapshot copies is often referred to as the delta. The total amount of disk space required by a given Snapshot copy equals the delta changes in the volume along with a small amount of Snapshot metadata.

Fractional Space Reservation

Fractional Space Reserve (FSR) can be managed from 0 to 100%; the key to this is remembering that FSR provides the ability to take Snapshot copies and manage the writable space. SQL Server in a managed environment can easily support FSR set to 0. It is important to make sure that you follow the guidance for FSR set forth in TR-3483: Thin Provisioning in a NetApp SAN or IP SAN Enterprise Environment. This technical report outlines the key items for Microsoft applications running on NetApp storage systems.

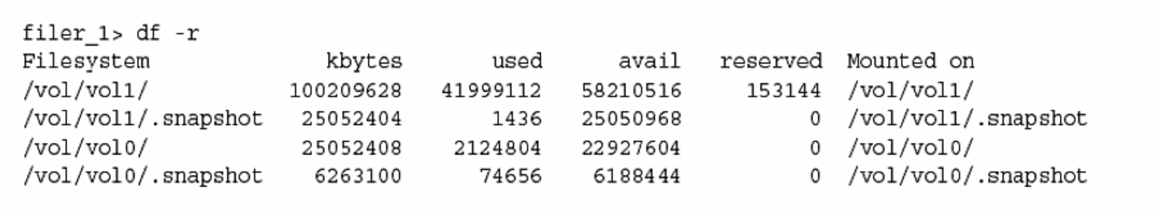
Best Practice: Fractional Space Policy

The threshold value for the policy "deletion of backup sets" should always be less than that of "dismount databases."

Best Practice: Fractional Space Reservation

When a LUN is fully space reserved (fractional space reservation set to 100%), write operations to that LUN are protected against failure caused by an out-of-space condition due to Snapshot copy disk space consumption. If you do require a fractional space reservation policy, work with your NetApp Support representative. The representative can help implement a fractional space reservation policy that fits your environment.

Fractional reserve has an effect on the output of the **`df -r`** command. The **`df -r`** command reports the amount of space reserved to allow LUNs to be changed.



```
filer_1> df -r
Filesystem              kbytes       used      avail   reserved  Mounted on
/vol/vol1/           100209628   41999112   58210516     153144  /vol/vol1/
/vol/vol1/.snapshot   25052404       1436   25050968          0  /vol/vol1/.snapshot
/vol/vol0/            25052408    2124804   22927604          0  /vol/vol0/
/vol/vol0/.snapshot    6263100      74656    6188444          0  /vol/vol0/.snapshot
```

Disk Space Consumed by Read/Write Snapshot Copies

Even though read/write Snapshot files are initially linked to the existing blocks in the Snapshot copy, it is necessary for the NetApp storage system to allocate blocks equal to the entire size of the database files should it be completely overwritten and a copy of it created. There must be enough available disk space to accommodate the entire size of the original LUN while the read/write Snapshot copy is mounted. The space consumed by the read/write Snapshot copy is marked free disk space when it is dismounted using the `Disconnect Disk` command in SnapDrive. If the space is not available, consider using FlexClone to access Snapshot copies of databases.

Space Requirements Overview

Estimating storage requirements can be complicated, especially when a sizing project involves server consolidation. The basic areas of focus consist of several related objects:

1. System databases
   a. How large does `tempdb` grow throughout the day?
2. User databases
   a. How frequently will the database and log be backed up?
   b. How large is each log dump?
   c. How large will the databases grow over the next two to three years?
3. SnapInfo
   a. How many full backups will take place and be retained?
   b. How many log backups?
   c. How many stream-based backups of system databases?
4. Snapshot copy creation and retention
   a. What is the approximate delta between Snapshot copies?
   b. How many transaction log and Snapshot copies must be available?
   c. When and how often should an archive backup take place?

Work to establish a rough estimate for each of the four areas. The Consolidation Methodologies section of the Appendixes puts all guidelines into practice in a consolidation project; refer to this section for more detail.

5 Backing Up and Restoring Databases with SnapManager for SQL Server

How to back up and restore SQL Server databases is thoroughly discussed in the SnapManager for SQL Server (SMSQL) Installation and Administration Guide. Only areas that must be well understood are repeated here.

During SMSQL backup:

- Snapshot copies are created of all volumes used by the databases being backed up.
- System databases are backed up using conventional streaming-based backup.
- User databases placed in a LUN that also contains system databases are backed up using streaming-based backup.
- Backup of transaction logs always uses streaming-based backup to provide point-in-time restores.

The following steps happen when SMSQL restores a database using a LUN clone split restore:

1. The administrator issues a standard LUN restore operation using SnapDrive.
2. SnapDrive sends a snapshot-restore-file ZAPI command to the controller to execute.
3. Any LUN clone split running for the LUN being restored is aborted.
4. User I/O is paused.
5. A new LUN clone is created backed by the LUN in the Snapshot copy being restored from. This LUN is initially created without its space reservation to help avoid problems with insufficient free space.
6. The original LUN is deleted. The delete is processed piecemeal by the zombie delete mechanism in the background.

7. The new LUN clone is renamed to have the same name as the original LUN. Any host connections to the LUN are specially preserved.
8. The space reservation is applied to the new LUN clone. Now that the original LUN is deleted the new LUN clone should be able to regain the same reservation (since the restored LUN is the same size).
9. User I/O is resumed. At this point the LUN is fully restored from the host's perspective. The host OS and applications can begin doing reads and writes to the restored data.
10. A split of the LUN clone is started. The split runs in the background. This will be a space-efficient split. The space reservation for the LUN clone is dropped during the split and reapplied once the split is complete.
11. The SnapDrive LUN restore operation completes. All of this usually requires about 15 seconds.

5.1 Benefits

The benefits of the new LUN clone split restores are:

- The SnapDrive restore operation completes in about 15 seconds.
- The LUN remains online and connected to the host throughout the restore. The host does not have to rescan for the LUN after the SnapDrive restore operation completes.
- The LUN is fully restored from the host's perspective once the SnapDrive restore operation completes, in about 15 seconds.
- The LUN is available for host I/O (reads and writes) immediately after the SnapDrive restore operation completes, in about 15 seconds.
- The split of the LUN clone that was created during the restore runs in the background and does not interfere with the correctness of host I/O to the LUN. Writes done by the host OS and applications are preserved in the clone as you would expect. This is the standard way LUN clones operate.
- The LUN clone split is a new, space-efficient split that is usually 90% or more efficient, meaning it uses 10% or less of the fully allocated size of the LUN for the restore operation. The space-efficient LUN clone split is also significantly faster than an old-style block copy LUN clone split.
- You can determine the progress of the LUN clone split by asking for the LUN clone split status for the LUN. The status reports a percentage complete for the split.
- An administrator who accidentally starts the restore using the wrong Snapshot copy can easily issue a new restore operation using the correct Snapshot copy. Data ONTAP automatically aborts any LUN clone split that is running for the prior restore, creates a new LUN clone using the correct Snapshot copy, and automatically splits the LUN clone to complete the restore operation. Previously, the administrator had to wait until the previous SFSR restore was complete before restarting the SFSR restore with the correct Snapshot copy.
- Multiple LUN clone split restores can be in progress simultaneously.
- No new procedures or commands are needed to use this functionality. Administrators should issue LUN restore operations using SnapDrive as they normally would, and Data ONTAP handles the details internally.
- If the controller is halted while the LUN clone split is running, when the controller is rebooted the LUN clone split is restarted automatically and will then run to completion.

5.2 Backup Recommendations

SMSQL's full backup using Snapshot technology is quick and does not adversely influence database performance compared to traditional backup with SQL Server BACKUP statements. Transaction log backup is functionally the same using SMSQL as the transaction log backup done using BACKUP Log statements and is streamed out to the SnapInfo directory for safekeeping. The length of time required to back up a transaction log depends mainly on the active section of the transaction log that has to be extracted and copied to the dump directory. When planning the backup and restore process, the methodology or considerations are different from considerations when backup is done by Enterprise Manager or SQL Server Management Studio. The two main considerations are:

1. Recovery point objective (RPO)—to what point in time must the data be recovered?
2. Recovery time objective (RTO)—how long will it take to get the database back online and rolled forward or backward to the RPO?

Consider the following hypothetical customer scenario:

- **Customer:** ACME Corp
- **Database:** CommerceDB
- **Failure Type:** Database corruption or deletion
- **RTO:** It cannot take longer than 30 minutes to complete a restore
- **RPO:** Data should be brought back to a time no later than 15 minutes from failure

ACME's IT group has shown that a full backup every 30 minutes, with a transaction log backup every 15 minutes on the hour, makes it possible to restore the database in 20 minutes (without the verification process) to the minimum recovery point objective of 15 minutes. Backup every 30 minutes during a 24-hour time frame creates 48 Snapshot copies and as many as 98 transaction log archives in the volume containing SnapInfo. A NetApp volume can host up to a maximum of 255 Snapshot copies so the proposed configuration could retain over 5 days of online Snapshot data.

It is generally a good idea to categorize all your organization's systems that use SQL Server in terms of RTO and RPO requirements. Once these requirements have been established, SMSQL can be tailored to deliver the appropriate level of protection. Proper testing before deployment can also help take the guesswork out of determining what can be realistically achieved in production environments.

In situations in which business requirements are not entirely clear, refer to the following recommended default database backup frequency.

Best Practice: Minimum Backup Frequency
All backup and recovery processes should be created according to each environment's unique requirements. When in doubt, however, the general recommendation is to back up the database every hour and the transaction log every 30 minutes during production time and possibly less often during off-peak hours.

5.3 Archiving Backups

Best Practice: Always Archive or Mirror Snapshot Copies
NetApp strongly recommends archiving or mirroring backup sets as soon as possible. Disasters do occur, and if the storage device containing the databases and backup Snapshot images is adversely affected, it will not be possible to restore the databases from the primary storage system. Archive media might be tape, another FAS system, or a NearStore® storage-enabled device. The archive process is described in further detail in the SMSQL 2.1 Installation and Administration Guide.

6 Host and Storage System Installation and Configuration Best Practices

Several things must be done in preparing a NetApp storage system and SQL Server host to create a reliable system with optimal performance. Refer to the latest SnapDrive and SnapManager for SQL Server Administration Guides to make certain the proper licenses and options are enabled on the NetApp storage device.

6.1 NetApp Storage System Requirements

For all NetApp storage system requirements, refer to the NetApp Interoperability Matrix Tool.

6.2 Installing and Configuring SnapDrive 4.1 (or Higher)

Preparing for Installation

Prior to launching the installation of SnapDrive, use this checklist to help eliminate the potential for errors or delays during or after the installation.

- Resolve any hardware, cabling, or network issues or other errors.
- Make sure all of the necessary software and printed or electronic documentation (found on http://now.netapp.com) is organized and nearby before beginning.
- Configure DNS, hostname, and IP address–related services:
  - Verify that all related systems, including storage systems, servers, and clients, are configured to use an appropriately configured DNS server.
  - Manually add the NetApp storage systems' hostnames to DNS.
  - Enable, configure, and test RSH access on the storage systems for administrative access (for more information, see Chapter 2 of the Data ONTAP 7.x Storage Management Guide).
- License all of the necessary protocols and software on the storage system.
- Make sure the storage system's date and time are synchronized with the Active Directory® domain controllers. This is necessary for Kerberos authentication. A time difference greater than five minutes results in failed authentication attempts.
- Verify that all of the service packs and hot fixes are applied to the Microsoft Windows 2003 Server or Microsoft Windows 2008 Server and Microsoft SQL Server 2005 and 2008.

SnapDrive Installation Tips

Select the same account used by the Microsoft SQL Server services when selecting the service account for the SnapDrive and SnapManager for Microsoft SQL Server services.

When creating or configuring the properties for the domain service account, select the Password never expires checkbox. This protects the account and service from failing due to user password policies.

Note: It is important to make certain that the service account has the following permissions:

- Read and write or full control access to the locations in which LUNs will be created (likely if it is already a member of the administrators group)
- RSH access to the storage systems; for more information on configuring RSH, see the Data ONTAP System Administration Guide.

Creating and Managing LUNs with the SnapDrive GUI

Once installed, SnapDrive can be used to create LUNs on NetApp storage systems for use by Windows 2005, 2008, 2008R2, and 2012 hosts.

Best Practice: Tips When Creating LUNs

- When specifying a UNC path to a share of a volume when creating a LUN, use IP addresses instead of hostnames. This is particularly important with iSCSI, as host-to-IP name resolution issues can interfere with the locating and mounting of iSCSI LUNs during the boot process.
- Use SnapDrive to create LUNs for use with Windows to avoid complexity.
- Calculate disk space requirements to accommodate data growth, Snapshot copies, and space reservations.
- Be sure to leave automatic Snapshot scheduling off as configured by SnapDrive.

Snapshot Writable LUNs or FlexClone Volumes

Snapshot backups of SQL Server databases are read-only, point-in-time images. This protects the integrity of the Snapshot backups. In certain situations, SMSQL restores a LUN in a Snapshot copy for temporary read/write access (restored to an alternative location using a writable Snapshot copy[1]). Writable Snapshot copies are used during a verification process, when a DBA restores a database to an alternative location for recovering from operational mistakes, or in some cases when external archive backups take place. A restored writable Snapshot copy is for short-term use only.

Best Practice: Snapshot Writable LUNs

Avoid creating Snapshot copies on volumes with active Snapshot-writable[2] LUNs.

NetApp does not recommend creating a Snapshot copy of a volume that contains an active Snapshot copy because this creates a “busy Snapshot”[3] issue. Avoid scheduling SMSQL Snapshot copies when:

- Database verifications are running
- Archiving a Snapshot-backed LUN to tape or other media

6.3 Storage Mirroring and Disaster Recovery

NetApp SnapMirror technology mirrors data to one or more NetApp storage volumes over a local area network (LAN) or wide area network (WAN) to another NetApp storage system. Once source and destination relationships are established, a SnapMirror baseline transfer initializes the mirror to create a replica of the source on the destination. SnapMirror maintains the synchronization of the replica through efficient, incremental updates that use bandwidth intelligently to distribute deltas across a WAN.

Mirroring Frequency

The frequency of SnapMirror update events (in most environments) is most often determined by the frequency of SMSQL backups. SnapDrive triggers SnapMirror updates upon completion of a SnapManager backup procedure. A supplemental type of Snapshot copy optimized for use with

[1] A writable Snapshot copy or Snapshot-writable LUN is a LUN that has been restored from a Snapshot copy and enabled for writes. A file with a .rws extension holds all writes to the Snapshot copy.
[2] There are two ways to determine whether you have a busy Snapshot copy: (1) View your Snapshot copies in FilerView. (2) Use the following Data ONTAP command to list the busy Snapshot copies:
```
snap list usage VolumeName BusySnapshotName
```
[3] A Snapshot copy is in a busy state if there are any LUNs backed by data in that Snapshot copy. The Snapshot copy contains data that is used by the LUN. These LUNs can exist either in the active file system or in another Snapshot copy. For more information about how a Snapshot copy becomes busy, see the Block Access Management Guide for your version of Data ONTAP.

SnapMirror, called “rolling Snapshots,” can be used to augment standard SQL Server backups. Rolling Snapshot copies are controlled outside of SMSQL by using the scripting executable `sdcli.exe` for SnapDrive. Be mindful of the following rolling Snapshot technology best practices:

Best Practices: When Using Supplemental Rolling Snapshot Copies
• Be sure to thoroughly read the SMSQL Installation and Configuration Guide section “Minimizing your exposure to loss of data.”
• Be sure to configure rolling Snapshot copies to occur only between SMSQL backup and SnapMirror processes. This prevents overlapping Snapshot schedules.

Key SnapMirror Concepts and Tips

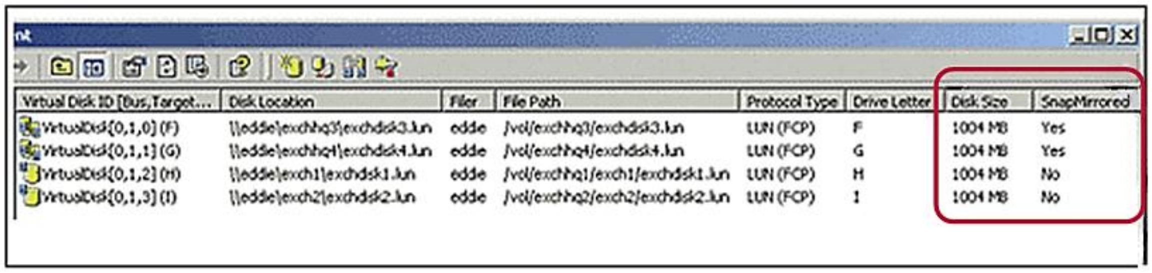
- SnapMirror relationships are based on sources and destinations.
- A destination updates its mirrored copy or a replica of its source by "pulling" data across a LAN or WAN when an update event is triggered by the source. The pull events are triggered and controlled by SnapDrive.
- Consistent with the pulling nature of SnapMirror, relationships are defined by specifying sources from which the destination NetApp storage systems synchronize their replicas.
- Destination systems are identified on source systems by assigning destination privileges by using the snapmirror.access protocol access option or by inclusion in the snapmirror `.allow` file.

As discussed earlier, SnapManager is integrated with SnapDrive, which interfaces directly with a NetApp storage system by using either iSCSI or FCP disk-access protocols. SnapManager relies on SnapDrive for disk management, controlling Snapshot copies, and SnapMirror transfers.

Checklist for Configuring SnapManager and SnapMirror

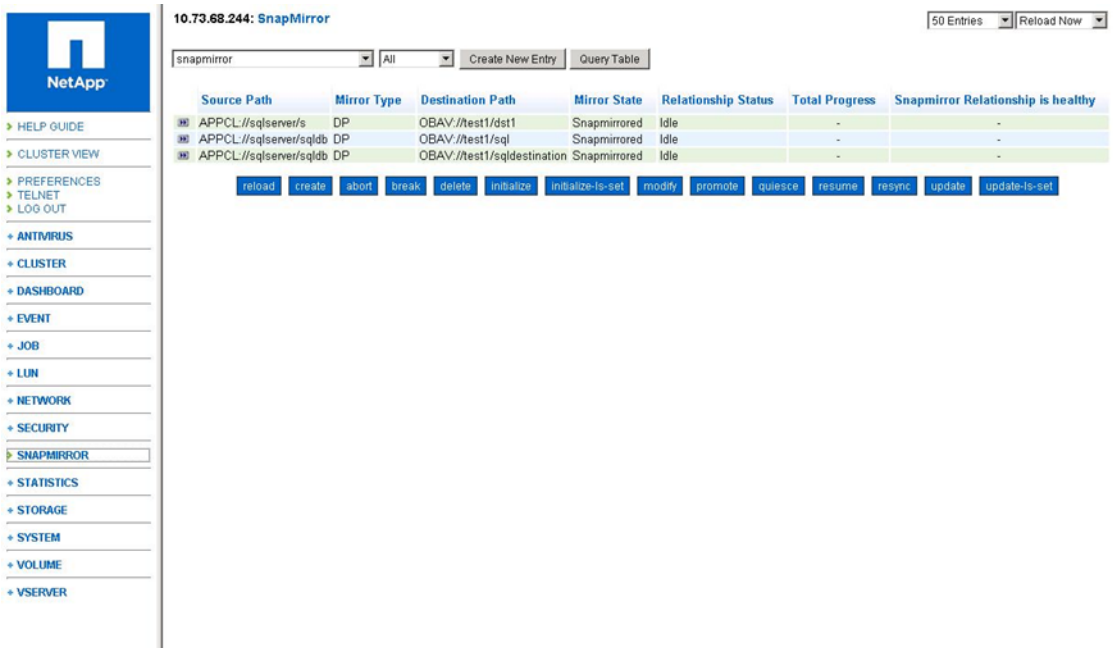
1. Install (using CIFS setup) both storage systems into the same Windows domain as SQL Server.
2. Configure SQL Server, SnapDrive, and SnapManager services to use the same logon service account.
3. Make sure the SnapDrive service account has the "Log on as a service" rights in the Windows 2000\2003 operating system (normally occurs during installation).
4. Verify that RSH commands work between the SQL Servers and both storage devices using the specified service account.
5. License and enable FCP or iSCSI on both storage devices.
6. License and enable SnapMirror on both storage devices.
7. Establish SnapMirror relationships.
8. Make sure the storage device's SnapMirror schedule is turned off by assigning the “- - - -“value (four dashes separated by spaces) in the custom schedule field.
9. Initialize the mirror configurations.
10. SnapMirror updates to the specified destinations occur after the completion of every SMSQL backup.
11. Use SnapDrive and FilerView to verify the successful completion and state of the configured mirrors.

Figure 2) SnapMirror status in SnapDrive.



Virtual Disk ID [Bus,Target...	Disk Location	Filer	File Path	Protocol Type	Drive Letter	Disk Size	SnapMirrored
VirtualDisk{0,1,0} (F)	\\eddie\exchhq3\exchdisk3.lun	eddie	/vol/exchhq3/exchdisk3.lun	LUN (FCP)	F	1004 MB	Yes
VirtualDisk{0,1,1} (G)	\\eddie\exchhq4\exchdisk4.lun	eddie	/vol/exchhq4/exchdisk4.lun	LUN (FCP)	G	1004 MB	Yes
VirtualDisk{0,1,2} (H)	\\eddie\exch1\exchdisk1.lun	eddie	/vol/exchhq1/exch1/exchdisk1.lun	LUN (FCP)	H	1004 MB	No
VirtualDisk{0,1,3} (I)	\\eddie\exch2\exchdisk2.lun	eddie	/vol/exchhq2/exch2/exchdisk2.lun	LUN (FCP)	I	1004 MB	No

Figure 3) SnapMirror status in FilerView.



10.73.68.244: SnapMirror

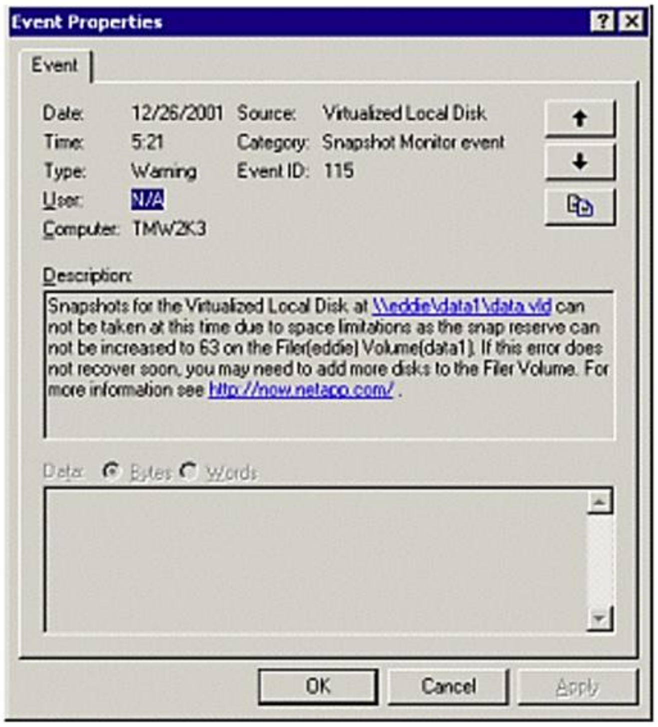
Source Path	Mirror Type	Destination Path	Mirror State	Relationship Status	Total Progress	Snapmirror Relationship is healthy
APPCL://sqlserver/s	DP	OBAV://test1/dst1	Snapmirrored	Idle	-	-
APPCL://sqlserver/sqldb	DP	OBAV://test1/sql	Snapmirrored	Idle	-	-
APPCL://sqlserver/sqldb	DP	OBAV://test1/sqldestination	Snapmirrored	Idle	-	-

7 Best Practices for Ongoing Management and Monitoring

7.1 Snapshot Reserve Using Monitoring

The task of monitoring the Snapshot reserve is automatically configured at the time of LUN creation. Simply monitor the application event log for warning events generated in the SnapDrive source and Snapshot Monitor event categories. Figure 4 demonstrates that, due to Snapshot copy consumption, the reserve must be expanded to 63%, and there is not enough disk space available to do so. To rectify this situation, simply grow the FlexVol volume, remove some of the oldest SMSQL Snapshot copies, or add new disk resources to the aggregate.

Figure 4) Space alert.



Date: 12/26/2001 Source: Virtualized Local Disk
Time: 5:21 Category: Snapshot Monitor event
Type: Warning Event ID: 115
User: N/A
Computer: TMW2K3

Description:
Snapshots for the Virtualized Local Disk at \\eddie\data1\data.vld can not be taken at this time due to space limitations as the snap reserve can not be increased to 63 on the Filer(eddie) Volume(data1). If this error does not recover soon, you may need to add more disks to the Filer Volume. For more information see http://now.netapp.com/ .

Data: Bytes Words

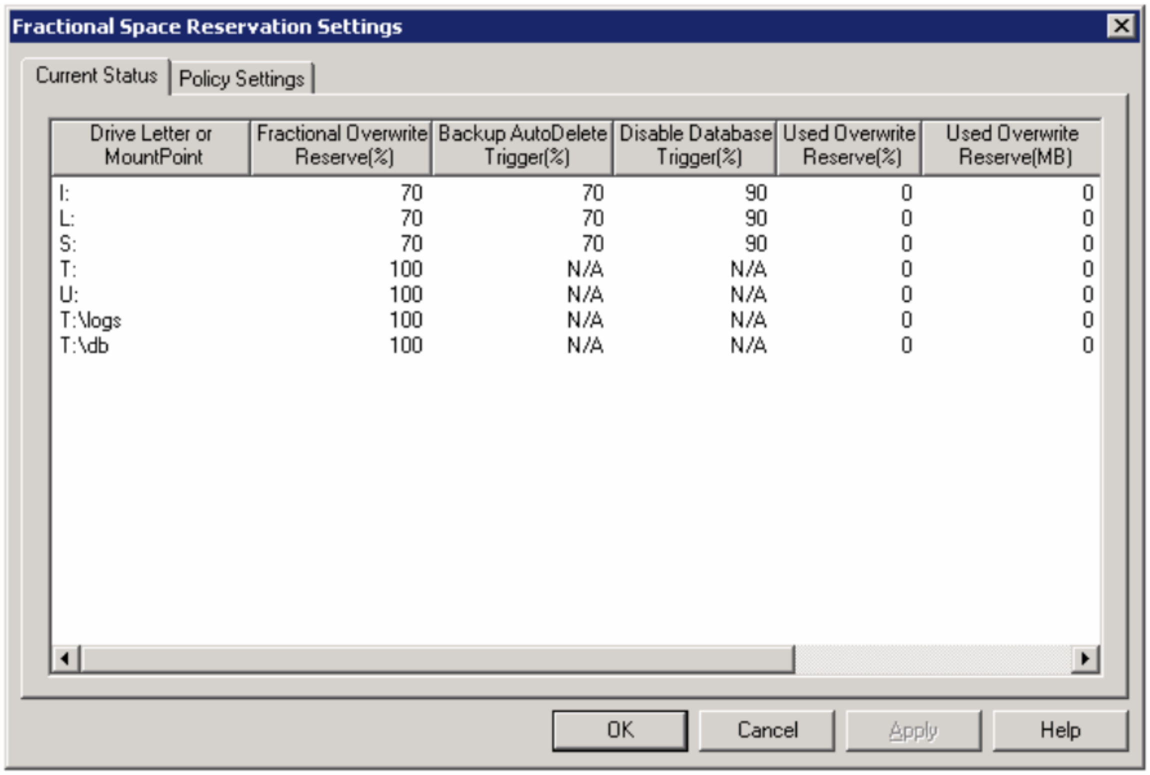
7.2 Disk Space Usage Monitoring

The amount of disk space used by each database managed by SQL Server 2000 and 2005 should be monitored to make sure that the logical drives (LUNs) do not run out of space. The default settings for SQL Server databases will automatically extend the preallocated file space when there is not enough free file space to extend a table (if the database has been defined with auto extend on). SQL Server stops processing transactions when it cannot extend its file space.

Two things must be done to monitor free space. The first is to free preallocated space that is available for table extends. If Perfmon or a similar monitoring tool like Microsoft Operations Manager (MOM) is already being used for monitoring system performance, it is easy to add counters and rules to monitor database space. A database's properties show its current size and how much space is free. When free space gets below or close to 64KB, the preallocated file space is extended. The second thing is to monitor the free LUN space, which can be done with Windows Explorer. You may also use the system stored procedure sp_databases to show the database files and sizes.

7.3 Fractional Space Usage Monitoring

In the menu, click Options->Fractional Space Reservation Settings to access the Current Status dialog.



Drive Letter or MountPoint	Fractional Overwrite Reserve(%)	Backup AutoDelete Trigger(%)	Disable Database Trigger(%)	Used Overwrite Reserve(%)	Used Overwrite Reserve(MB)
I:	70	70	90	0	0
L:	70	70	90	0	0
S:	70	70	90	0	0
T:	100	N/A	N/A	0	0
U:	100	N/A	N/A	0	0
T:\logs	100	N/A	N/A	0	0
T:\db	100	N/A	N/A	0	0

In the Fractional Space Reservation Settings dialog box, use the Current Status tab to view the current space consumption in the storage system volumes that contain LUNs that store SQL data or SnapInfo directories. You see the following information:

Drive Letter or Mount Point

The drive letter or NTFS mount point on which the LUN is mounted.

Fractional Overwrite Reserve (%)

The amount of space reserved for overwrites on the storage system volume that contains the LUN. It is expressed as a percentage of the total size of all space-reserved LUNs in the volume.

Backup AutoDelete Trigger (%)

It is the setting for the storage system volume that contains the LUN. It is expressed as a percentage of overwrite reserve utilization that triggers automatic deletion of SQL backup sets.

Disable Database Trigger (%)

It is the setting for the storage system volume that contains the LUN. It is expressed as a percentage of overwrite reserve utilization that triggers automatic disabling of SQL databases.

Used Overwrite Reserve (%)

The amount of overwrite reserve allocated for the storage system volume that contains the LUN. It can be expressed in one of two ways:

- As a percentage of the total size of all space-reserved LUNs in the volume
- In megabytes for the total size of all space-reserved LUNs in the volume

Available Reserve (MB)

This is the amount of overwrite reserve available for the storage system volume that contains the LUN.

7.4 NetApp Storage System Event Monitoring

Monitor the storage system's event logs to promote proper operation. Issues might be discovered in the event logs that require administrative action. One such action could be to replace a disk in a volume if spare disks are not available.

This task can be completed by using the FilerView® utility built into Data ONTAP. Start the utility by pointing a Web browser to http://filername/na_admin. Next, click the FilerView link. FilerView starts and asks for the credentials of a storage system administrator. Once logged on to FilerView, click the Filer menu option on the left side of the screen. Then choose the Syslog Messages menu option. Review the log on the right side of the screen for any issues that may require administrative action. For more information on FilerView, refer to the Data ONTAP System Administration Guide.

7.5 Terminal Service or Remote Desktop

NetApp does not recommend using Microsoft Terminal Server for Windows as a way of administering SnapDrive or SnapManager for SQL Server. When you create LUNs from a terminal server session, the drives can look like they were not created or have been disconnected when in fact they are functioning properly.

Best Practice: Use Alternative Means for Remote Management
NetApp recommends avoiding Terminal Server for server management when possible. When using Terminal Server with Windows 2003/2008/2008R2 (not Windows 2000), you may use a remote desktop to connect if you use a command line parameter/console, as shown: `%SystemRoot%\System32\mstsc.exe /console.`

8 SnapManager for SQL Server Simplifies SQL Server Database Management

By providing the rich backup and restore features of SMSQL, performance and utilization optimizations of FlexVol, and robust data protection capabilities of SnapMirror, NetApp provides a complete data management solution for SQL Server environments. The recommendations and examples presented in this document will help organizations get the most out of NetApp storage and software for SQL Server deployments. For more information about any of the solutions or products covered in this report, contact NetApp.

SMSQL and its previous versions are aimed at supporting the following major functions with respect to SQL Server databases:

1. Space-efficient backup of SQL Server databases
2. Quick restore of SQL Server databases from these backups
3. Migration of SQL Server database files to NetApp storage with minimal effort and downtime
4. Verification of backup sets
5. Mirroring of backup sets to secondary locations for DR planning

In addition, SMSQL supports some very specialized functions including database cloning, moving backup sets to secondary storage, and the advanced backup set verification option. These are discussed later in this document.

8.1 SMSQL Architecture: Components Used by SnapManager for SQL Server

SnapManager for SQL Server introduces a significant change in architecture compared to the previous versions. The SMSQL software consists of two major parts:

- The client-side module that is responsible for displaying the GUI and interacting with the user.
- The server-side module that houses all the services and DLLs responsible for executing the different functionalities of SMSQL. These services are housed in a single EXE named SnapMgrService.EXE and run as a Windows service named “SnapManager Service.”

This architecture allows SMSQL to offer centralized management of multiple SQL Server hosts in an organization’s infrastructure.

The Client and Server modules communicate among themselves using the Windows Communication Foundation (WCF) Web services. This replaces the traditional use of RPC for intercomponent communication. Furthermore, the WCF uses pure TCP communication for managing communications between the client and the server and the TCP traffic flows through Port 808 (by default). Note that this port is the default port prescribed by Microsoft to facilitate TCP/IP traffic for WCF-based components. Thus, the SMSQL installer always checks for the .NET version and makes sure that it is version 3.5 SP1.

TIP: Using the SMSQL GUI, it is possible to monitor SQL Server instances in a heterogeneous Windows environment. This means that it is possible to monitor all SQL Server instances running on various versions of Windows and on different platforms (x86, x64, and IA-64) from one single SMSQL GUI running on any platform and Windows OS. This implies that one can set up a true central management console to monitor a large SQL Server farm. The reason for this capability is the use of WCF as a means of communication between the client and server components of SMSQL.

8.2 SMSQL Concepts

SnapInfo Directory

The SnapInfo directory is the main repository of the SnapManager for SQL Server software. All the metadata related to the SMSQL instance installed on a host is kept in SnapInfo along with all transaction log backups taken through SMSQL. Hence, the main data items that SnapInfo contains are:

- Metadata related to SMSQL configuration and backup sets
- Streamed full backup of system databases
- Streamed full backup of user-defined databases that share a virtual disk with a system database
- All backed-up transaction logs (through SMSQL)

For a SQL Server host, there can be one or more SnapInfo directories. This is because SMSQL allows one or more SQL Server instances on a host to share a SnapInfo directory as well as each instance to have its own SnapInfo directory. This can be configured at any time by running the Configuration Wizard.

Note: For large production systems, either of two approaches can be followed to allocate SnapInfo directories:

- Give each highly active instance that sees a lot of backups and transactional activity and has greater criticality to the business operations a SnapInfo LUN and preferably volume of its own.
- For SQL Server instances in a production host that do not see too much activity and house noncritical databases, divide these instances into groups of two or three and give them a SnapInfo LUN of their own. Note that you should not group together instances that have databases with large TLOGs.

TIP: It is important to note that, as part of any backup process, full backup or transactional log backup, the SnapInfo directory is Snapshot copied.

Important Concepts for SQL Server Backups

SMSQL primarily performs complete backup management for SQL Server databases. Thus a basic understanding of the SQL Server backup/restore model is very important to comprehending how SMSQL functions.

We begin with the different recovery models available in SQL Server 2000, 2005, and 2008. Recovery models dictate the level to which the transactions in databases are logged into the TLOG. Hence, the recovery models govern the extent to which a database can be recovered. Table 3 compares the three types of recovery models in SQL Server.

Table 3) Recovery models in SQL Server.

Recovery Model	Extent of Transaction Logging	Recoverability	Loss of Data
Simple	Minimal TLOG is truncated at each checkpoint. TLOG cannot be backed up.	Up until the last full backup	Maximum
Bulk-logged	Moderate Does not log bulk operations such as index creations, BULK COPY, and so on.	End of last known TLOG backup	Moderate
Full	Maximal All transactions are logged.	Point in time	Minimum

Note: The recovery model of a database directly affects the amount of space consumed by each TLOG backup. Note that the size of a TLOG backup equals the used portion of the TLOG and not the actual TLOG file size. The used portion's size may be determined by using the command **`DBCC SQLPERF('LOGSPACE')`**.

Sizing for SMSQL Datastores

One of the most common questions while installing SMSQL is how to size volumes that are used by SMSQL to store data. The primary data that SMSQL generates and stores in its own datastores is the data related to the SnapInfo folders. As previously noted in the SnapInfo Directory section, the main data items that SnapInfo contains are:

- Metadata related to SMSQL configuration and backup sets
- Streamed full backup of system databases
- Streamed full backup of user-defined databases that share a virtual disk with a system database
- All backed-up transaction logs (through SMSQL)

Of these components, the following items are easy to size and thus give you a good idea of the amount of space that would be consumed:

- Metadata—Metadata per backup set is less than 10KB.
- System databases—Total size of all system databases (that can be backed up with SMSQL) except tempdb, which are MASTER, MODEL, and MSDB = 15MB (at the maximum).

Note: The only database that is subject to a larger change rate is the MSDB, since it stores job histories, backup histories, SQL Server jobs (including replication related), and so on.

For the remaining two items, the chances of having streamed backups of user databases is low since this happens only when the user databases are on the same LUN/volume as the system databases.

Note: It is preferable to avoid a configuration in which the system databases share the same volume as the user databases. This is because SMSQL cannot create Snapshot copies of volumes that have system databases on them, because I/O cannot be suspended for system databases.

Therefore, the major consumers of space are the transaction log backups of different databases. The main difficulty in determining the size of these backups arises from the following factors:

- The sizes of transaction log files vary and are subject to autogrowth.
- The actual amount of log file space used varies greatly across all the databases in a SQL Server instance.

These two factors make the sizing exercise for SnapInfo very cumbersome. In order to overcome these factors to some extent, NetApp recommends the following approach:

Step 1: For each SnapInfo folder:

a. Step 2: Based on recommendations in section 2.2, determine the number of SQL Server instances that will share the current SnapInfo folder.

Step 2: For each SQL Server instance:

a. Run the script listed in the Load Generation Scripts section of the Appendixes. The script generates the values:
b. Base Value: Sum of the current log file sizes of all the databases in the instance. Let this value be called BASE_VAL.

Step 3: Now determine the maximum number of TLOG backups per database that will be kept in the SnapInfo folder. Let this be N.

Step 4: Therefore, the predicted LUN size = BASE_VAL × N. Note that the LUN size is a cumulative total for all the SQL Server instances.

Step 5: Remember the LUN size for this iteration.

Step 6: Once steps 2–5 are run for each concerned SQL Server instance, we get the target LUN size needed for the SnapInfo folder.

Step 7: Note that each time SMSQL completes a Full or TLOG backup, it makes a Snapshot copy of the SnapInfo volume. So now we must determine the number of Snapshot copies to be kept online. Let this value be M.

Step 8: From the list of LUN sizes that were obtained from steps 4–7, choose the highest value. You can assume that the Snapshot copy size will at most be this highest value, so name it HIGH_VAL.

Step 9: Therefore the volume size for the SnapInfo folder = LUN size obtained from step 8 + (HIGH_VAL × M).

For these steps NetApp assumes that the number of databases in the instance remains fairly constant.

Hence, we now have an approximation procedure to determine the size of the SnapInfo volume that takes into account:

- Average TLOG sizes across all databases
- Number of TLOG backups to be kept online
- Number of Snapshot copies of the SnapInfo volume to be kept online

TIP: By creating a schedule for regular TLOG backups, one can keep the max size of the TLOG file bounded to the value existing at the start of the preceding steps. In BULK-LOG and FULL recovery modes, inactive portions of the TLOG file are made available to SQL Server for further transaction logging.

TIP: By default, SMSQL does not impose Fractional Space Reservation policies on the SnapInfo folder. However, by following the sizing exercise mentioned previously, one can impose a lower value than the default.

How Does Backup Work

SMSQL is based on the Virtual Devices Interface™ (VDI) technology from Microsoft that allows third-party utilities to take consistent backups of SQL Server databases.

For more information on VDI, refer to the following link:

www.microsoft.com/downloads/thankyou.aspx?familyId=416f8a51-65a3-4e8e-a4c8-adfe15e850fc&displayLang=en

When a backup operation is triggered, SMSQL:

1. Checks the SnapManager license.
2. Renames the most recent SnapInfo directory (if necessary).
3. Renames the most recent Snapshot copy (if necessary).
4. Creates a new directory in the SnapInfo directory for this backup. At the same time it also creates a .SML file that contains the metadata for this backup.
5. SMSQL enumerates all the databases that reside on the current NetApp volume even if they are in different LUNs.
6. SMSQL then initiates VDI backup using the `BACKUP DATABASE…WITH SNAPSHOT` command for each of the databases sharing the volume.
7. SQL Server receives the `BACKUP` commands and starts suspending the I/O activity on all databases that share the same NetApp volumes.
8. SQL Server returns an acknowledgement to SMSQL, confirming that I/O is suspended.
9. SMSQL now creates Snapshot copies of each of the involved volumes serially.
10. SMSQL returns acknowledgement to SQL Server about Snapshot copy completion.
11. SQL Server unfreezes the I/O on the databases that were frozen in step 7.
12. SMSQL proceeds with further actions such as TLOG backup, SnapMirror update, and so on.

TIP: There are two important points to note in the preceding steps:

- I/O for all databases in a volume is frozen, regardless of which database is being backed up.
- If a database uses multiple volumes, then these are copied by Snapshot one by one. However, if the database uses multiple LUNs on the same volume, then all of these LUNs are copied by Snapshot together since Snapshot copies are volume based.

There are several considerations to keep in mind regarding the SMSQL backup process:

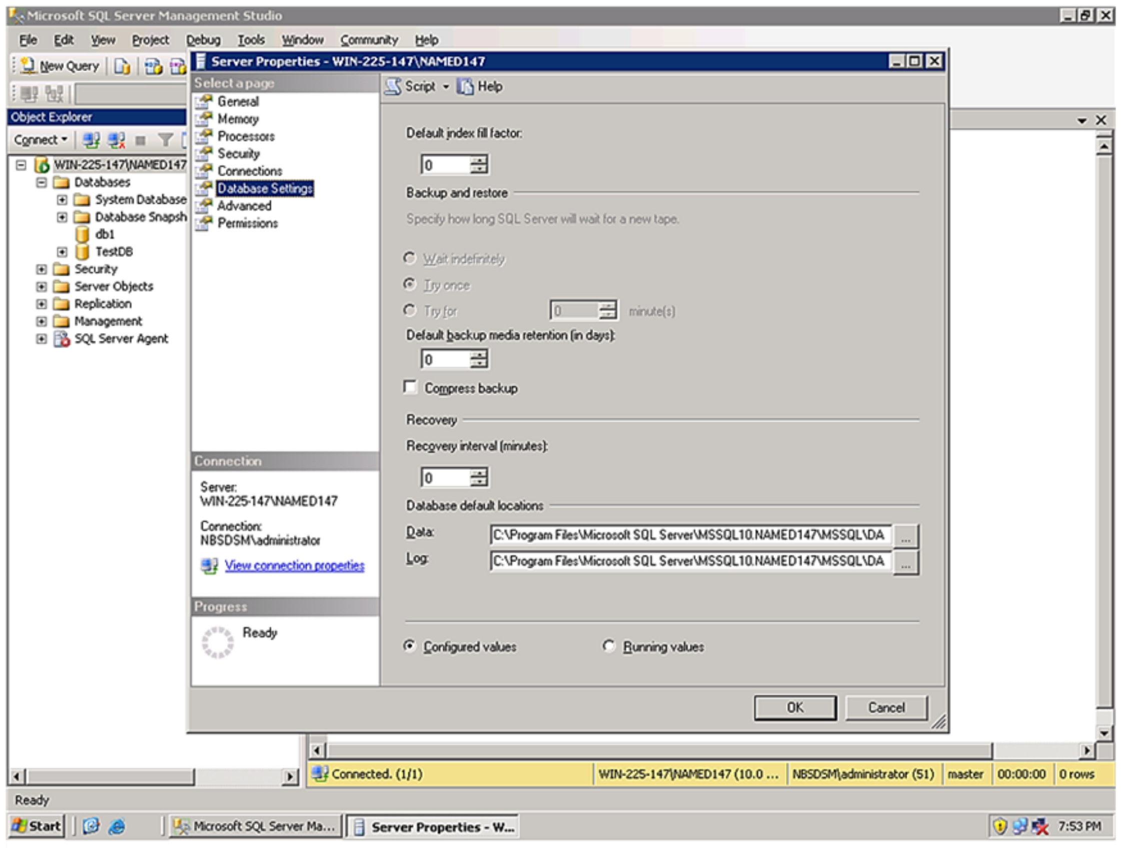
1. Microsoft recommends that databases not be frozen for more than 10 seconds as part of the Snapshot copy process. Check the backup logs at `C:\Program Files\NetApp\SnapManager` for `SQL Server\Report\Backup [Instance Name]` to determine whether the freeze window exceeded 10 seconds. Note that there are several factors that affect the time needed to complete the Snapshot copy operation:
   - Size of the volumes on which the database LUNs reside and the rate of change in them. Note that SMSQL Snapshot copies volumes serially, so if a database is spread across a large number of volumes, the copying takes more time.
   - Number of databases sharing the same volume, even though the databases may be in different LUNs on the same volume. This is because NetApp Snapshot copying is volume wide and so also is the Snapshot copy granularity from a VDI standpoint.
   - How busy the storage controller is in terms of CPU usage, IOPS bandwidth, and so on.
   - Space left on the volume. The more full the volume gets (especially after 50–60% fullness), the slower the Snapshot copies become.

  - Network delays. These could cause delays in the acknowledgement from the storage controller reaching SMSQL and then SQL Server. Until the acknowledgement occurs, SQL Server will keep the databases frozen.

2. It is not advisable to put large databases on the same volume even if they are on separate LUNs inside the volume. This might have implications for the time Snapshot takes to copy the volume.
3. Currently there is a limitation of 35 databases that may share a single volume. The reason for this is that in order to manage the freezing and unfreezing of I/O for each database, SQL Server has to use four to five worker threads per database. This may lead to a situation in which SQL Server may run out of worker threads and fail in an unexplained manner. Thus, Microsoft strongly recommends that no more than 35 databases share a storage volume. Find more information on this issue at https://kb.netapp.com/support/index?page=content&id=3011203.
4. SMSQL supports both synchronous and asynchronous mirroring, which means that one can mirror the database volumes in any manner without any effect on SMSQL backups. If the database volumes are synchronously mirrored, then there is no need to select the Update SnapMirror option for the backup plan. Refer to the section “How SnapManager Uses SnapMirror” on page 270 of the SnapManager for SQL Server (SMSQL) Installation and Administration Guide.
5. Data ONTAP has a limitation of 255 Snapshot copies per volume, so one must be careful about how many Snapshot copies are kept online. SMSQL is integrated with NetApp SnapVault® technology through Protection Manager datasets. For this reason, older Snapshot copies should be archived according to their lifecycle and restore needs. Refer to section 5 of this report for details.

Compression of Transaction Log Backup

SMSQL supports compressed TLOG backups for SQL Server 2008 databases. The TLOG backup compression comes into play only if it has been set at the SQL Server level by the DBA. SMSQL for its part does not use the `WITH COMPRESSION` qualifier with the TLOG backup statement. It may be enabled by checking the Compress Backup option in the Server Properties dialog box.



Server Properties - WIN-225-147\NAMED147

Default index fill factor: 0

Backup and restore

Specify how long SQL Server will wait for a new tape.

- Wait indefinitely
- [✓ Try once]
- Try for 0 minute(s)

Default backup media retention (in days): 0

☐ Compress backup

Recovery

Recovery interval (minutes): 0

Database default locations

Data: C:\Program Files\Microsoft SQL Server\MSSQL10.NAMED147\MSSQL\DA

Log: C:\Program Files\Microsoft SQL Server\MSSQL10.NAMED147\MSSQL\DA

[✓ Configured values] Running values

Connection

Server: WIN-225-147\NAMED147

Connection: NBSDSM\administrator

View connection properties

Progress

Ready

Connected. (1/1) | WIN-225-147\NAMED147 (10.0 ... | NBSDSM\administrator (51) | master | 00:00:00 | 0 rows

The TLOG backup compression feature simply compresses the backup file that is generated as a result of the `BACKUP LOG` statement. In the case of SMSQL, if the option is set, then the .TRB file that is saved to the SnapInfo folder as part of the TLOG backup is appreciably smaller than a regular TLOG backup. Since the end result is just a smaller file on the SnapInfo LUN, there is no effect on the way Snapshot-based backups work in SMSQL.

Protecting Your Databases

The central piece for protecting SQL Server databases from catastrophic disasters at a data center revolves around the ability of SMSQL to mirror backups to remote locations. Refer to the section “How SnapManager Uses SnapMirror” on page 270 of the SnapManager for SQL Server (SMSQL) Installation and Administration Guide for more information on how to update SnapMirror relationships from within SMSQL backup plans.

In addition, SMSQL offers the ability to verify backups present on SnapMirror destination volumes. This feature is discussed in more detail in section 9.1 of this report. Furthermore, TR-3604: NetApp Disaster Recovery Solution for Microsoft SQL Server discusses various solutions for planning DR for SQL Server user databases based on different RTO/RPO targets.

Nonetheless, some key concepts must be kept in mind while planning how database volumes should be replicated to the DR site using SnapMirror.

- **CHECKPOINT.** Apart from forcing dirty database pages onto disk, `CHECKPOINT` also alters the minimum recovery LSN for a database. The minimum recovery LSN, or minLSN, is the LSN from which SQL Server starts scanning a database’s transaction LOG to perform a database recovery.

- **Database consistency.** Note that taking NetApp Snapshot copies outside of SMSQL for database volumes does not guarantee application-consistent copies. Hence, any Snapshot copies taken for database volumes must necessarily be initiated by SMSQL.
- **Network traffic generated by replication through SnapMirror updates.** Note that SMSQL supports database volumes being both synchronously and asynchronously mirrored by Snapmirror. However, if the underlying database volumes are synchronously mirrored, then there is no need to check the Update SnapMirror option for the backup plan.

It is important to note that the following SnapMirror configurations do not work:

- Database volume asynchronously mirrored by SnapMirror and log volume synchronously mirrored by SnapMirror.
- Database and log volumes asynchronously mirrored by SnapMirror with SnapMirror updates being done outside SMSQL. Note that even if the updates are scheduled to run at the same time, this arrangement does not work.
- Database volume synchronously mirrored by SnapMirror and TLOG volume asynchronously mirrored by SnapMirror.

These SnapMirror configurations do not work because of the way database checkpointing and database recovery work. In these three scenarios there is a mismatch between the last LSN that wrote to the database file before it got replicated and the minLSN. Specifically, the minLSN in most cases is later in time than the last LSN that wrote to the database. This causes a consistency issue during database recovery and therefore the recovery of the database from the replicated volumes fails.

Performing Table-Level Restores with SMSQL

Restoring particular tables from within backup sets is a common requirement in many database environments. In the case of SQL Server, this requirement becomes more challenging because of the lack of support for table-level restores from within SQL Server 2000 or 2005. However, SMSQL provides an effective solution in this space with the use of the database cloning feature, which is discussed in more detail in the next section.

The following steps can be used to perform table-level restores from a database backup that is present online:

1. Start the Clone Wizard from the SMSQL GUI.
2. Choose Clone Databases from the Existing Backup Sets option.
3. Follow the steps in the wizard and verify that the database clone is created on the same SQL Server host (preferably) as the target database. Determine that the name of the clone is different from the name of the original database.

   TIP: Depending on how much later in time the database must be recovered to, an adequate number of TLOG backups (if they exist) may be chosen during the closing operation.
4. Once the clone of the database is ready and attached to a SQL Server instance, use `BCP`, `SELECT INTO`, SSIS packages, and so on to import the table into the original instance.
5. Once the table is recovered in the original instance, run the cloning wizard again, choose the Delete Clones option, and delete the clone that was created in step 3.

   TIP: It is advisable to use the Delete Clone option since it performs a complete cleanup of the clone, including LUNs that were mounted at the host.

Advantages of this method include:

- No additional space is needed since we will not be writing anything to the clone.
- Cloning of databases in SMSQL is done either through FlexClone technology or read-write Snapshot copies, which do not call for additional space overhead unless that clone database is written to.

- Cloning is far quicker and less disruptive compared to a database restore.

Database Cloning

One of the most important features that has been introduced in SMSQL is the ability to create clones of a database. SMSQL supports the following operations with respect to database clones:

- **Clone Creation.** The supported sources are:
  - Active databases
  - Database backups
- **Clone Deletion.** This includes cleaning up all clone-related resources including host-side LUN mappings. Autodeletion of clones is added as a new feature for the SMSQL 5.2 release.
  - When creating a clone database, the DBA has an option to delete the clone database after some period of time.
  - Clones with autodeletion enabled can be deleted automatically by the scheduled clone deletion job. When this is happening, the associated schedule job for clone resync is removed automatically.
  - The time period can be configured as minutes, hours, or days. The default period is 1 day.
  - The status of the delete operation is logged in the Windows Event Log. E-mail notification is based on the current e-mail notification settings.
  - If there are user connections while deleting the clone, the clone deletion will fail.
- **Clone Refresh**.
  - An option to refresh the cloned database is given when the administrator creates a clone database either based on an existing live database or based on a backup that is already created. So the cloned database can be synchronized with the live database.
  - The synchronization interval with the live database can be configured as a number of minutes, hours, or days. The default is every 12 hours.
  - The clone refresh/resync can be scheduled using the Windows task scheduler or the SQL Server Agent.
  - Clone refresh can be based on the SnapMirror destination volume. The limitation of clone on destination VMDK is also applied to clone refresh. If the database resides on VMDKs, it cannot be cloned from the destination volume to the local SQL Server.
  - When the live database is used for synchronization, a new backup will be created before the clone operation starts.

SnapManager contains a Clone Wizard that provides a convenient interface for performing the following cloning operations:

- Clone a database from an existing backup set
- Clone active production databases
- Clone a database on a SnapMirror destination volume
- Delete cloned databases

Note: It is desirable to have FlexClone licensed on the storage system where the backup sets or the active database is based. However, this is not a compulsory prerequisite. In case FlexClone is not licensed, the LUN is completely backed up by the Snapshot copy.

TIP: If you plan to use a database clone for a longer duration, NetApp recommends that FlexClone be licensed on the applicable storage systems so, if necessary, the FlexClone volume can be split and made into a LUN of its own.

TIP: Note that cloning is a feature that does not depend on the SQL Server version since the underlying mechanism is the same. The only caution that must be exercised is that clones of databases or backup sets must be created on similar SQL Server versions as the original database's SQL Server version.

In the following subsections we look at some of the internals of the cloning operations and discuss some best practices for database clones and how to use them optimally.

To understand how database cloning can be used and to obtain the step-by-step process for creating a database clone, refer to Chapter 11 of the SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide. In this section we examine some of the internals of how database clones are created from within SQL Server.

1. There are two scenarios for creating a database clone:
   a. From online backup sets of the database
   b. From the active database that is present online
2. In the case of creating a clone of an active database, SMSQL first creates a FULL backup of the database. The rest of the steps are similar.
3. SMSQL begins by contacting the SDW of the host on which the clone database will be attached to a SQL Server instance.
4. SMSQL now calls upon the services of SDW to create a clone of the database. For this, it passes the Snapshot copy either as the one the user has created or the one created in step 2.
5. SDW goes ahead and begins a clone creation process at the storage level. It begins by checking for the presence of the FlexClone license. If FlexClone is licensed, then a FlexClone clone of the Snapshot copy is created. Otherwise the Snapshot copy is mounted as a read-write copy. SDW then proceeds to connect to the LUN inside the FlexClone clone or the mounted Snapshot copy.
6. Then the SMSQL instance on the originating host initiates a VDI-based restore of the database to the desired SQL Server instance on the desired host.

Database cloning is a very useful feature and it can be successfully deployed in the following scenarios:

- Creation of database clones for development and test environments.

  TIP: For such an environment it suffices to create clones from historical backup sets with a limited number of logs played forward.

- Creation of staging environments that act as exact replicas of production environments.

  TIP: For such an environment make sure that, during the clone creation process, the option to back up the TLOG immediately after the backup is selected. This increases the currency of the cloned copy.

- It is preferable to have FlexClone licensed for storage systems on which clones that are to be used for longer duration are created.
- It is of great utility in reporting environments where load from the main OLTP database can be diminished in order to run reports during peak time.

Note: If a database resides on a virtual machine with a VMDK disk, it is not possible to clone the database to a physical server.

Dataset Integration with SMSQL

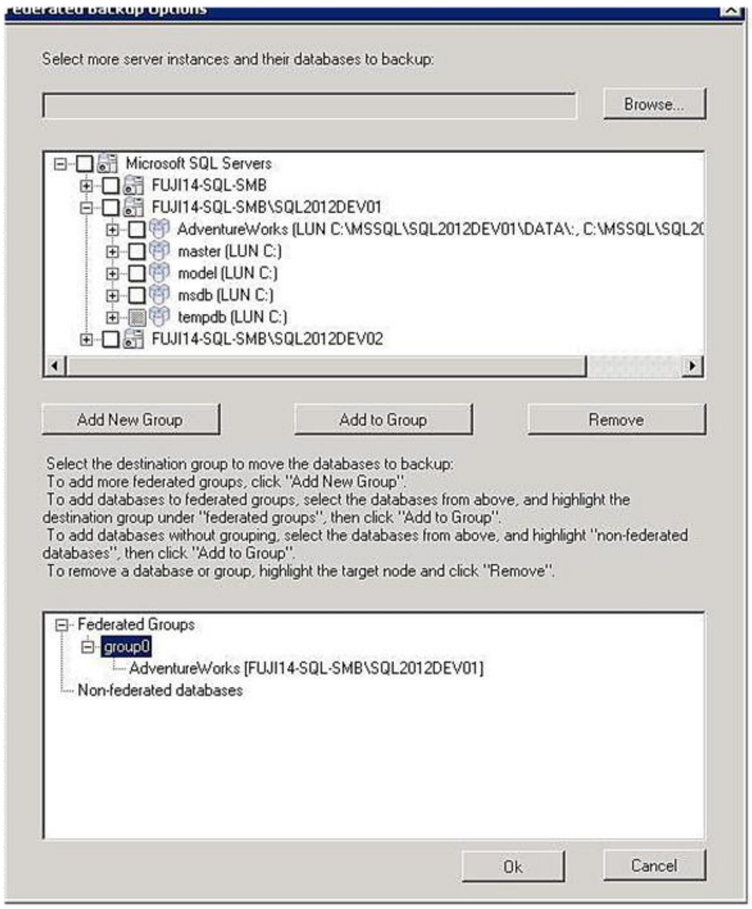
SMSQL is integrated with NetApp Protection Manager datasets. This integration allows DBAs to archive database backups to a NetApp SnapVault secondary destination. Thus SMSQL offers a complete disk-to-disk backup archival solution integrated with the product. The benefits of this feature are:

- Allowing expensive primary storage space to be used by fewer online backups without losing older backup sets

- Planning added disaster recovery functionality
- Planning legal compliance strategies that require organizations to preserve older backups

Federated Backup

Federated backup in SMSQL 5.2 provides the new federation feature so that users can specify databases not only from different SQL instances on the same host, but also from remote hosts on different or the same storage controllers. Users can connect to the remote servers, select databases, add them to the backup list, and create Snapshot copies on remote servers as well as local backups at the same time.


Select more server instances and their databases to backup:

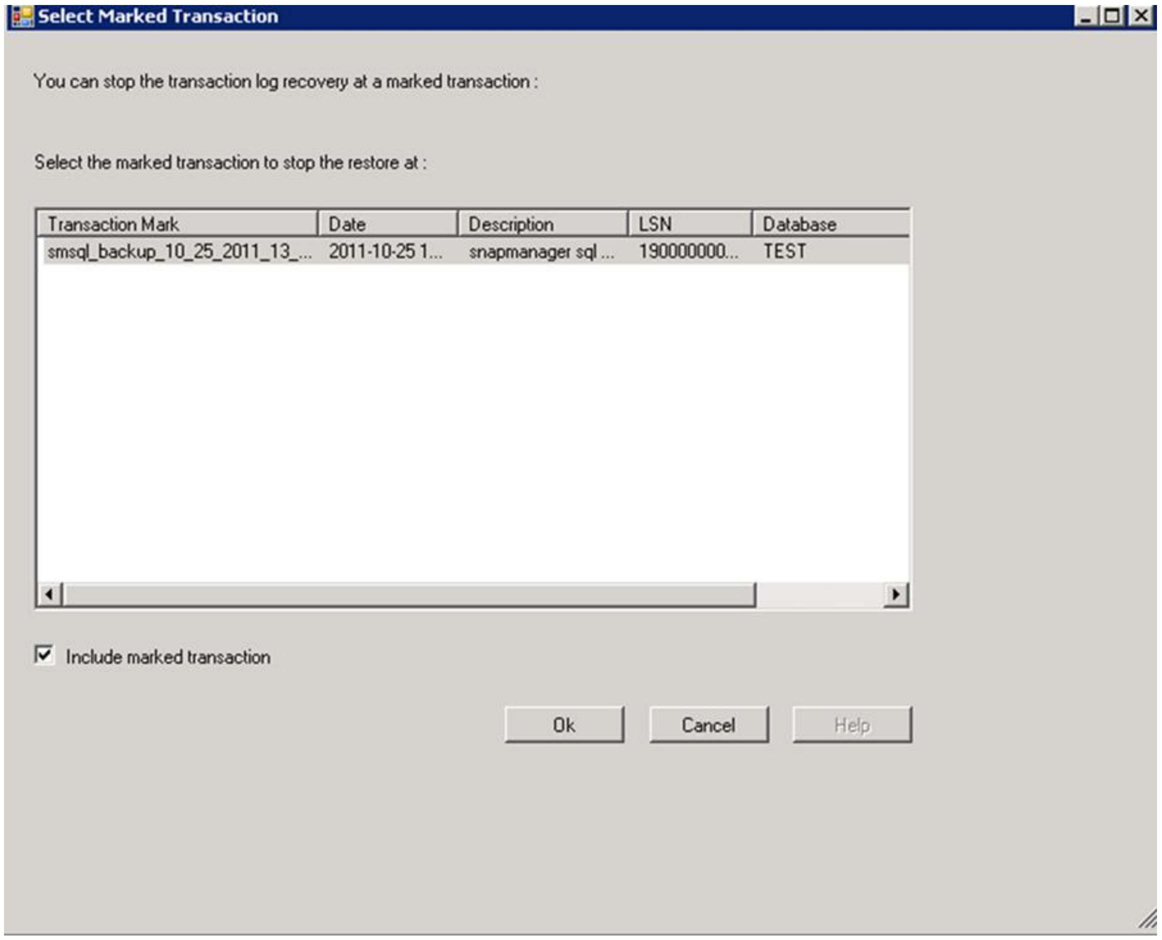
Microsoft SQL Servers
FUJI14-SQL-SMB
FUJI14-SQL-SMB\SQL2012DEV01
AdventureWorks (LUN C:\MSSQL\SQL2012DEV01\DATA\:, C:\MSSQL\SQL2
master (LUN C:)
model (LUN C:)
msdb (LUN C:)
tempdb (LUN C:)
FUJI14-SQL-SMB\SQL2012DEV02

Select the destination group to move the databases to backup:
To add more federated groups, click "Add New Group".
To add databases to federated groups, select the databases from above, and highlight the destination group under "federated groups", then click "Add to Group".
To add databases without grouping, select the databases from above, and highlight "non-federated databases", then click "Add to Group".
To remove a database or group, highlight the target node and click "Remove".

Federated Groups
group0
AdventureWorks [FUJI14-SQL-SMB\SQL2012DEV01]
Non-federated databases

Federated Log Restore and Restore Log to Mark

The following screenshot shows restoration of log files in the Federated feature and setting up a marker for restoring a log to mark.



You can stop the transaction log recovery at a marked transaction :

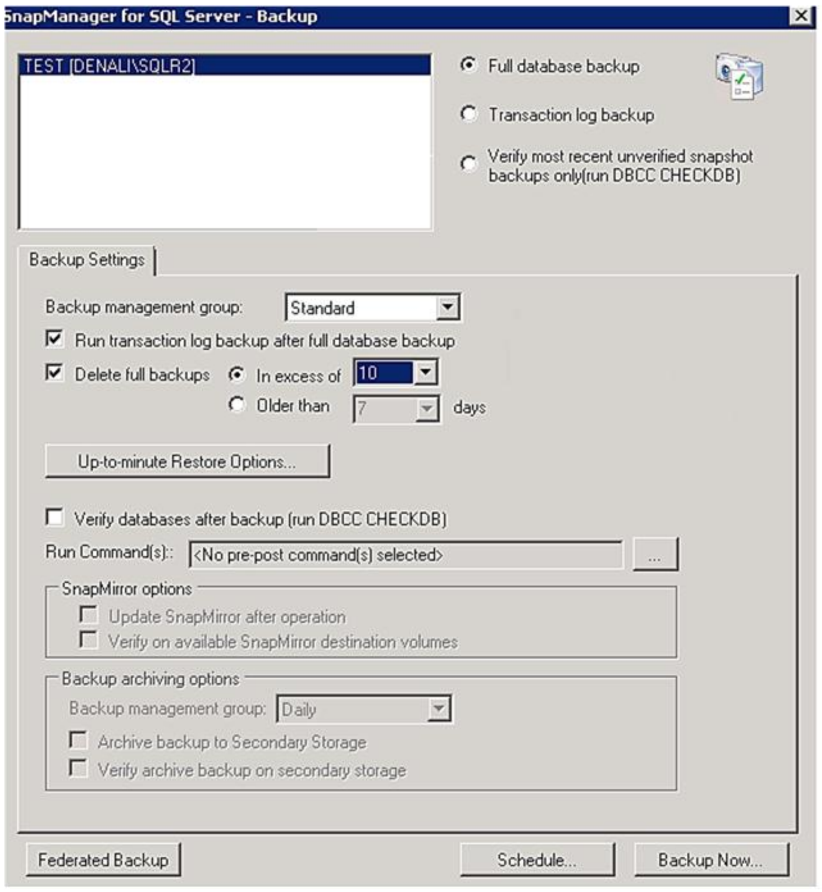
Select the marked transaction to stop the restore at :

Transaction Mark	Date	Description	LSN	Database
smsql_backup_10_25_2011_13_...	2011-10-25 1...	snapmanager sql ...	190000000...	TEST

☑ Include marked transaction

Backup Retention Management

This is an efficient way to manage the retention of transaction log backups, because we need to limit the number of old transaction logs we save. When the new backup runs, it marks the previous backups as being point-in-time only, then deletes the log(s) as well as the Snapshot copies to reduce log traffic.


SnapManager for SQL Server - Backup

TEST [DENALI\SQLR2]

[✓ Full database backup] Transaction log backup Verify most recent unverified snapshot backups only(run DBCC CHECKDB)

Backup Settings

Backup management group: Standard

[✓] Run transaction log backup after full database backup

[✓] Delete full backups [✓ In excess of] 10 Older than 7 days

Up-to-minute Restore Options...

[ ] Verify databases after backup (run DBCC CHECKDB)

Run Command(s):: <No pre-post command(s) selected>

SnapMirror options

[ ] Update SnapMirror after operation

[ ] Verify on available SnapMirror destination volumes

Backup archiving options

Backup management group: Daily

[ ] Archive backup to Secondary Storage

[ ] Verify archive backup on secondary storage

General Support for SQL Server 2012

SnapManager 6.0 for SQL Server is the latest NetApp software solution that provides data protection and management in SQL Server environments.This release is primarily targeted to support SQL Server 2012.

SQL Server 2012 introduces new high-availability technology called AlwaysOn. AlwaysOn provides both database- and instance-level availability and enables businesses to achieve the required "9s" for their mission-critical applications. It is an advanced version of the database mirroring concept, which previously was limited in scope to single nonreadable secondary databases.

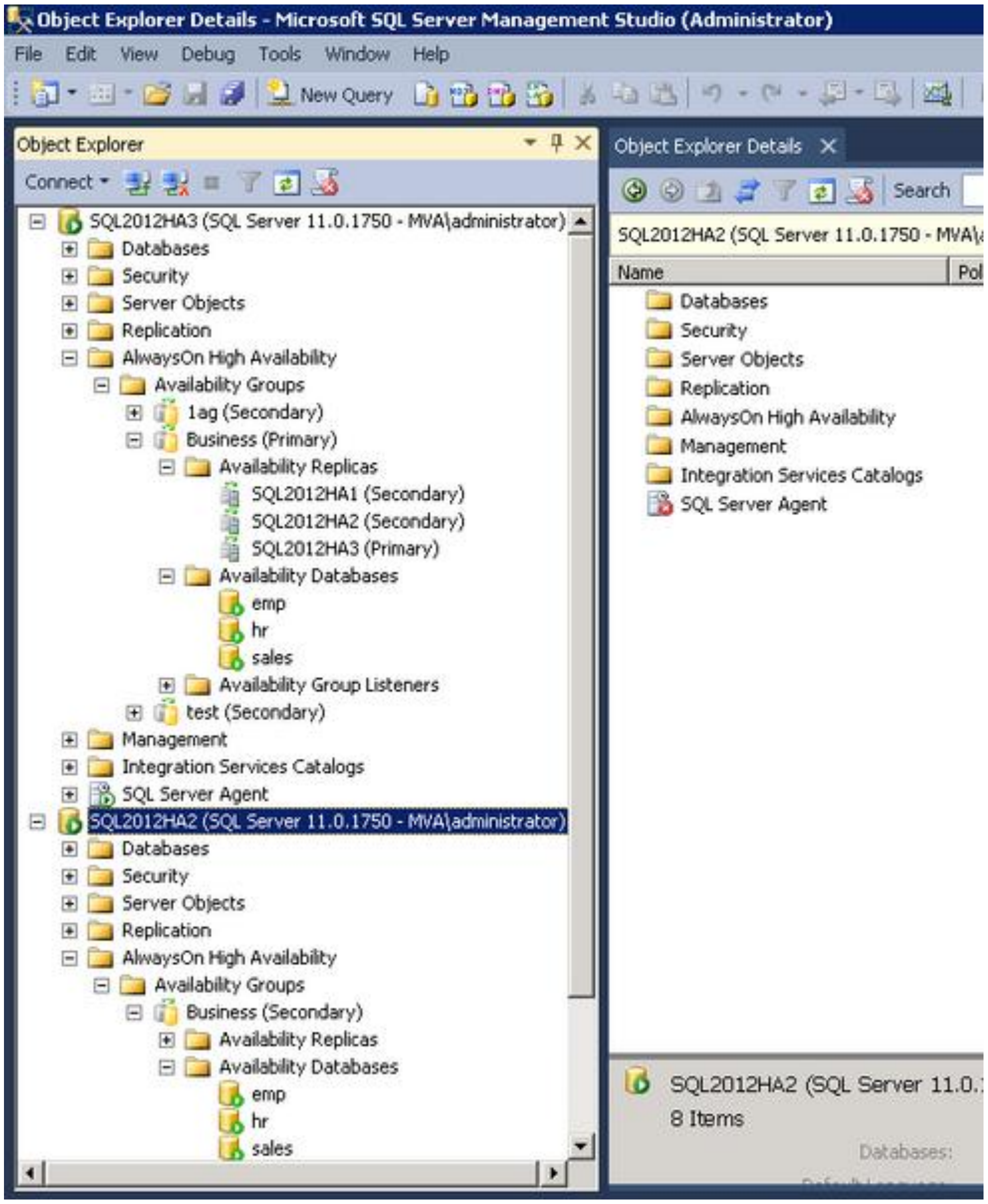
AlwaysOn introduces the following enhancements:

- Provides enhanced failover functionality using Windows Server Failover Clustering (WSFC)
- Eliminates the requirement for shared storage and removes the single point of failure
- Offers multiple availability databases, compared to database mirroring
- Features AlwaysOn listener service, which accepts all requests to database Availability Groups
- Can expand over multiple subnets
- Gives the option to offload backup and maintenance operations to a read-only replica database
- Offers support for both synchronous and asynchronous nodes

Using this version of SnapManager 6.0, users can:

- Create Snapshot based full backup and transaction log backup on SQL Server 2012 user databases

- Create stream-based full backup and transaction log backup on SQL Server 2012 system databases (except tempdb)
- Perform restore on SQL Server 2012 databases, including restoring a Snapshot full backup and applying transaction log backups
- Perform up-to-the-minute restore, point-in-time restore, and restore-to-mark restore
- Create federated full backup and federated log backup with SQL Server 2012 databases included in one or more federated groups along with databases from other versions of SQL Server instances
- Create clone databases on SQL Server 2012 instances
- Automatically refresh the clone database on a SQL Server 2012 instance
- Automatically delete the cloned database on a SQL Server 2012 instance when it is no longer needed

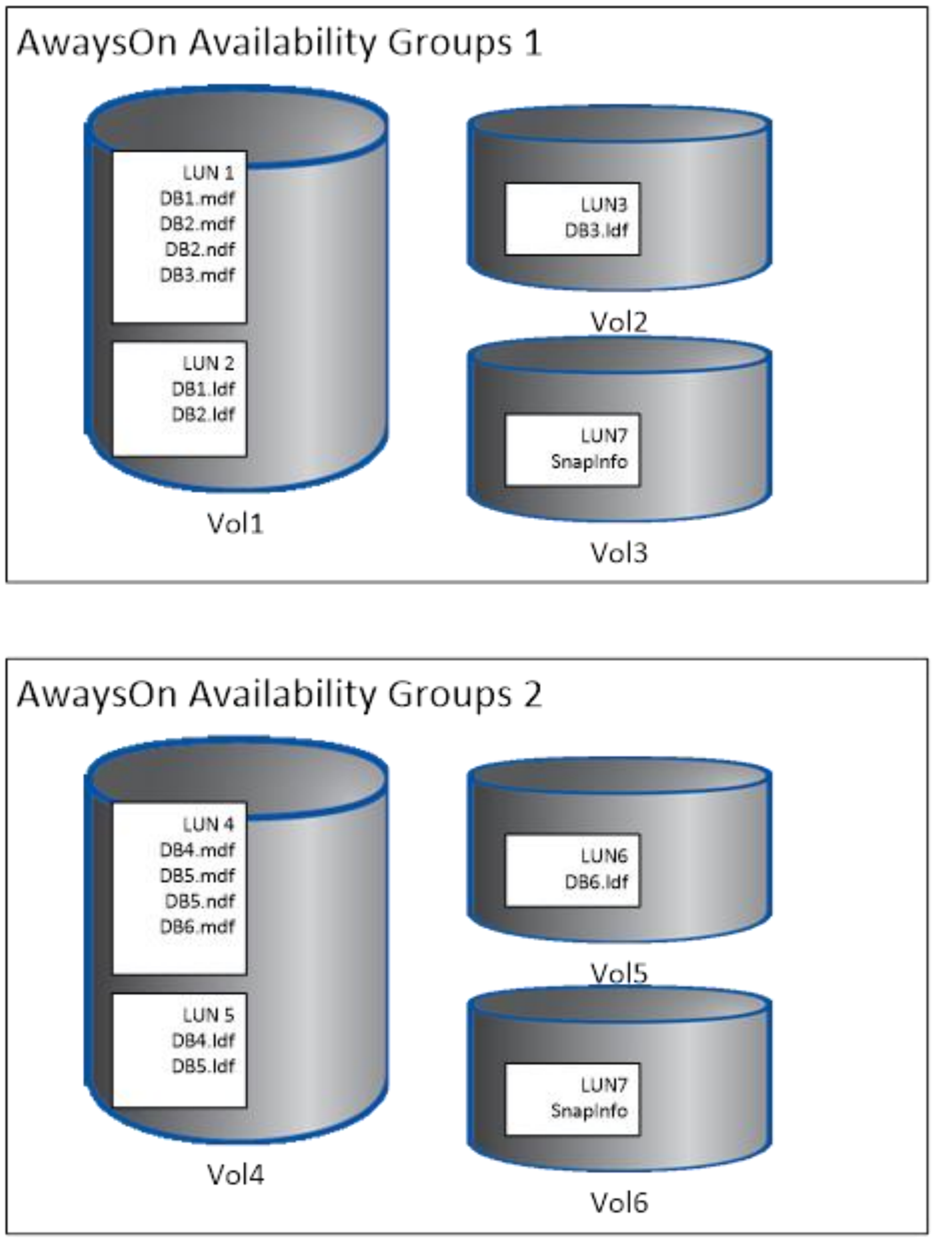


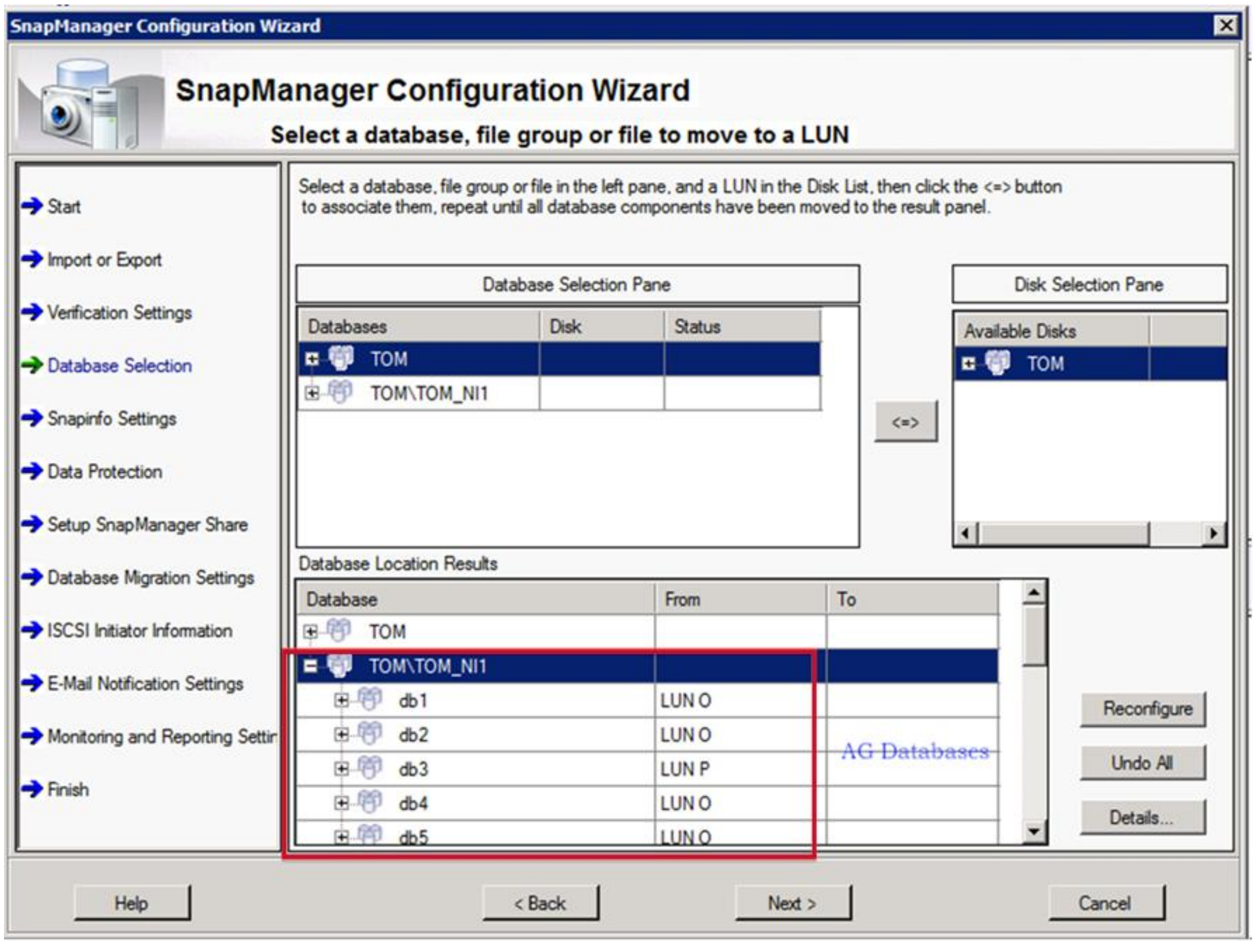
Configuration with AlwaysOn

Migration of Availability Group (AG) databases is accomplished by using the SnapManager Configuration Wizard. SMSQL 6.0 supports a mix of primary, secondary, non-Availability group, and system databases during configuration. When the migration is complete, the existing Availability Group setup is not disturbed and databases remain active.

Best Practices: Configuration with AlwaysOn
• NetApp recommends that user databases, including Availability Group databases not share the same LUN or VMDK as system databases. • To improve Availability Group backup performance, NetApp recommends using the preferred backup replica option with variable weights for each replica. • At least one full backup should be present on all the participating AlwaysOn nodes. NetApp recommends regularly backing up transaction logs from the preferred replica or if the SMSQL driving server is the primary replica. • It is important to check the status of the Availability Group databases before launching the AG backup. SMSQL 6.0 will skip the backup of the Availability Group databases if the Availability Group databases are not synchronizing or in synchronization with a particular SQL Server replica. • By default, the secondary replicas are nonreadable in SQL Server. It is important to make the secondary replica readable. To do this, set the “Readable Secondary” property for AG replica to Yes. The replicas will not be able to take part in backups without this setting. • In automatic failover, you can have one primary replica and one out of the four maximum secondary replicas participating. If the autofailover option is not selected, the user has to perform a manual failover of the Availability Group. Even if a single database in the AG primary replica is corrupt or deleted (unhealthy) there will be an automatic failover of AG (if configured) to secondary replicas. • The primary instance of each Availability Group must have its own SnapInfo LUN/Volume. The secondary failover instance must also have its own SnapInfo LUN/volume. • Use SMSQL share to synchronize all transaction logs from the backup. SMSQL share can be used for various Availability Groups.

Figure 5) AlwaysOn Availability Groups.




SnapManager Configuration Wizard
SnapManager Configuration Wizard
Select a database, file group or file to move to a LUN
Start
Import or Export
Verification Settings
Database Selection
SnapInfo Settings
Data Protection
Setup SnapManager Share
Database Migration Settings
ISCSI Initiator Information
E-Mail Notification Settings
Monitoring and Reporting Settir
Finish
Database Selection Pane
Disk Selection Pane
<=>
Database Location Results
AG Databases
Reconfigure
Undo All
Details...
Help
< Back
Next >
Cancel

Select a database, file group or file in the left pane, and a LUN in the Disk List, then click the <=> button to associate them, repeat until all database components have been moved to the result panel.

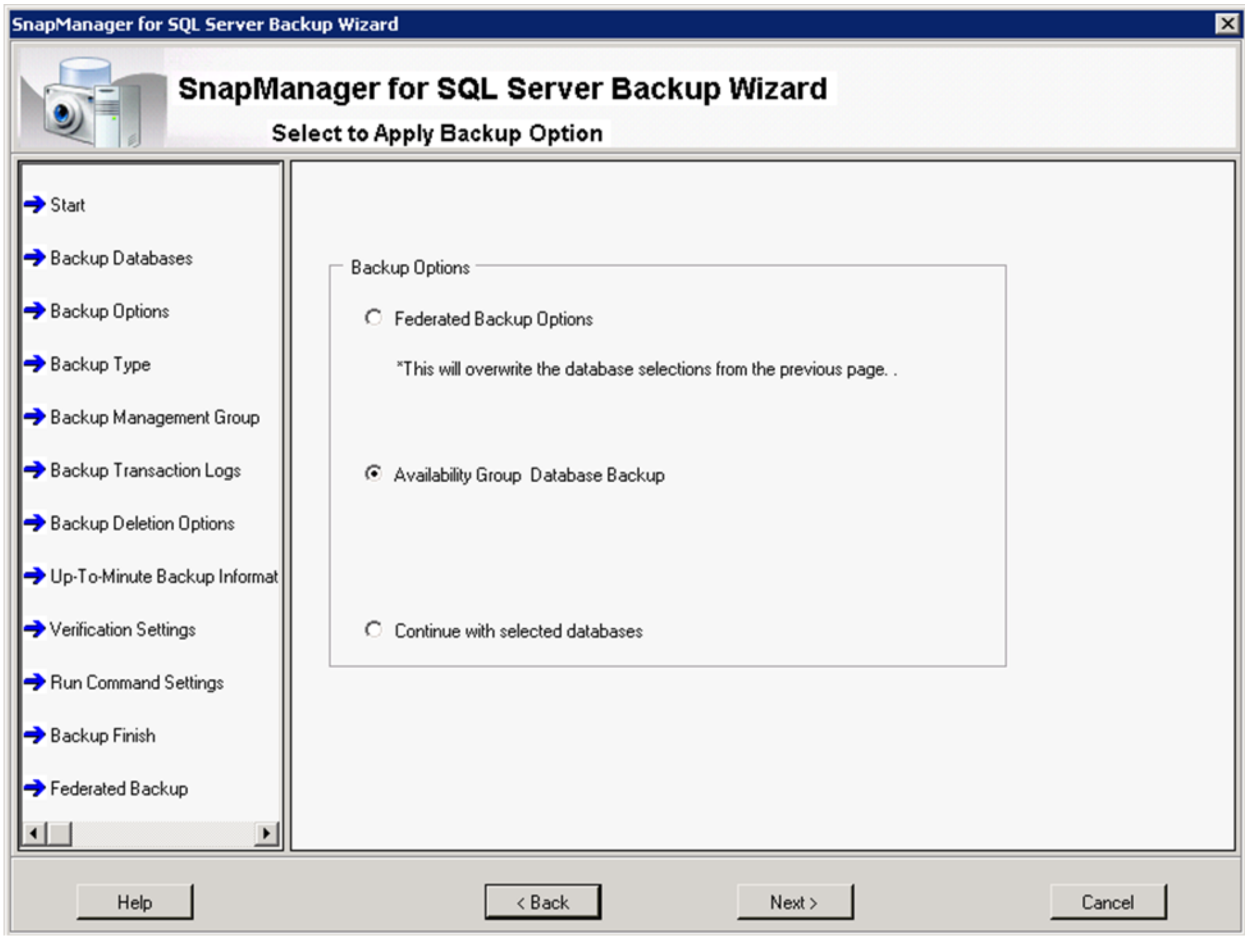
Databases	Disk	Status
TOM		
TOM\TOM_NI1		

Available Disks
TOM

Database	From	To
TOM		
TOM\TOM_NI1		
db1	LUN O	
db2	LUN O	
db3	LUN P	
db4	LUN O	
db5	LUN O	

Backup with AlwaysOn

To initiate SQL Server 2012 Availability Group database backup from the Backup Wizard, select the Availability Group Database Backup option in the Select to Apply Backup Option page. The Availability Group Database Backup feature allows users to back up AG databases across servers in a single operation. Users can select databases or AGs, add them to backups with parameters like Preferred Replica or Primary, Secondary, and Backup Priority, and make Snapshot copies on local as well as remote AG database servers. Connection between servers where the AG backup job was created is maintained by SMSQL.



Backup Options

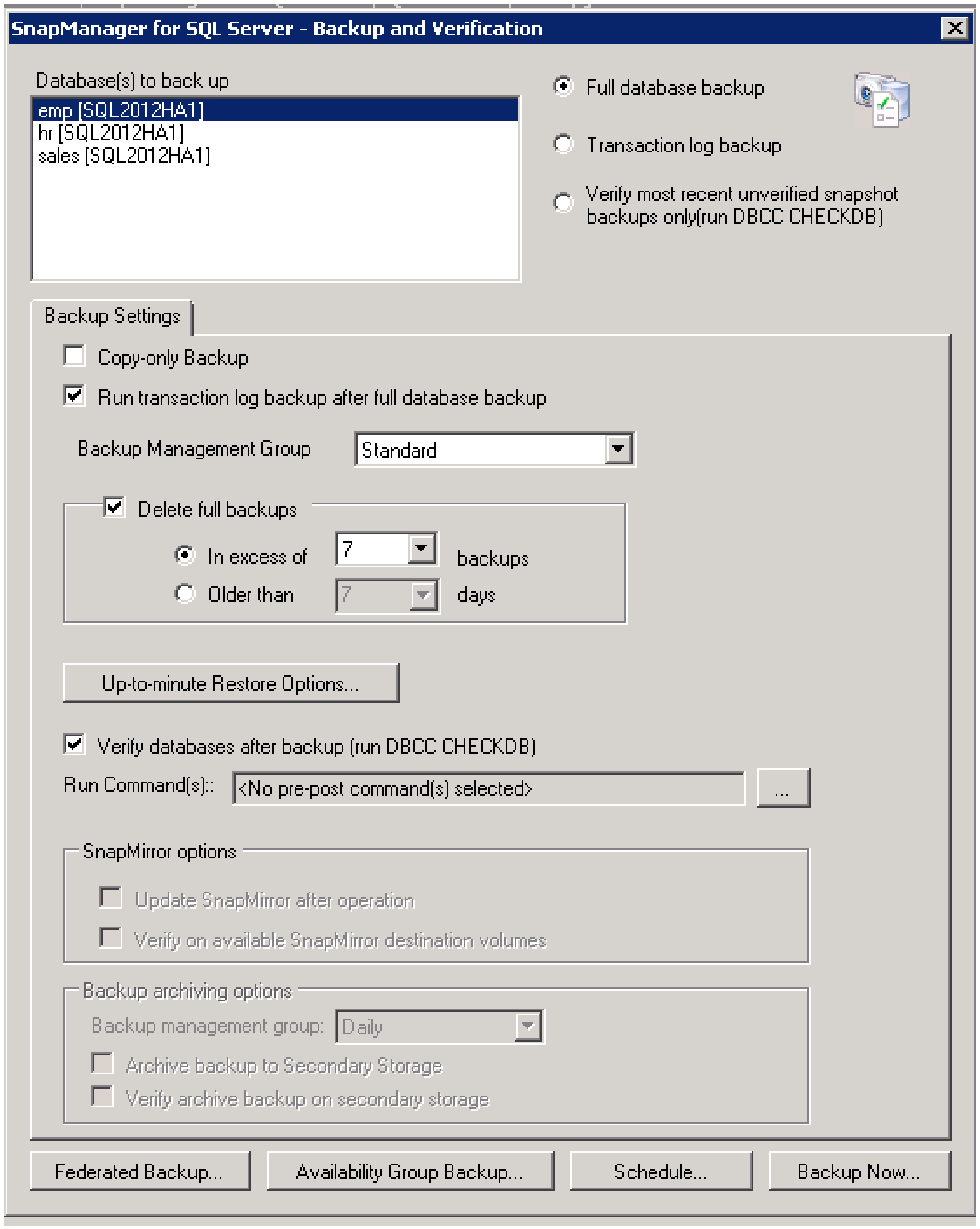
( ) Federated Backup Options

*This will overwrite the database selections from the previous page. .

[✓ Availability Group Database Backup]

( ) Continue with selected databases

The other option to initialize an Availability Group backup is to right-click one or more databases from the backup view and select the Backup and Verify option. The Availability Group Backup button is now displayed.



SnapManager for SQL Server - Backup and Verification

Database(s) to back up
- emp [SQL2012HA1]
- hr [SQL2012HA1]
- sales [SQL2012HA1]

(●) Full database backup
( ) Transaction log backup
( ) Verify most recent unverified snapshot backups only(run DBCC CHECKDB)

Backup Settings

[ ] Copy-only Backup
[✓] Run transaction log backup after full database backup

Backup Management Group: Standard

[✓] Delete full backups
(●) In excess of 7 backups
( ) Older than 7 days

Up-to-minute Restore Options...

[✓] Verify databases after backup (run DBCC CHECKDB)

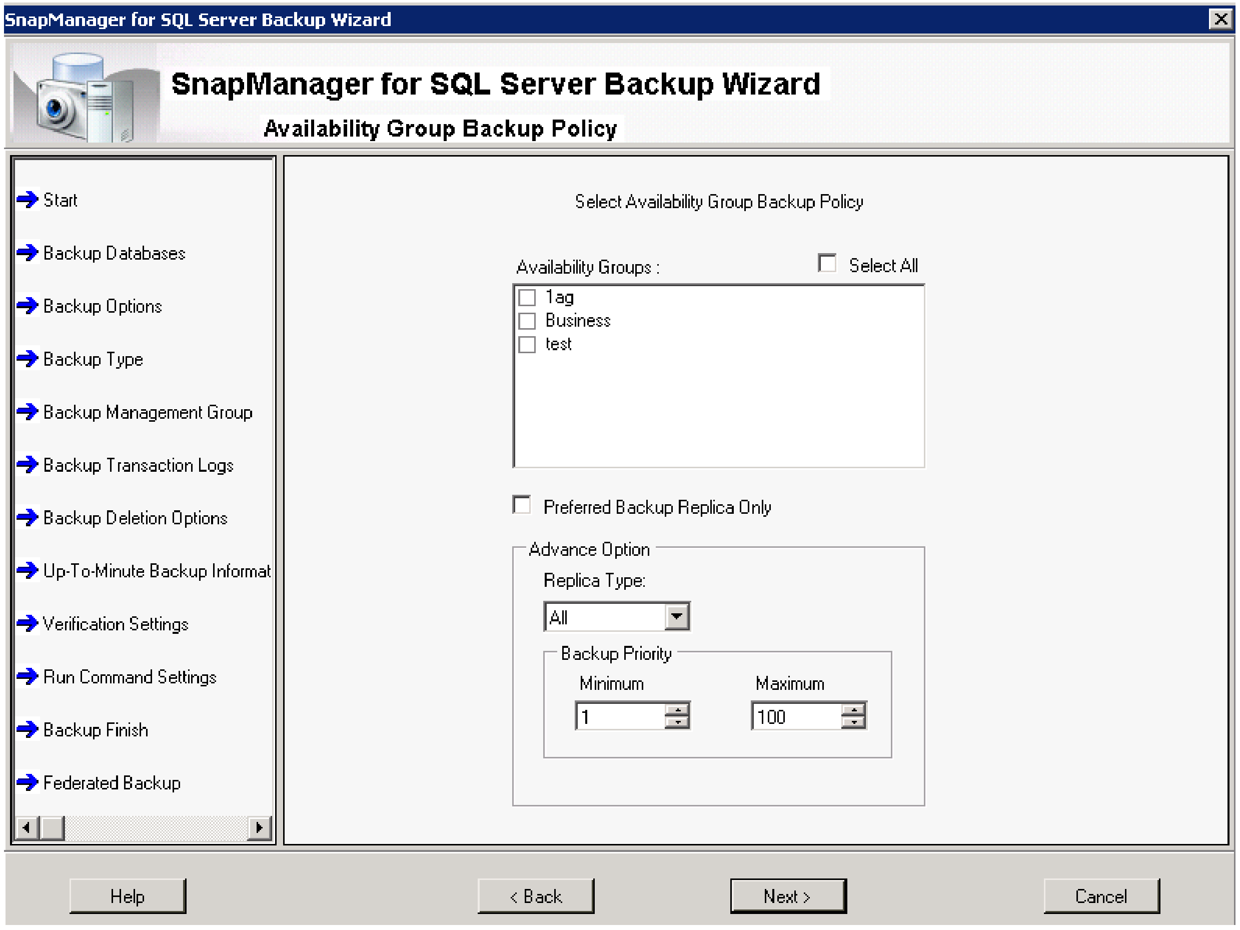
Run Command(s):: <No pre-post command(s) selected> ...

SnapMirror options
[ ] Update SnapMirror after operation
[ ] Verify on available SnapMirror destination volumes

Backup archiving options
Backup management group: Daily
[ ] Archive backup to Secondary Storage
[ ] Verify archive backup on secondary storage

Federated Backup... | Availability Group Backup... | Schedule... | Backup Now...

The option to select Preferred Backup Replica is based on the backup preference set at the SQL 2012 level. The other option is to specify the Replica Type and the Backup Priority range.



Note: If you select the Preferred Backup Replica Only option, SMSQL inherits the backup replica settings from SQL Server 2012. You can use the Replica Type drop-down menu to select the nodes on which the backup is to be taken.

Note: In the Backup Priority fields, set minimum and maximum values. By default 1,100 indicates that all the nodes, including primary, are part of the Availability Group backup plan.

Best Practice: Availability Group Backup Performance

The backup job is assigned to the SQL Server agent on the driving server—the server from which the SMSQL Wizard launches the backup.

To improve performance, NetApp recommends using the Preferred backup Replica option.

Best Practice: Backup Intervals

At least one full backup should be present on all participating AlwaysOn nodes. NetApp also recommends that you perform transaction log backups at a regular interval from the preferred replica or from the SMSQL driving server, if it is the primary replica.

Note: After Availability Group backup starts, SnapManager waits at multiple synchronization points until all the backup threads—each on a different server—in one AG replica are in the signaled state. If the driving server cannot send a backup request or connect to one of the remote servers, or if backup fails on one of the servers, SnapManager times out in 10 minutes. The timeout value can be manually set in the registry of the connected server.

The registry key for the timeout value of the AG log backup with mark is:

– `FederatedReadyMarkTLBTimeOut`

A new registry key introduced in SMSQL 6.0 for Federated and Availability Group is the timeout verification value.

– `FederatedReadyFreeHandlesTimeOut`

Use the following command to disable AG backup synchronization:
`DisableFederatedBackupSynch`

Best Practice: Availability Group Database Status

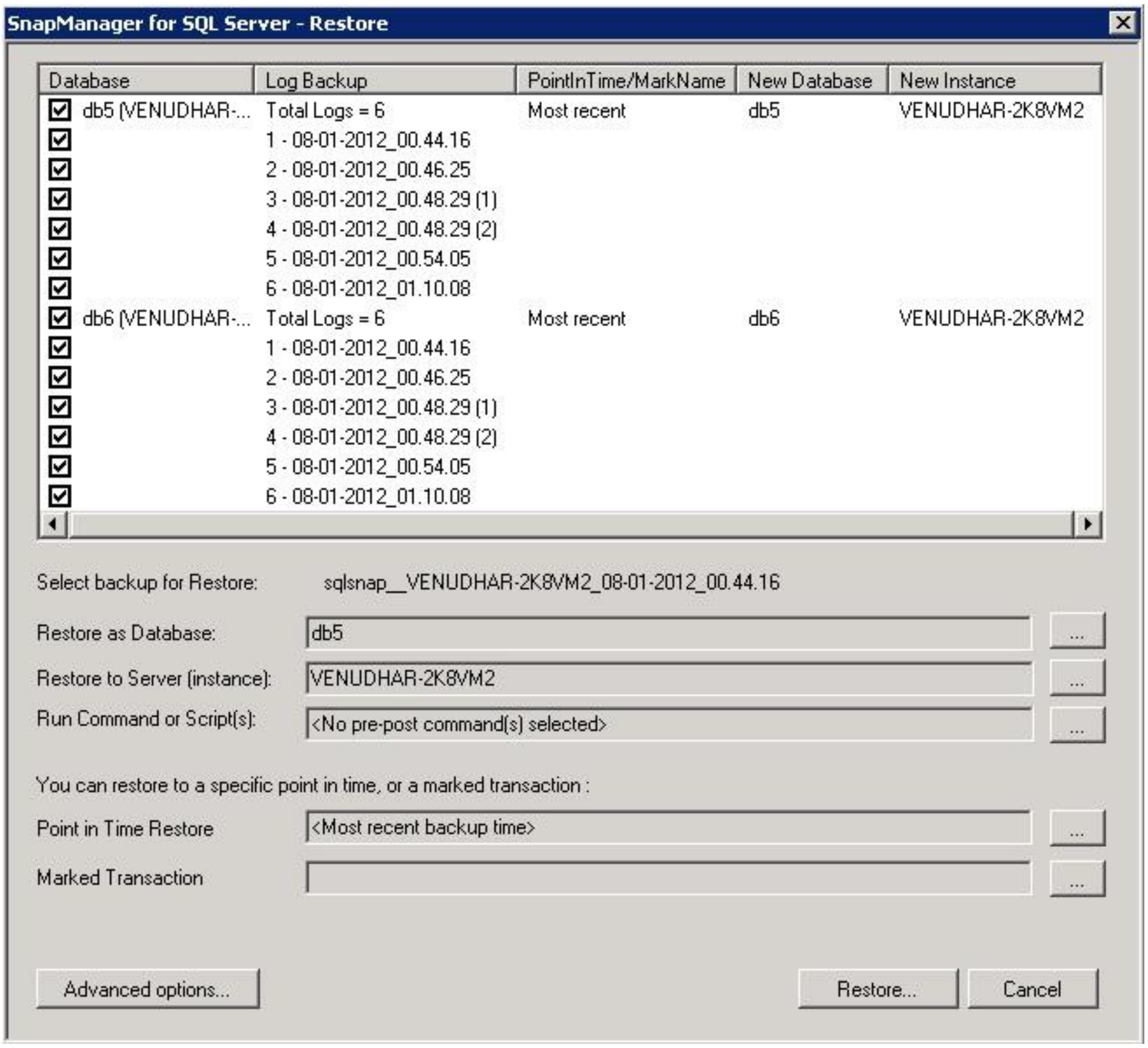
It's important to check the database status of Availability Group databases before launching AG backup. SMSQL 6.0 skips AG database backup if the AG databases are not in Synchronizing or Synchronized state on a particular SQL Server replica.

Best Practice: Make Secondary Replicas Readable

By default, the secondary replicas are nonreadable in SQL Server. Therefore it is important to make the secondary replica readable by setting the Readable Secondary property for the AG replica to Yes. If this property is set to No, these replicas are not included in backups.

Restore and Reseed with AlwaysOn

SnapManager 6.0 introduces restore and reseed functionalities to facilitate faster recovery of Availability Group databases and to restore them to a synchronized state. To perform a restore of the entire Availability Group setup to a point in time, use the Restore Wizard. The option to restore AG databases is supported on both primary and secondary replicas.



	Database	Log Backup	PointInTime/MarkName	New Database	New Instance
☑	db5 (VENUDHAR-...	Total Logs = 6	Most recent	db5	VENUDHAR-2K8VM2
☑		1 - 08-01-2012_00.44.16			
☑		2 - 08-01-2012_00.46.25			
☑		3 - 08-01-2012_00.48.29 (1)			
☑		4 - 08-01-2012_00.48.29 (2)			
☑		5 - 08-01-2012_00.54.05			
☑		6 - 08-01-2012_01.10.08			
☑	db6 (VENUDHAR-...	Total Logs = 6	Most recent	db6	VENUDHAR-2K8VM2
☑		1 - 08-01-2012_00.44.16			
☑		2 - 08-01-2012_00.46.25			
☑		3 - 08-01-2012_00.48.29 (1)			
☑		4 - 08-01-2012_00.48.29 (2)			
☑		5 - 08-01-2012_00.54.05			
☑		6 - 08-01-2012_01.10.08			

Select backup for Restore: sqlsnap__VENUDHAR-2K8VM2_08-01-2012_00.44.16

Restore as Database: db5

Restore to Server (instance): VENUDHAR-2K8VM2

Run Command or Script(s): <No pre-post command(s) selected>

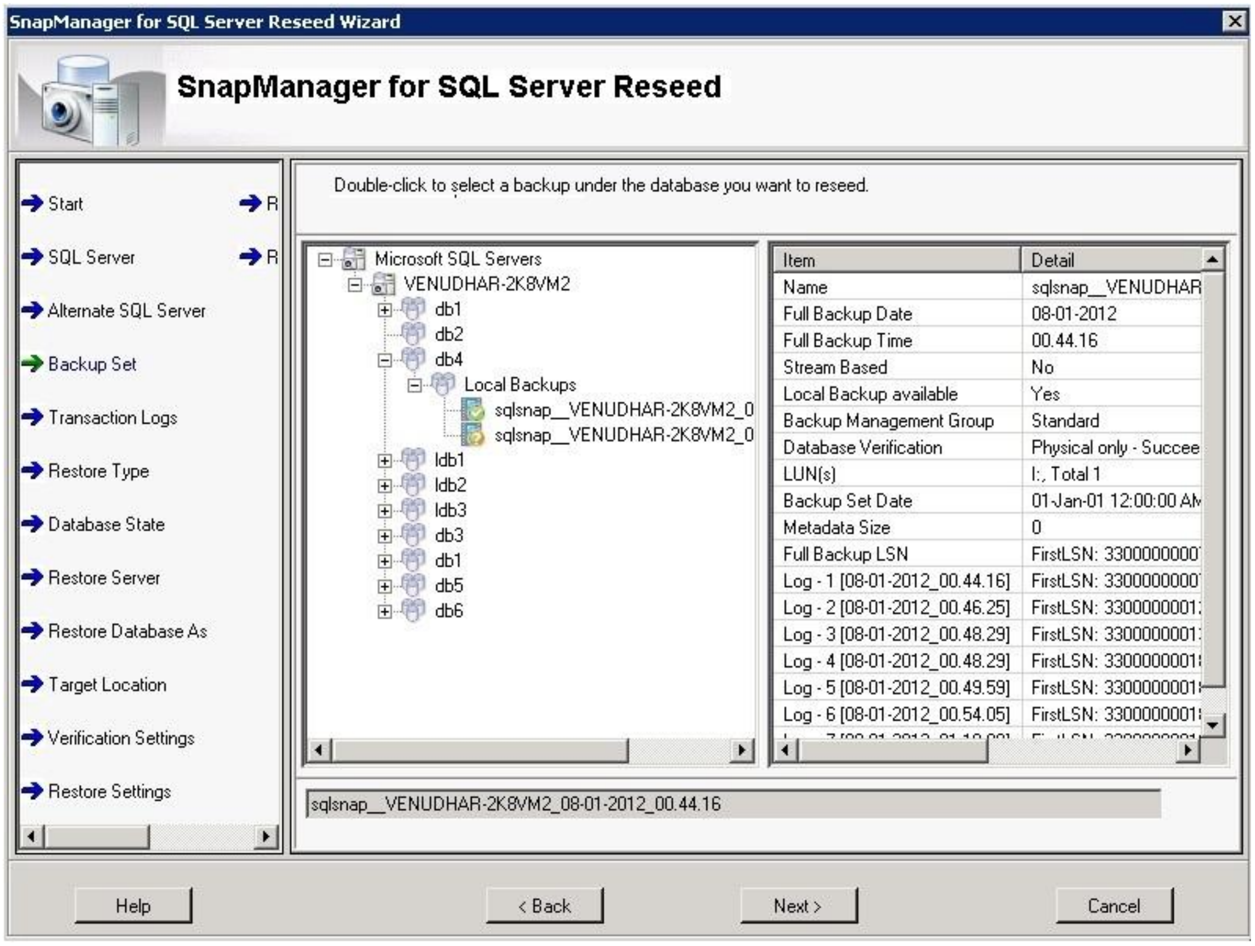
You can restore to a specific point in time, or a marked transaction :

Point in Time Restore <Most recent backup time>

Marked Transaction

Note: When the Availability Group databases are restored by using the Restore Wizard, the SQL Server databases are still not part of the Availability Group.

A significant feature of SnapManager 6.0 for SQL Server is the Reseed Wizard. In an AlwaysOn environment, AG failover can happen whenever there is a failure or corruption of one or more AG databases. This failover can be automatic or manual, based on the availability replica's configuration. In such cases, the primary replica switches to the secondary role, and one of the secondary replicas becomes the primary. This means that primary databases are generally always active, and secondary databases can be in an unhealthy state, such as suspend, removed, not-synchronizing, and so on. The reseed operation recovers such databases and brings them back into the Availability Group and into synch with primary databases. Therefore Reseed is applicable only for secondary databases.



Double-click to select a backup under the database you want to reseed.

```
Microsoft SQL Servers
  VENUDHAR-2K8VM2
    db1
    db2
    db4
      Local Backups
        sqlsnap__VENUDHAR-2K8VM2_0
        sqlsnap__VENUDHAR-2K8VM2_0
    ldb1
    ldb2
    ldb3
    db3
    db1
    db5
    db6
```

Item	Detail
Name	sqlsnap__VENUDHAR
Full Backup Date	08-01-2012
Full Backup Time	00.44.16
Stream Based	No
Local Backup available	Yes
Backup Management Group	Standard
Database Verification	Physical only - Succee
LUN(s)	I:, Total 1
Backup Set Date	01-Jan-01 12:00:00 AM
Metadata Size	0
Full Backup LSN	FirstLSN: 3300000000
Log - 1 [08-01-2012_00.44.16]	FirstLSN: 3300000000
Log - 2 [08-01-2012_00.46.25]	FirstLSN: 3300000001
Log - 3 [08-01-2012_00.48.29]	FirstLSN: 3300000001
Log - 4 [08-01-2012_00.48.29]	FirstLSN: 3300000001
Log - 5 [08-01-2012_00.49.59]	FirstLSN: 3300000001
Log - 6 [08-01-2012_00.54.05]	FirstLSN: 3300000001

sqlsnap__VENUDHAR-2K8VM2_08-01-2012_00.44.16

Note: One primary and one of the four (maximum) secondary replicas can participate in an autofailover. If the Autofailover option is not selected, the user must initiate a manual failover of the Availability Group.

If even one database in the AG primary replica becomes corrupted or deleted (unhealthy), then automatic failover of the AG (if configured) to secondary replicas occurs. When the failover occurs, the primary becomes the secondary replica database.

Best Practices: Reseed Wizard

- Always run the Reseed Wizard on the secondary replica where there is a failure after failover.
- Only databases that are unhealthy are reseeded.
- The databases that are part of the Availability Group and healthy are not reseeded.
- The databases that are part of a primary are skipped and not reseeded.
- If the user has used the Restore Wizard, then all the databases are restored; when using the Reseed Wizard, only unhealthy databases are reseeded.

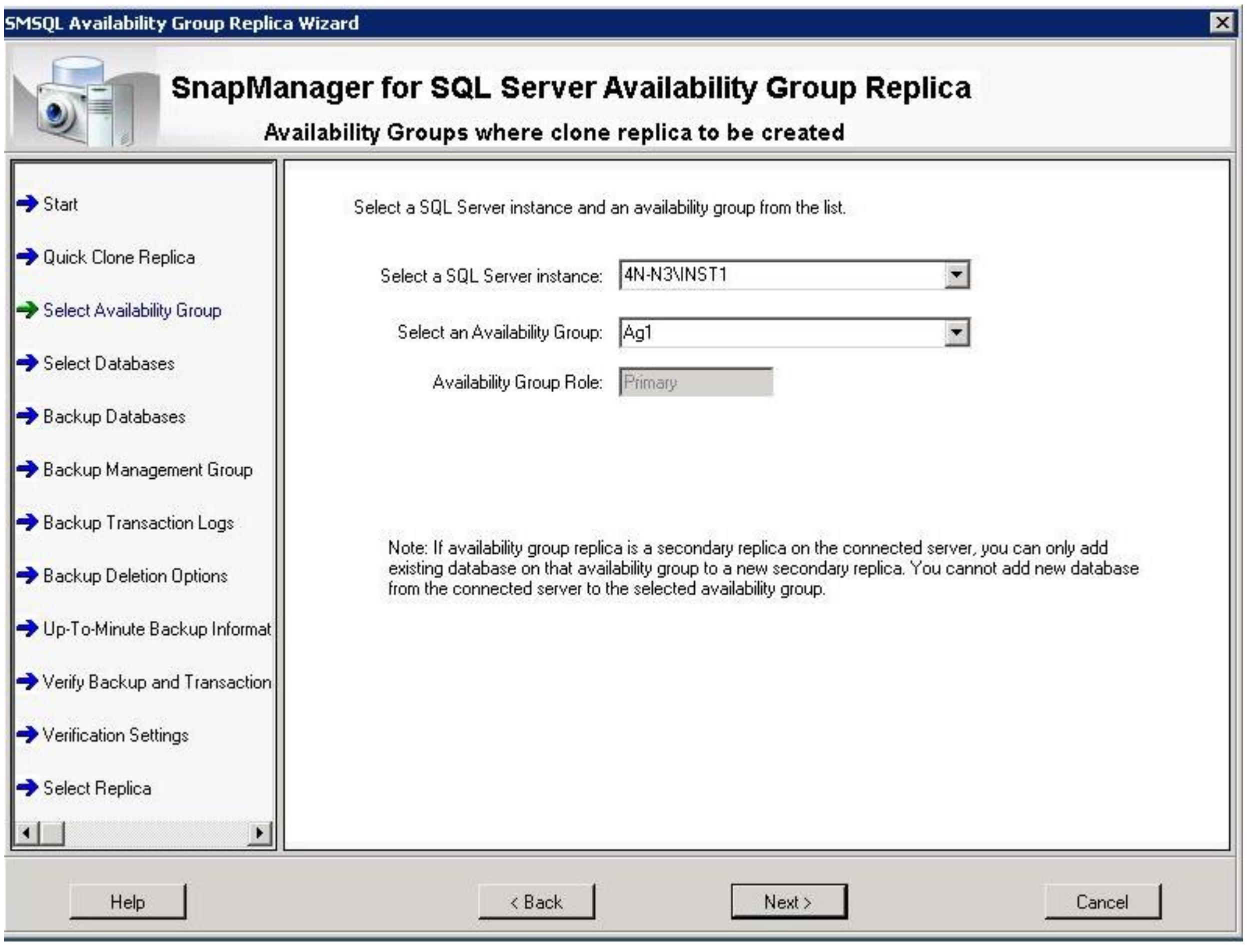
Best Practice: Share for Reseed

For the Reseed Wizard to be successful, a SnapManager share should be configured for all the replicas that should have read/write permission to store the log backups.

AlwaysOn Clone Replica Wizard

The SMSQL cloning feature is now extended from the database level to the SQL Server 2012 Availability Groups level. Availability Group Clone Replica quickly replicates databases to a remote cluster SQL Server instance before grouping them in an Availability Group. Database replication leverages Snapshot-based virtual replication to create the database replica. Replicated databases are then associated with another instance in the same cluster so that Availability Group failover can take place when required.

The Replica Wizard can be launched from the Actions pane in the SnapManager console.


SMSQL Availability Group Replica Wizard
SnapManager for SQL Server Availability Group Replica
Availability Groups where clone replica to be created
Start
Quick Clone Replica
Select Availability Group
Select Databases
Backup Databases
Backup Management Group
Backup Transaction Logs
Backup Deletion Options
Up-To-Minute Backup Informat
Verify Backup and Transaction
Verification Settings
Select Replica
Help
< Back
Next >
Cancel

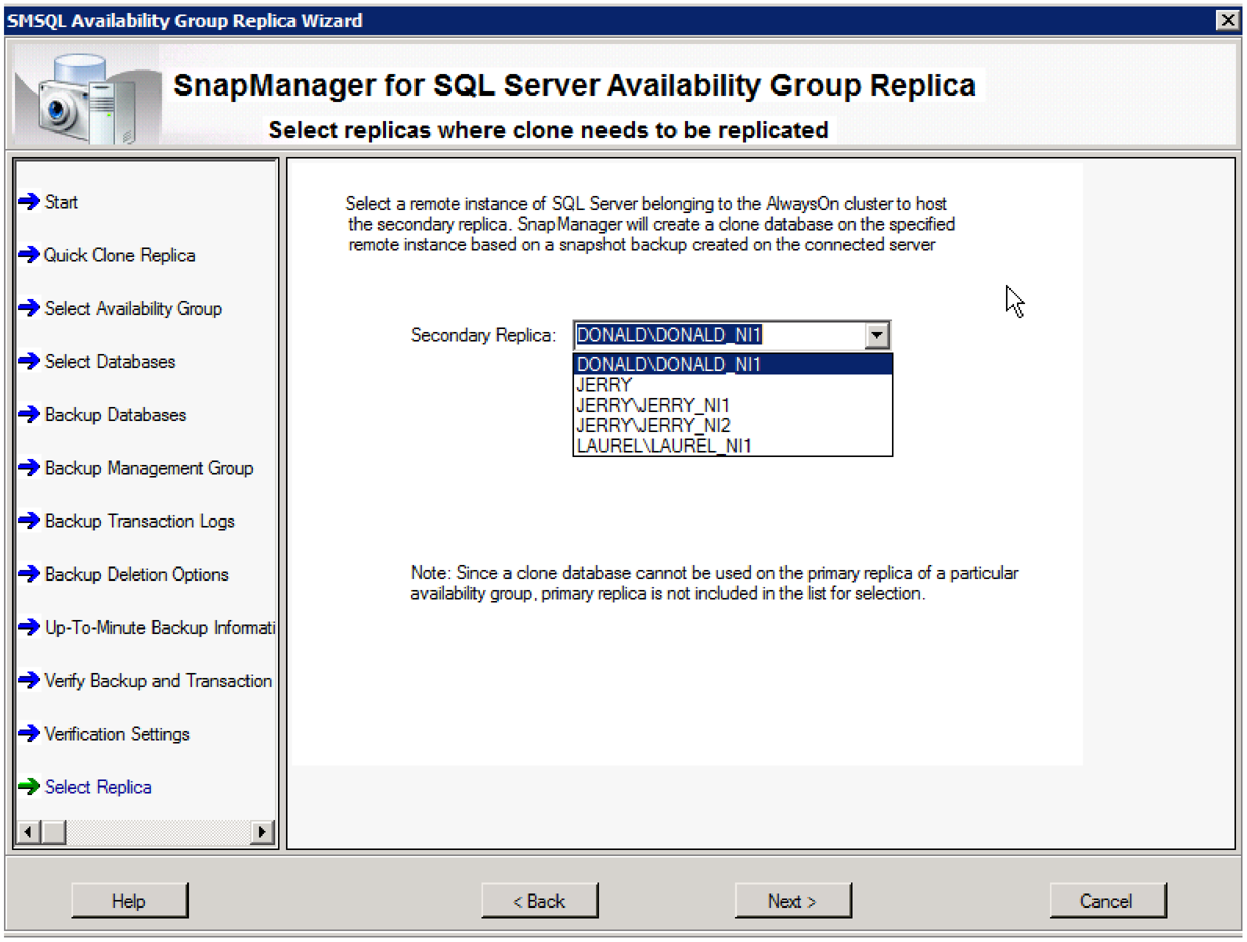
Select a SQL Server instance and an availability group from the list.

Select a SQL Server instance: 4N-N3\INST1

Select an Availability Group: Ag1

Availability Group Role: Primary

Note: If availability group replica is a secondary replica on the connected server, you can only add existing database on that availability group to a new secondary replica. You cannot add new database from the connected server to the selected availability group.


SMSQL Availability Group Replica Wizard
SnapManager for SQL Server Availability Group Replica
Select replicas where clone needs to be replicated
Start
Quick Clone Replica
Select Availability Group
Select Databases
Backup Databases
Backup Management Group
Backup Transaction Logs
Backup Deletion Options
Up-To-Minute Backup Informati
Verify Backup and Transaction
Verification Settings
Select Replica
Help
< Back
Next >
Cancel

Select a remote instance of SQL Server belonging to the AlwaysOn cluster to host the secondary replica. SnapManager will create a clone database on the specified remote instance based on a snapshot backup created on the connected server

Secondary Replica: DONALD\DONALD_NI1

DONALD\DONALD_NI1
JERRY
JERRY\JERRY_NI1
JERRY\JERRY_NI2
LAUREL\LAUREL_NI1

Note: Since a clone database cannot be used on the primary replica of a particular availability group, primary replica is not included in the list for selection.

Note: This option is disabled if the Availability Group is on an SQL Server failover cluster instance.

Note: There can be up to two automatic failover replicas and up to three synchronous commit replicas.

Best Practice: Replica

NetApp recommends using the Replica Wizard because Snapshot based replication offers faster creation of the availability replica by reducing uptime for secondary replicas as compared to native SQL Server.

Because Snapshot copies are taken from live databases on NetApp storage, the lag between primary and secondary clone replication is less.

PowerShell Cmdlet Enhancements for AlwaysOn

As in previous releases, most of the new features have been extended with Windows PowerShell cmdlets and switches. To support Availability Group backup in Windows PowerShell, the following new switches have been added to the new-backup and verify-backup cmdlets:

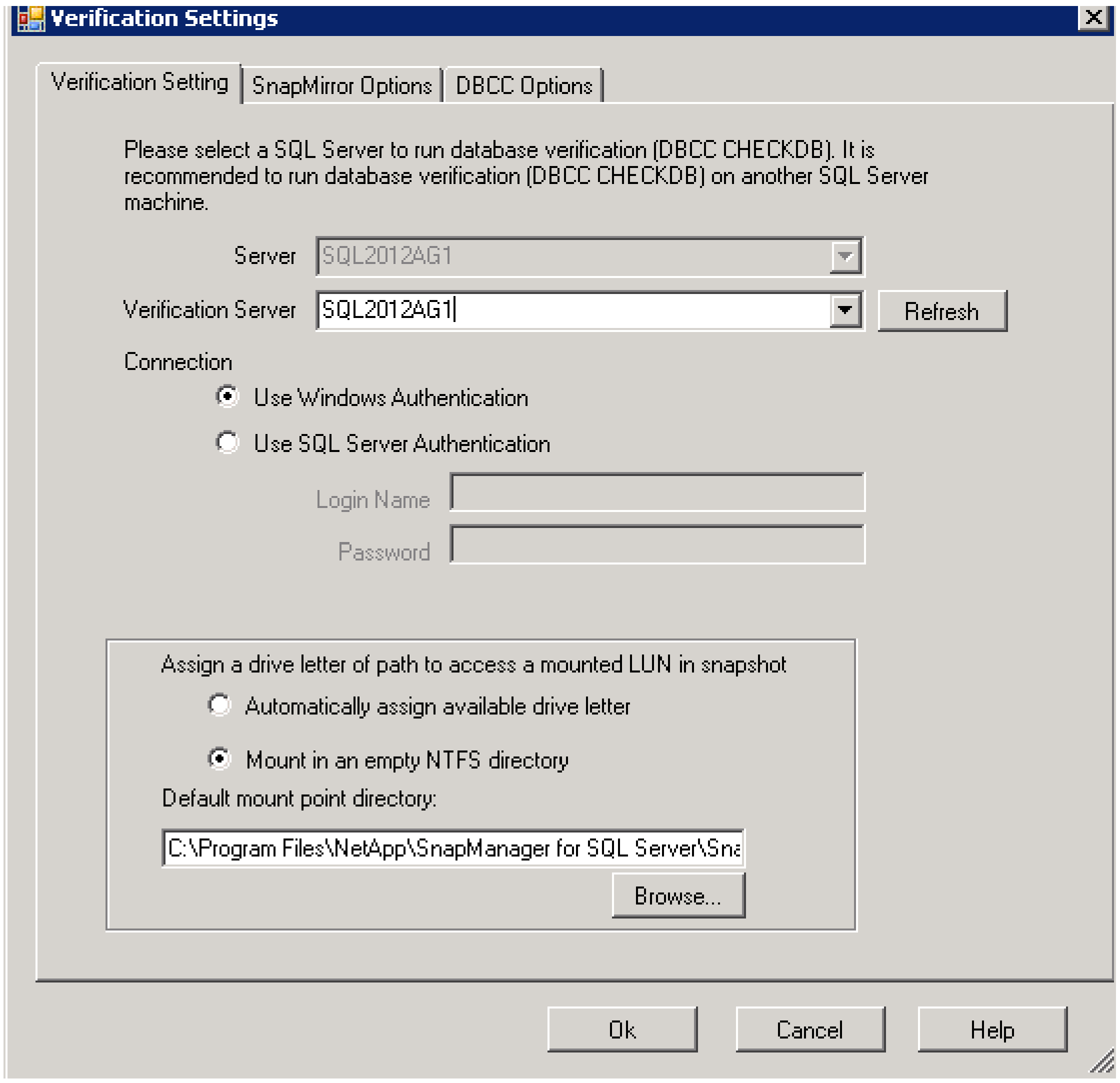
- -AvailabilityGroup: A parameter to define AG names.
- -PreferBackupReplica switch with new-backup: A switch to allow backup of only one replica, as suggested (or logic provided) by Microsoft.
- -BackupPriority switch: Based on priority, you can take a backup among multiple replicas. Use this switch with parameters to define the Min, Max range.

- -Primary: If this switch is defined, backup is taken only on the primary replica.
- -Secondary: If this switch is defined, backup is taken on all secondary replicas.
- restore-backup –svr <ServerName> -inst <SQLInstance> –ag <AGName> -backup <BackupName>
- reseed-backup –svr <ServerName> -inst <SQLInstance> –ag <AGName> -backup <BackupName>
- clone-replica -svr <SERVERNAME> -inst <INSTANCENAME> -ag <AGNAME> -tgInst <TARGETINSTANCENAME> -SychronousCommit -FailoverMode

Server-Based Verification

This version of SMSQL moves from instance-based verification to server-based verification. In previous releases, if the verification setting was changed in one instance, the settings in other instances on that server also changed; now other instances in the same server are not affected.

By default, SMSQL picks the last verification server on any instance of the server. NetApp recommends using a shared mount point if you are setting verification for SQL Server FCI. It is also advisable to set a remote server for verification if there are OLTP databases on the host in a production environment.



Please select a SQL Server to run database verification (DBCC CHECKDB). It is recommended to run database verification (DBCC CHECKDB) on another SQL Server machine.

Server: SQL2012AG1

Verification Server: SQL2012AG1

Connection

[✓ Use Windows Authentication]

Use SQL Server Authentication

Login Name:

Password:

Assign a drive letter of path to access a mounted LUN in snapshot

Automatically assign available drive letter

[✓ Mount in an empty NTFS directory]

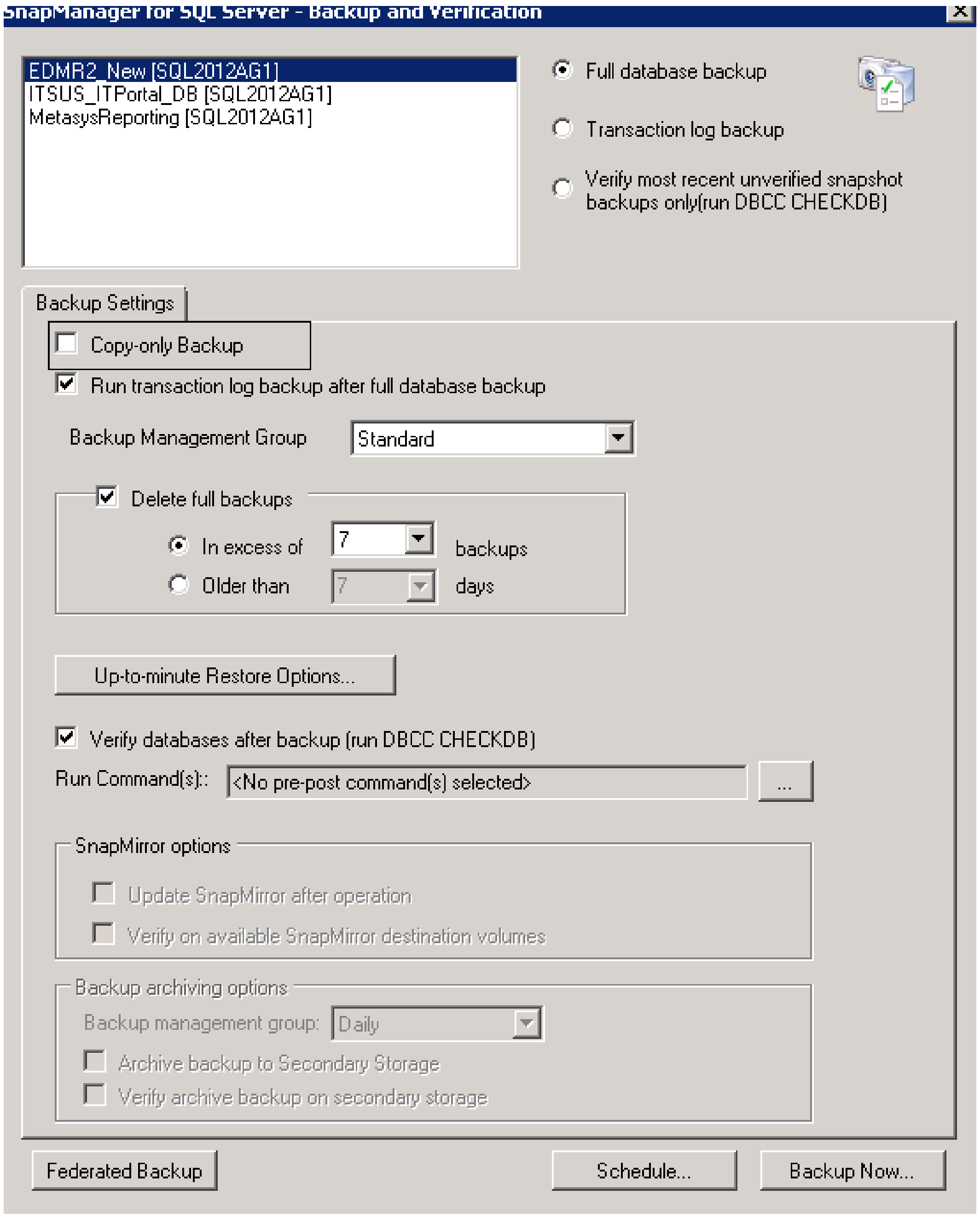
Default mount point directory:

C:\Program Files\NetApp\SnapManager for SQL Server\Sna

Copy-Only Backup

A copy-only backup is a SQL Server backup that is independent of the sequence of other backups. Taking a backup changes the database and affects how later backups are restored. Copy-only backup is useful for taking a backup for a special purpose without affecting the overall backup and restore procedures for the database. It is sometimes useful for performing an online restore.

There is no requirement to create a log backup with NO-RECOVERY and use it with previous log backups for the restore sequence. Copy-only backup is an alternative to this procedure. This feature has been integrated as a backup option in SMSQL 6.0.



SnapManager for SQL Server - Backup and Verification

EDMR2_New [SQL2012AG1]
ITSUS_ITPortal_DB [SQL2012AG1]
MetasysReporting [SQL2012AG1]

(●) Full database backup
( ) Transaction log backup
( ) Verify most recent unverified snapshot backups only(run DBCC CHECKDB)

Backup Settings

☐ Copy-only Backup
☑ Run transaction log backup after full database backup

Backup Management Group: Standard

☑ Delete full backups
(●) In excess of 7 backups
( ) Older than 7 days

Up-to-minute Restore Options...

☑ Verify databases after backup (run DBCC CHECKDB)

Run Command(s):: <No pre-post command(s) selected> ...

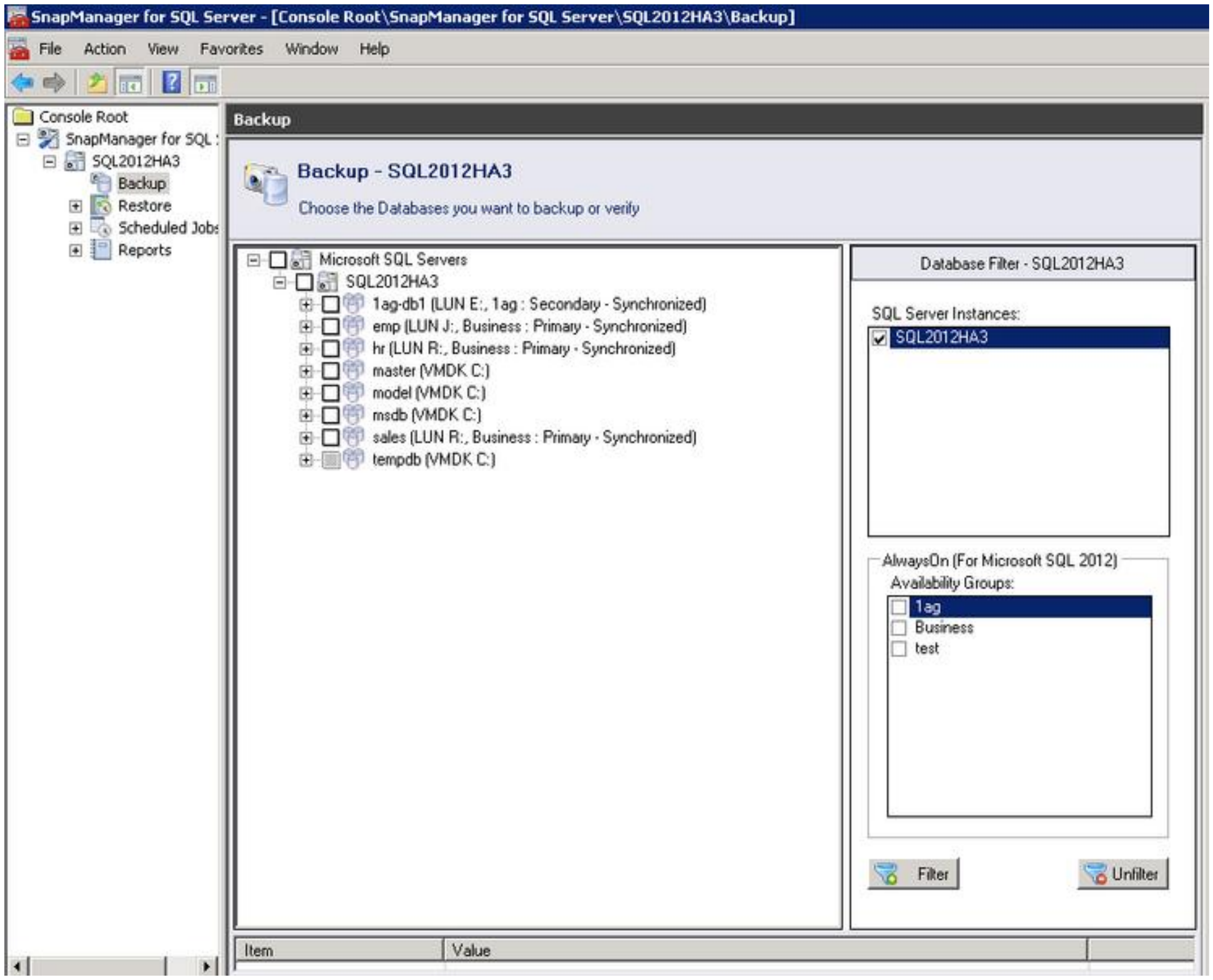
SnapMirror options
☐ Update SnapMirror after operation
☐ Verify on available SnapMirror destination volumes

Backup archiving options
Backup management group: Daily
☐ Archive backup to Secondary Storage
☐ Verify archive backup on secondary storage

Federated Backup | Schedule... | Backup Now...

Database Backup Filter

A new user interface panel, the database backup filter, has been added to SMSQL 6.0. This simplifies the use of the filter backup based on Availability Group parameters, which includes Preferred Replica, Replica Option, and Backup Priority Range. NetApp recommends using this filter from SMSQL MMC for manageability of Availability Groups.


SnapManager for SQL Server - [Console Root\SnapManager for SQL Server\SQL2012HA3\Backup]
File Action View Favorites Window Help
Console Root
SnapManager for SQL :
SQL2012HA3
Backup
Restore
Scheduled Jobs
Reports
Backup
Backup - SQL2012HA3
Choose the Databases you want to backup or verify
Database Filter - SQL2012HA3
SQL Server Instances:
AlwaysOn (For Microsoft SQL 2012)
Availability Groups:
Filter
Unfilter
Item
Value

Microsoft SQL Servers
- SQL2012HA3
  - 1ag-db1 (LUN E:, 1ag : Secondary - Synchronized)
  - emp (LUN J:, Business : Primary - Synchronized)
  - hr (LUN R:, Business : Primary - Synchronized)
  - master (VMDK C:)
  - model (VMDK C:)
  - msdb (VMDK C:)
  - sales (LUN R:, Business : Primary - Synchronized)
  - tempdb (VMDK C:)

SQL Server Instances: [✓] SQL2012HA3

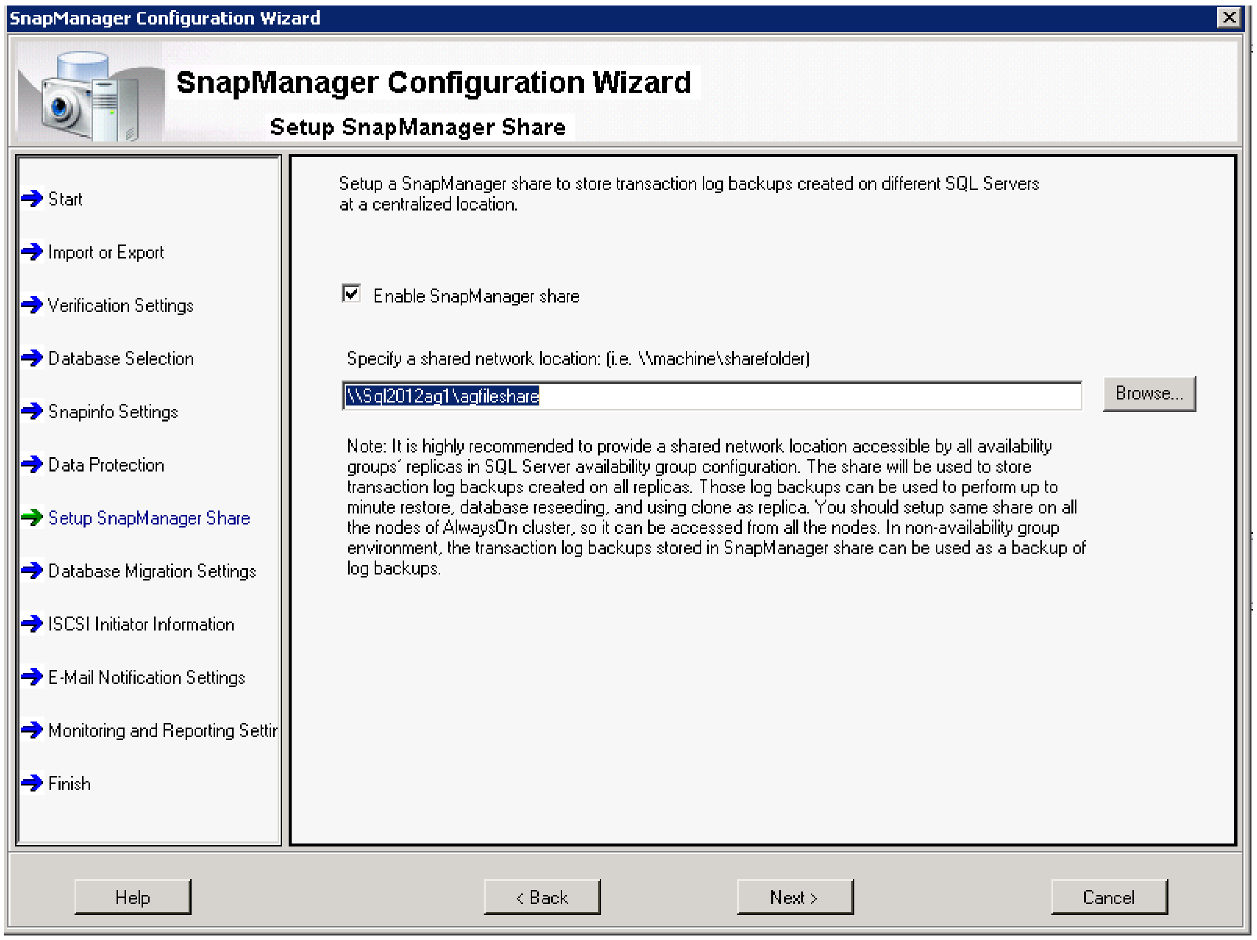
Availability Groups: [ ] 1ag, [ ] Business, [ ] test

SnapManager Share

A central repository share location is maintained to gather backup logs from all replicas of Availability Group databases. During the time of backup, log backups are copied to this share from all nodes. These log backups are gathered, and a continuous sequence of log backups (log chain) is formed during restore.

NetApp highly recommends setting the repository share. In Backup Settings, there are options for share retention and copying. You must create and configure a share for the Availability Groups' log backups in order to manage AlwaysOn Availability Groups.

You should configure a SnapManager share for all the replicas that have read/write permissions in order for the SMSQL service to store the log file backups for the Reseed Wizard to work. Also, you should select the Enable SnapManager Share checkbox in the Setup SnapManager Share page in the SMSQL Configuration Wizard.


SnapManager Configuration Wizard
SnapManager Configuration Wizard
Setup SnapManager Share
Start
Import or Export
Verification Settings
Database Selection
Snapinfo Settings
Data Protection
Setup SnapManager Share
Database Migration Settings
ISCSI Initiator Information
E-Mail Notification Settings
Monitoring and Reporting Settir
Finish
Browse...
Help
< Back
Next >
Cancel

Setup a SnapManager share to store transaction log backups created on different SQL Servers at a centralized location.

☑ Enable SnapManager share

Specify a shared network location: (i.e. \\machine\sharefolder)

\\Sql2012ag1\agfileshare

Note: It is highly recommended to provide a shared network location accessible by all availability groups' replicas in SQL Server availability group configuration. The share will be used to store transaction log backups created on all replicas. Those log backups can be used to perform up to minute restore, database reseeding, and using clone as replica. You should setup same share on all the nodes of AlwaysOn cluster, so it can be accessed from all the nodes. In non-availability group environment, the transaction log backups stored in SnapManager share can be used as a backup of log backups.

Performance Improvements

Some key performance improvements to SMSQL have been made in this release, including:

- SMSQL Snapshot cache
- On-demand lazy load enumeration of SMSQL backups
- Grouping databases into larger Snapshot groups
- Batching I/O into larger transfer requests
- SnapDrive cache for mount and dismount operations

SMSQL Snapshot Cache Performance

When enumerating Snapshot copies for each LUN, a relatively expensive call is made by SMSQL to SnapDrive, and, ultimately, to the NetApp storage controller. This feature caches the results of previous calls to query the NetApp storage for the existing Snapshot copies. The cache does not persist for the lifetime of the SMSQL service.

The operations that use cache are:

- Delete phase of backup
- Restore enumeration of existing backups
- Delete individual dialog load
- Cmdlet enumeration of existing backups

The registry key to enable or disable the SMSQL cache on the SMSQL server:

```
 Key: HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance\SnapManager for SQL Server\Server
ValueName: EnableEnumSnapshotCache
REG_DWORD:
0 = disable cache
1 = enable SMSQL lun snapshot cache (default
```

On-Demand Lazy Load Restore

This feature helps to enumerate backups for restore on an as-needed basis. It preemptively stages the enumeration so the entire set of backups is not loaded at once.

The registry key to enable/disable on-demand (deferred or lazy) restore in the GUI:

```
Key: HKLM\SOFTWARE\Network Appliance\SnapManager for SQL Server\Client
ValueName: EnableOnDemandRestoreTree
REG_DWORD:
0 = disable and load all backups all the time (sme -6.0 behavior),
1 = enable loading of backup detail only on as needed basis, lazy load on demand (default)
```

Backup Grouping Performance

This feature groups databases that share storage into larger groups to be processed as a single backup set (or subjob) of the entire backup job. The one law of backup that must be understood is that Snapshot copies are currently always made at the storage volume level and the volume can have many qtrees, LUNs, and, of course, file system objects.

The advantages of the grouping:

- This new grouping by default reduces the per-group overhead (of VDI, NetApp, Windows, and so forth) for each backup subjob.
- This also allows more NetApp volumes to be copied by Snapshot concurrently with increased parallel processing.
- This new grouping improves performance for databases that aren't already on a single NetApp storage volume.

Default Limits

- Minimum databases per backup set = 35
- Maximum databases per backup set = 255

This feature groups databases into a minimum of 35 databases in a backup set. If each database is located on a separate NetApp volume, then after 35 databases it will commit the backup set to execute. If the combined number of databases exceeds the maximum, this feature will commit the set as a backup set for the overall current backup.

The registry key that can be configured to modify the minimum and maximum number of volumes to combine:

```
KEY: HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance\SnapManager for SQL Server\Server
TYPE: REG_DWORD
VALUE: "MaxDbConcurrentBackup" = default is 255
VALIE: "MinDbConcurrentBackup" = default is 35
```

The registry key to enable or disable the SMSQL grouping:

```
Key: HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance\SnapManager for SQL Server\Server
ValueName: EnableBackupSetGrouping
REG_DWORD:
0 = disable the multi-volume backup set grouping, reverts backup sets based on volume boundaries.
1 = enable the multi-volume backup set grouping, (default)
```

Batching I/O

This feature helps to batch more enumeration requests into a single call to SnapDrive for Windows.

This improvement is noticeable in configurations in which there is a high number of LUNs and databases. It is also effective when there are many outstanding backups on the hard drives. If only one or two backups exist the improvement is less noticeable. This reduces the per SnapDrive for Windows calls and batches the storage controller I/O for several LUNs into a single larger call, thus reducing the per-call overhead.

However, this happens only on the Delete All Backups dialog load in the SMSQL MMC client.

SnapDrive for Windows Cache

This feature is in the server side only. This improvement is noticeable in the enumeration of LUNs within SnapDrive and is meant primarily to improve verification speeds.

The registry key to enable or disable the SnapDrive cache on the SMSQL server calls into SnapDrive:

```
Key: HKEY_LOCAL_MACHINE\SOFTWARE\Network Appliance\SnapManager for SQL Server\Server
ValueName: DisableSDLunEnumCache
REG_DWORD:
0 = enable the SnapDrive cache (default)
1 = disable the SnapDrive cache
```

Remote Backup Verification with SMSQL

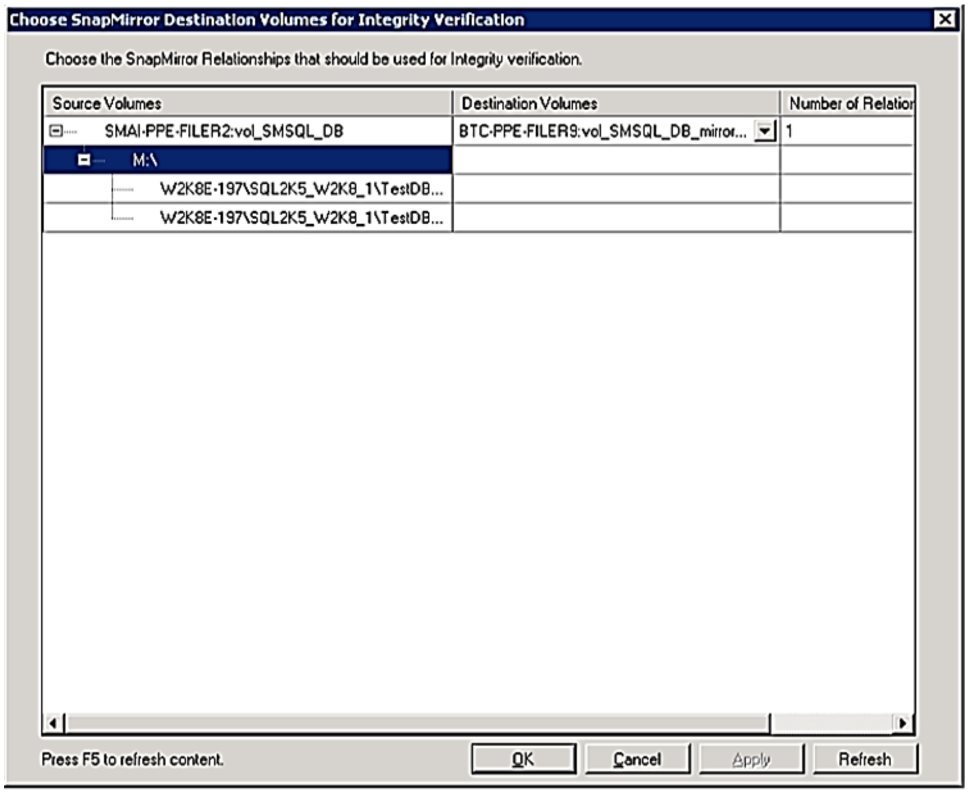
SMSQL introduced an important feature with respect to backup verification. It now supports the verification of backup sets at SnapMirror and SnapVault destinations without breaking the relationships.

The prerequisites for enabling SMSQL 5.0 to be able to verify backup sets at the SnapMirror destination are:

1. Data ONTAP version 7.1 or newer
2. SnapMirror license on the source storage controller
3. FlexClone license on the destination storage controllers
4. Enough disk space on the destination storage controller to create a FlexClone volume
5. On the SQL Server host:
    a. Database(s) placed in volume(s) with SnapMirror relationships; multiple destinations are OK
    b. SnapDrive 6.0.1

TIP: In case the source database volumes have multiple SnapMirror relationships, SMSQL verifies the first default SnapMirror destination that is returned by SDW.

TIP: SMSQL 5.0 provides a Details dialog box that shows all the information such as source volume, destination volume, and other details of a SnapMirror relationship.


Choose SnapMirror Destination Volumes for Integrity Verification

Choose the SnapMirror Relationships that should be used for Integrity verification.

Source Volumes	Destination Volumes	Number of Relatio
SMAI-PPE-FILER2:vol_SMSQL_DB	BTC-PPE-FILER9:vol_SMSQL_DB_mirror...	1
M:\		
W2K8E-197\SQL2K5_W2K8_1\TestDB...		
W2K8E-197\SQL2K5_W2K8_1\TestDB...		

Press F5 to refresh content.

TIP: In order to mount the backup for verification, SMSQL first asks SDW to create FlexClone clones of the SnapMirror destinations and mount the LUNs inside them to the verification server.

TIP: The verification server must be kept as a nonproduction SQL Server host. For environments with large databases, it is preferable to have one dedicated SQL Server host as the verification server for each SMSQL instance.

Powershell Integration with SMSQL

SMSQL supports the new SnapManager command line functionality called cmdlet through SnapManager PowerShell. The PowerShell integration provides the administrator in an enterprise environment with room for flexibility while administering large SQL environments using SMSQL.

The cmdlets that are currently available with SMSQL are:

- `New-backup`
- `Verify-backup`
- `Restore-backup`
- `Get-backup`
- `Delete-backup`
- `Clone-database`
- `Clone-backup`
- `Delete-config`
- `Import-config`
- `Export-config`

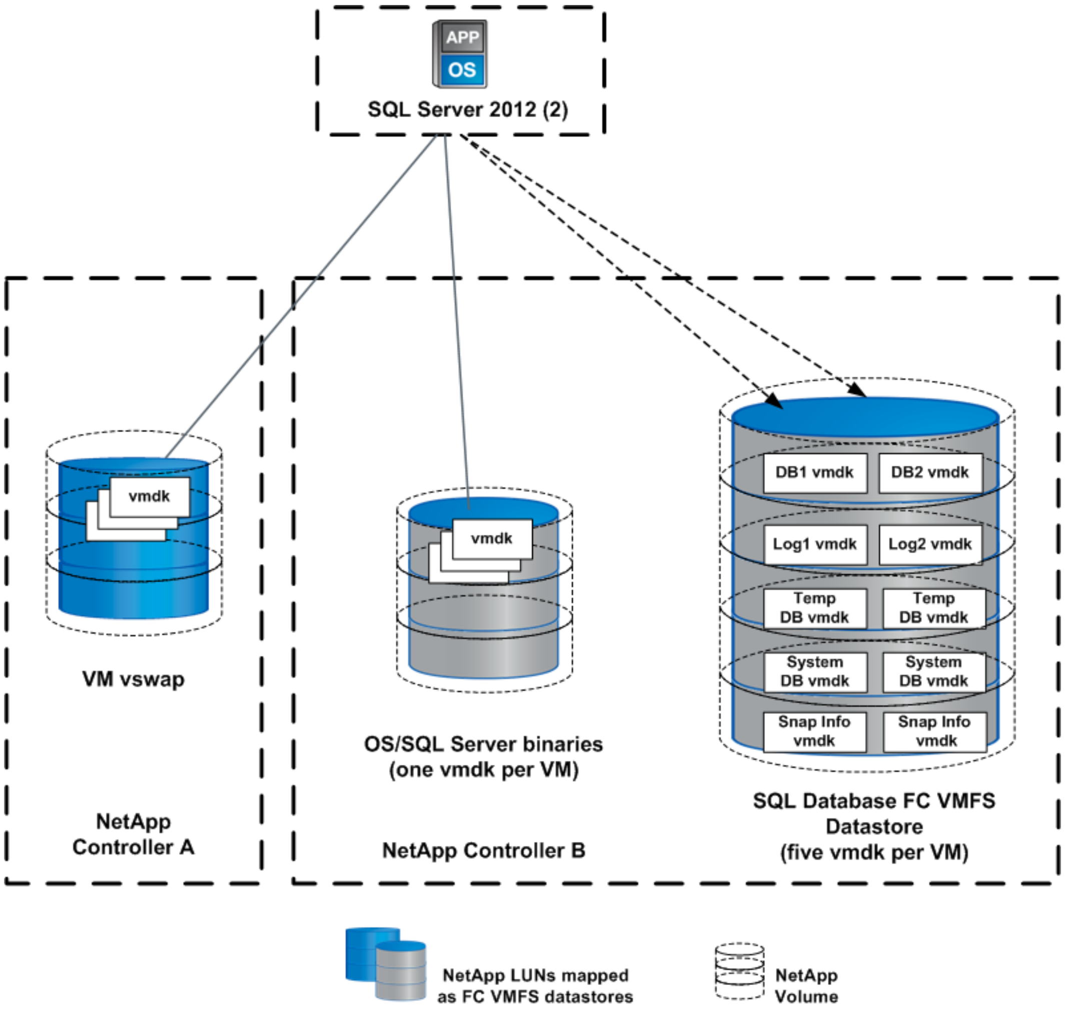
The following script is an example of using a cmdlet for clone automation.

```
*************************************************************************************************
**************
Add-PSSnapin NetApp.SnapManager.SQL.PS.Admin -ErrorAction silentlycontinue

$db = @()
$count = 0
$db_target = @()
$SQL_Target = @()
#$db_lastbkup = @()
$targetHost

$SQL_Target = Read-Host "Enter SQL Server Target for the Database clone."

$string_db = Read-Host "please specify the name of the database to clone.
If you have multiple database names, Please
mention them here as comma seperated values.
for ex: db0,db1,db2"

# formatting the input to be used
$db += ($string_db -split(","))
foreach ($a in $db) {
    if (($a -eq $null) -or (($count -gt 0) -and ($a.trim() -eq $db[$count-1]))) {break}
    else {
        $db[$count] = $a.trim()
        $count += 1
    }
}
# number of databases
$dbcount = $db.Count

for($i=0;$i -lt $count;$i++) {
    $db_target += $db[$i] + "_clone"
}

# get-backup cmdlet to check if the databases have backups. if no backups available, use new-
backup to create a backup
foreach ($database in $db) {
    $db_bkup = get-backup -Database $database | Where {$_.database -eq $database}
    if ($db_bkup -eq $null) {
        $sql_instance = read-host "No backups available. Please provide the sql server instance
for taking a backup"
        new-backup -Database $sql_instance,$dbcount,$database
    }
    #$db_bkup_list = get-backup -Database $database | Where {$_.database -eq $database} | Sort-
Object -Property time_stamp
    #$db_lastbkup += $db_bkup_list[$db_bkup_list.count-1].backup
}

# clone-backup to clone the latest backup
for ($i=0;$i -lt $dbcount; $i++) {
    clone-backup -Database $db[$i] -RestoreLastBackup 0 -TargetServerInstance $SQL_Target -
TargetDatabase $db_target[$i]
}
# wait for 5 minutes before the clones are cleaned up
#Write-Host "waiting for 5 minutes before cleaning up the clones" -ForegroundColor "red"
#Start-Sleep -Seconds 300

# Prompt to continue
Write-Host "Press Enter to continue" -ForegroundColor "red"
Read-Host -Prompt Wait

# delete the clones
$targethost = New-PSSession -ComputerName $SQL_Target
Invoke-Command -Session $targethost -ScriptBlock {Add-PSSnapin NetApp.SnapManager.SQL.PS.Admin -
ErrorAction silentlycontinue}
Invoke-Command -Session $targethost -ScriptBlock {param($SQL_Target,$db_target) delete-clone -
Server $SQL_Target -Database $db_target} -Argumentlist $SQL_Target,$db_target
***************************************************************************
```

SMSQL Support for VMDK

SMSQL provides support to VMDK over NFS and VMFS datastores. Figure 6 explains the layout of SQL Server on a virtualized VMDK environment.

Figure 6) SQL Server on VMDK.



There is a dependency on Virtual Storage Console and SnapDrive for Windows for configuring SnapManager for SQL Server on VMDK.

Note: For VMDK support there is a requirement for SnapDrive for Windows to be version 6.3 or higher.

Note: We must edit the default NFS datastore’s value in order to add more than the eight default datastores. The maximum number is 64.

Note: If the database is on VMDK, it cannot be cloned from destination volumes to a local SQL Server.

The software requirements for the VMDK implementation are specified in Table 4.

Table 4) Software requirements.

Solution Component	Minimum Revision
Primary Storage	
Data ONTAP	8.0.1
NFS, deduplication, FlexClone, SnapMirror, SnapRestore, and NearStore® licenses	N/A
Backup Storage	
Data ONTAP	8.0.1
NFS, deduplication, SnapMirror, and NearStore licenses	N/A
NetApp Management Software	
NetApp Virtual Service Console (VSC)	2.0.1
NetApp SnapManager for SQL Server	5.1
NetApp SnapDrive	6.3
Microsoft SQL Server 2008 R2 Enterprise Edition 64-bit Version 10.50.1734	
VMware vSphere Infrastructure	
ESX hosts	VMware® ESXi 4.1.0 (build 260247)
vCenter™ server	4.1.0
vCenter database	SQL Server 2005
Applications Virtual Machine Operating System	
Windows Server 2008	X64, Enterprise Edition, R2

SnapManager for SQL Configuration Limits

Table 5) Configuration limits.

Configuration Capacity	Maximum Supported with SMSQL 5.1
SQL Server instances per SQL Server computer or Windows cluster	Windows Cluster – 25 Standalone Windows host for SQL Server 2005 and 2008 – 50 SQL Server 2000 – 16
LUNs per SQL Server computer	165
VMDKs per SQL Server computer	56
Databases per LUN or VMDK	500
Databases per storage system volume	500
Databases per SQL Server instance	Virtual instance – 2,500 Standalone instances – 5,500
File groups per database	75

Configuration Capacity	Maximum Supported with SMSQL 5.1
Storage system volumes that can be used to store the following:	A single database – 30 LUNs connected to an individual SQL Server – 165 VMDK connected to an individual SQL Server – 56
Max NFS datastores	64
Fibre Channel LUNs/host	256
Paths to LUN	32
Software iSCSI Initiators	
LUNS/Host	2,563
LUNS concurrently connected	256
Total paths to LUN	1,024

Note: Each instance is not required to be configured or defined to the Microsoft SQL Server. Only the first instance must be defined on the initial build.

Note: The table entries are directories, not mount points.

Note: The task of relocating the database/log must be accomplished by using the SnapManager for SQL configuration wizard.

9 NetApp Disaster Recovery Solution for Microsoft SQL Server

Business continuance or continuity (BC) describes the process and procedures an organization puts in place to make certain that essential functions can continue during and after a disaster. Business continuity planning seeks to prevent disruption of mission-critical services and to reinstate full functioning quickly and efficiently. High availability is a system design protocol and associated implementation that provides a certain degree of operational continuance during a given measurement period.

Business continuity and high availability are not specific technologies. They should integrate a variety of strategies and technologies to address all potential causes of outage, balancing cost against acceptable risk and resulting in a resilient infrastructure. As a first step in business continuity, high-availability planning involves deciding which of the organization’s functions must be available and operational during a crisis. Once the crucial or mission-critical components are identified, it is essential to identify your recovery point objectives and your recovery time objectives for the identified crucial or mission-critical items, apportioning them in terms of cost and acceptable risk. To appropriately architect a disaster recovery solution, one must be familiar with the following terms.

Availability

Generally, this is the degree to which a system, subsystem, service, or equipment is in an operable state for a percentage of time in a functional condition. It refers to the ability of the user community to access the system.

Disaster Recovery (DR)

Disaster recovery is a process of regaining access to the data, hardware, and software necessary to resume critical business operations after a disaster. A disaster recovery plan should also include methods or plans for copying necessary mission-critical data to a recovery site to regain access to such mission-critical data after a disaster.

High Availability (HA)

HA is a system design protocol and associated implementation that provide a certain absolute degree of operational continuity for a system, service, or equipment during a given measurement period. High-availability planning should include strategies to prevent single points of failure that could potentially disrupt the availability of mission-critical business operations.

Recovery Point Objective (RPO)

The recovery point objective describes a point in time to which data must be restored or recovered in order to be acceptable to the organization's process supported by the data.

Recovery Time Objective (RTO)

The recovery time objective is the interface of time and service level within which service availability must be accomplished to avoid undesirable consequences associated with a break in continuity of a service or process.

Service-Level Agreement (SLA)

An SLA is a formally negotiated agreement between a service provider and a user (typically a customer) specifying the levels of availability, serviceability, performance, and operation of a system, service, or application.

9.1 SQL Server Disaster Recovery Options

SQL Server offers many DR, HA, and BC features that vary in their RPO and RTO. These options also vary in their relative complexities and resource needs. The following are the HA/BC features that SQL Server presents.

- **Log shipping.** Log shipping is primarily a BC solution. It involves repeated backing up of a database's transactional log file and its subsequent restoration on a standby version of the database (commonly on a different SQL Server). The backing up and restoration are done based on scheduled jobs on the source and destination servers. Note that log shipping is a per-database BC solution. A failover here must be manual, and the log shipping between the new primary and secondary servers must be reestablished. The connecting applications also must be aware of both servers involved in the log shipping pair.
- **Database mirroring (DBM).** Database mirroring is a full HA/BC solution in SQL Server at the database level. Source and destination SQL Server databases are known as principal and mirror, respectively. A client interacts with the principal and submits a transaction. The principal writes the requested change to the principal transaction log and automatically transfers the information describing the transaction to the mirror, where it is written to the mirror transaction log. The mirror then sends an acknowledgement to the principal. The mirror continuously uses the remote transaction log to "replicate" changes made to the principal database to the mirror database. In case of a disaster the mirror is made live, and new connections to the principal get diverted there automatically.
- **Replication.** Replication may be defined as object-based database mirroring of data on another database or databases. The key idea in replication is that changes made to objects such as tables are replicated to and from participating SQL Servers. It should be noted that replication is per object and provides fine-grained control over what to replicate within a database. Replication may be seen as an HA/BC solution or simply BC, depending on how it is configured. However, the key function of replication is to mitigate changes made from distributed sources to shared database objects.
- **Clustered SQL Server installation.** When SQL Server is installed in a cluster, it presents a true SQL Server-wide HA/BC solution. The failover is SQL Server service specific and saves extra configuration steps for each database as with the previously mentioned HA/BC options.

Table 6 presents a comparison of these options.

Table 6) Recovery options.

Feature	Log Shipping	DBM	Failover Clustering	Replication
Automatic failover	No	Yes	Yes	No
Ease of configuration	Easy	Medium	Hard	Medium-Hard
Granularity of recovery	Database	Database	SQL Server instance	Database object
RPO	Possible data loss	No data loss	No data loss	Some data loss is possible
RTO	Time taken for database recovery	<3 seconds	20–30 seconds plus time taken to recover databases	May run into minutes
Administrative overhead	Minimal	Checking mirror status	Maintaining cluster hardware	May get involved in case of complex publisher-subscriber scenarios

9.2 NetApp DR Solution for MS SQL Server

Based on user experiences, the key advantages of using NetApp solutions to create a DR plan for SQL Server databases include:

- **Ease of configuration.** The most user-friendly aspect of NetApp solutions is the ease with which you can deploy the previously discussed DR plan. Using the SMSQL GUI and a few Data ONTAP commands, you can arrive at a robust DR solution. This reduces administrative overhead in complex database environments.
- **Speed and performance.** Since SMSQL backups are based on volume Snapshot technology (Data ONTAP), the duration for which the database being backed up remains frozen is minimized. This means that for very large OLTP databases, minimal interference with transaction processing occurs. Sync SnapMirror updates are also reasonably fast and provide healthy RPO and RTO figures.
- **Database restoration options.** SMSQL provides two types of database restoration when it comes to restoring a database using transaction log backups: point-in-time restore and up-to-the-minute restore. These modes provide varying degrees of recovery in between full backups in the event of sudden loss of the primary site.
- **Simplified SQL Server–wide DR.** Note that using the simplified SMSQL GUI, an administrator can opt for per-database or system wide SQL Server DR plans. No special or extra steps must be taken for controlling the database(s) to which DR capability must be granted.

Table 7 provides a quick overview of some business problems and how they are addressed on the primary site to provide a resilient architecture.

Table 7) Business problems and NetApp solutions.

Business Problem	Addressed?	How	Description
Single point of failure	✓	Windows cluster + NetApp storage	Windows cluster addressing server resilience and NetApp storage cluster addressing resilience on the storage, providing no single point of failure for applications, server hardware, or storage

Business Problem	Addressed?	How	Description
Fast backup/recovery	✓	SMSQL 2.1	SMSQL automating the complex and manual backup process by creating fast and space-efficient Snapshot copies and providing faster recovery
Disaster recovery	✓	SMSQL + SDW + SnapMirror	SnapMirror replicating the database and log files, and SMSQL providing faster backups and rapid restores
Near-zero-minute RPO	✓	SnapMirror	Scheduled full backups of the SQL Server database replicated every four hours and the T-Log volume replicated synchronously using SnapMirror
Less RTO	✓	SDW/SMSQL	Volume SnapRestore providing an instantaneous restore

9.3 Disaster Recovery Solution for SQL Server User Databases

When architecting disaster recovery solutions for Microsoft SQL Server, it is important to review your current SLAs to derive RPO/RTO objectives.

9.4 Solution Components

SnapDrive for Windows

NetApp SnapDrive for Windows (SDW) is an enterprise-class storage and data management solution for Microsoft Windows Server environments. SnapDrive enables storage and system administrators to quickly and easily manage, map, and migrate data.

NetApp SnapManager for SQL

NetApp SnapManager for SQL Server (SMSQL) speeds and simplifies SQL Server data management. It empowers DBAs to utilize the capabilities of NetApp storage systems through a SQL Server-centric approach. It automates and simplifies the complex, manual, and time-consuming process associated with backup and recovery of SQL Server databases, leveraging the NetApp technology stack to create fast and space-efficient Snapshot copies.

NetApp SnapMirror

NetApp SnapMirror delivers the disaster recovery and data replication solution that today's global enterprises need. By replicating data at high speeds over LANs and WANs, SnapMirror provides the highest possible data availability and fastest recovery for mission-critical applications.

9.5 Disaster Recovery Solution Objective

The primary objective of this disaster recovery solution is to achieve the highest degree of operational continuity at the primary site with no single point of failure and to have a recovery site and replicate the production SQL Server databases for recovery in case of a disaster. Two scenarios were tested with the previously discussed NetApp components to achieve the two different levels of RPOs/RTOs outlined here:

Business Case 1 (Overview)

To meet a near-zero-minute RPO and a five-minute RTO, the data volume and the T-Log volumes were replicated to the DR site synchronously using NetApp SnapMirror.

Business Case 2 (Overview)

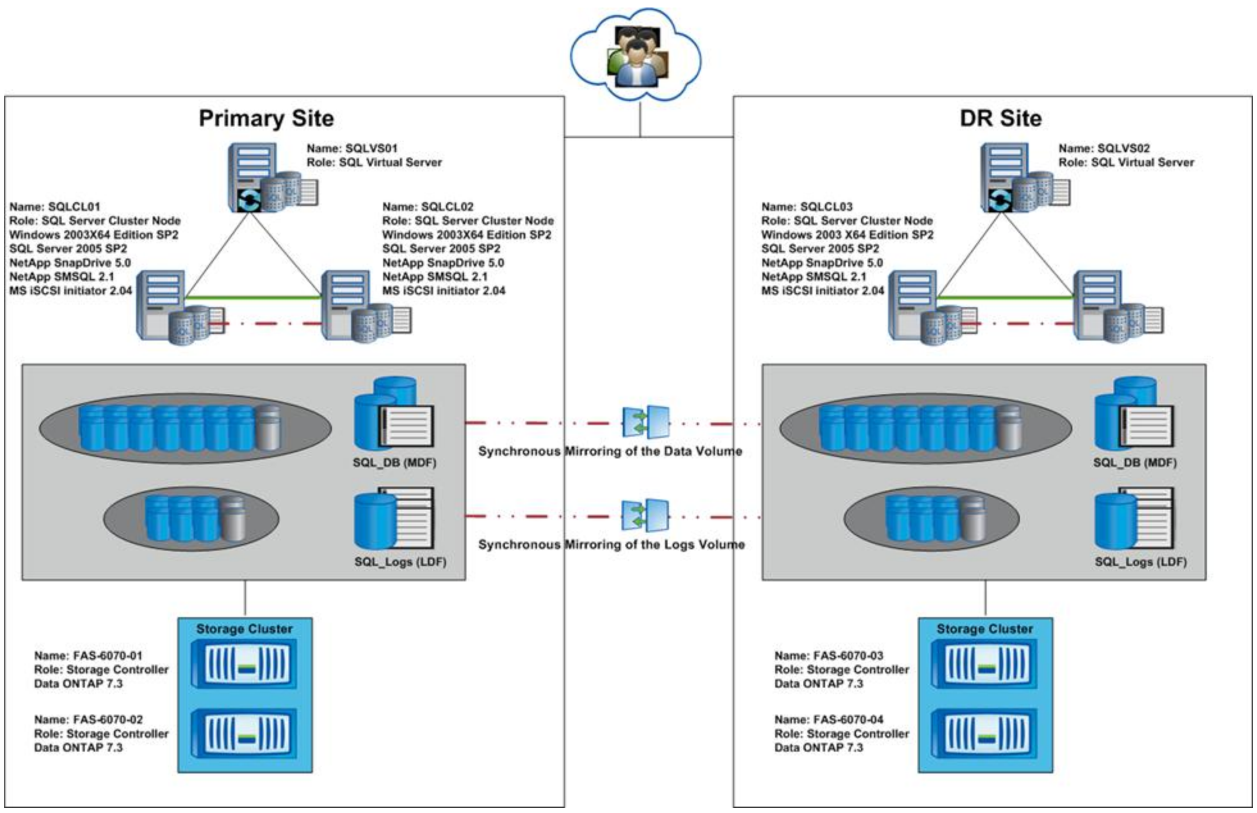
To meet a 15-minute RPO and a 17-minute RTO, SMSQL backups were scheduled and replicated to the DR site every 4 hours, and SnapDrive rolling Snapshot copies of the T-Log volume were replicated to the DR site every 15 minutes using SnapMirror.

9.6 Solution Architecture

The architecture used in this model to provide high availability and disaster recovery for Microsoft SQL Server contained a primary site for production and a DR site for recovery of the SQL Server. For the primary and secondary sites, SQL Server was installed on a two-node Windows cluster on a NetApp storage cluster. Network connectivity was provided between the NetApp storage clusters on the primary and secondary sites (also referred to as the DR site) for replication of data and facilitation of disaster recovery.

Figure 7 shows the basic architecture used in this scenario.

Figure 7) Disaster recovery architecture.



Database Load Generation

All the tests discussed in the following sections used a standard SQL Server database with one MDF and one LDF file. The MDF and LDF files were put in separate volumes. The database was put in FULL recovery mode with the “Recovery Interval” value set to default (0). The initial database size stood at 162GB. For load generation purposes, we used a custom script (see the Load Generation Scripts section in the Appendixes) and generated a load of 54,000 tpm using the OStress utility from Microsoft. The OStress utility may be obtained from http://support.microsoft.com/kb/887057. The growth rate of the database was observed to be around 6 to 7MB/sec.

Note: SMSQL and SnapDrive do not operate concurrently, so it is important to make sure that the two operations do not start at the same time. If the two operations are scheduled to run at the same time, the first operation that is processed will start and the other operation will fail.

10 Microsoft SQL Server Relational Engine: Fundamentals for NetApp Storage

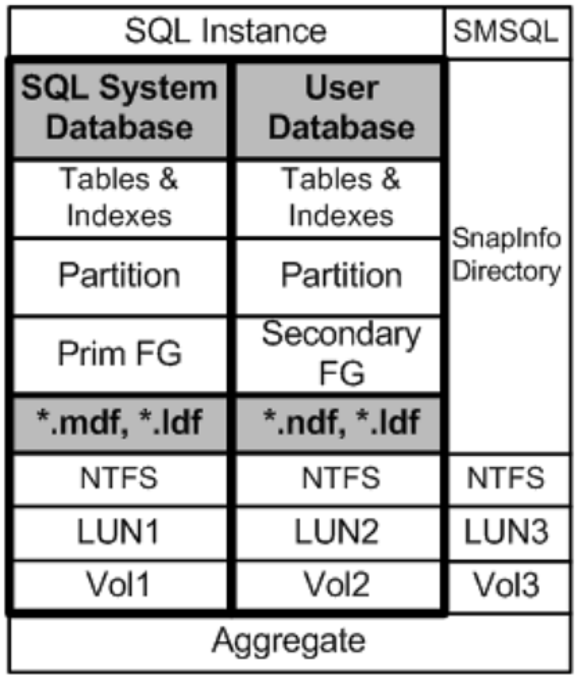
Microsoft SQL Server is a powerful and cost-effective database platform that can be used to meet a variety of enterprise and departmental needs. The layered storage infrastructure of the Microsoft SQL Server (MSSQL) relational engine enables storage design capabilities that were not possible in earlier releases of SQL Server. The combination of NetApp storage solutions and Microsoft SQL Server enable the creation of enterprise-level database storage designs that can meet today's most demanding application requirements. In order to optimize both technologies, it is vital to understand the Microsoft SQL Server relational engine storage architecture.

A good database storage design effectively supports the business requirements defined for the database. The storage design should accommodate the current requirements as well as 12 to 18 months of growth. This enables the initial deployment to be successful and the environment to grow smoothly over time as the business grows. This technical report discusses the SQL Server relational database storage features that can be used to help achieve that goal. When determining which features to implement in your designs, remember to keep the design as simple as possible while utilizing the appropriate features.

10.1 SQL Server Database Storage Introduction

There are two types of databases in SQL Server: system and user. There is no difference in the physical structure between these two database types. Each type at a minimum has data (*.mdf, *.ndf) and transaction log (*.ldf) files. Figure 7 depicts a SQL Server database storage stack starting with the database and ending with the aggregate. The highlighted portions show the database and the physical files that compose it.

Figure 8) SQL Server database storage stack.

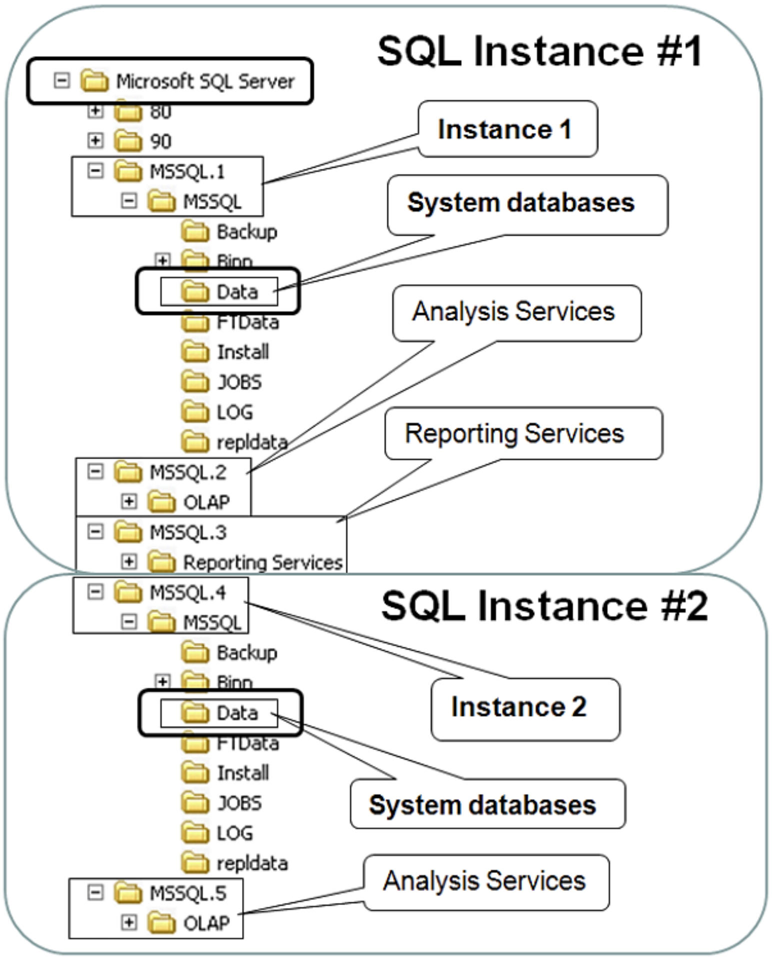


System databases are created each time an instance of SQL Server is installed, or by enabling functions. For example, the distribution database is created when SQL replication is enabled. Multiple instances of SQL Server can be installed on the same Windows Server. Once the SQL instance is installed and running, user databases can be created within each instance.

10.2 SQL Server Directory Structure

Figure 9 shows the directory structure of two SQL instances residing on a single Windows host. The optional SQL Server OLAP services and SQL Server Reporting Services are also installed. The directory structure changes depending on which services are installed. Note that each SQL instance includes the MSSQL directory tree, which contains the respective system databases for that instance.

Figure 9) SQL Server directory structure.

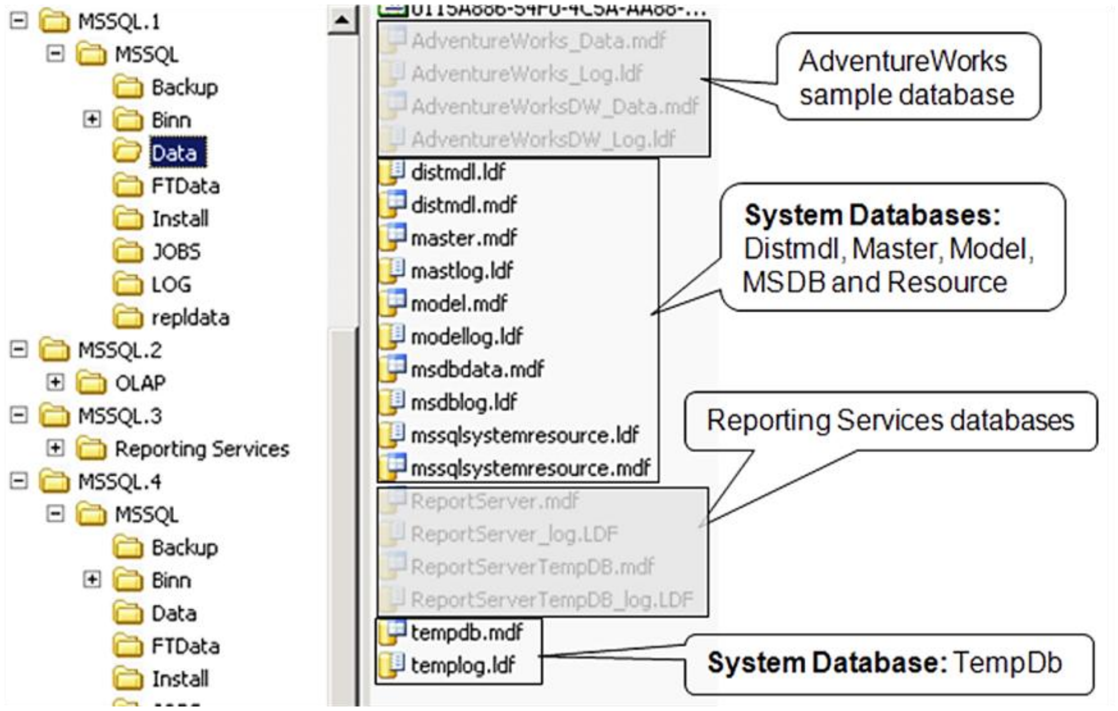


10.3 SQL Server System Databases

By default, the SQL Server system databases for each instance are stored in the MSSQL\Data subdirectory, seen in Figure 9. This is also the default directory for user databases, unless an alternate storage location is specified.

In Figure 10, the contents of the MSSQL\Data subdirectory are shown, highlighting the files comprising the system databases. Each SQL Server database (system or user) has at least one data file (*.mdf) and one transaction log file (*.ldf).

Figure 10) The MSSQL\Data subdirectory contents.



AdventureWorks is a sample user database provided for training and testing. ReportServer is not a system database but is installed if Reporting Services is installed.

Following is a brief description of each system database:

- Master database
  - Contains all the system-level information for the SQL Server instance.
- Model database
  - The template used when creating new user databases; determines attributes such as how large the database will be when the size is not explicitly stated.
- MSDB database
  - Used by the SQL Server Agent for managing jobs and alerts and for other features such as the Service Broker and Database Mail.
- System resource database
  - A read-only database of system objects that logically appear in the sys schema of each database but physically exist in this database.
- Tempdb database
  - Used to hold intermediate results and temporary objects. This database is recreated each time SQL Server restarts.

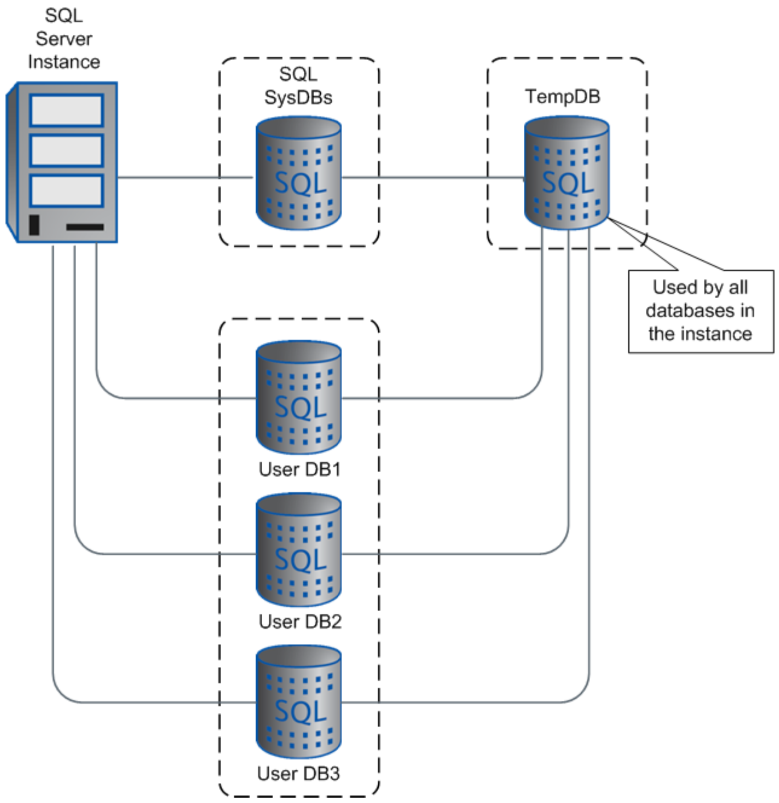
When NetApp SnapManager for SQL Server Configuration Manager is run, any database, system or user, can be migrated to NetApp storage by moving the physical database files. This provides an easy way to migrate databases onto NetApp storage systems.

10.4 A Closer Look at the tempdb System Database

When it comes to storage and the SQL Server system databases, tempdb is the system database with which you should be most familiar. Depending on factors such as query logic and table sizes, tempdb can grow very large and be subject to intense I/O processing. This can lead to poor database performance, and can actually cause tempdb to run out of space.

There is only one tempdb in a SQL Server instance, but it is a global resource used by all the other databases within the instance, including the other system databases. Therefore it is important to understand how each database uses it. Figure 11 illustrates how tempdb is a shared resource.

Figure 11) Tempdb database.



Tempdb, as its name implies, is used to hold temporary information; it is recreated each time SQL Server is restarted. Following are details on how tempdb is used.

- Holds temporary explicitly created user objects such as stored procedures, cursors, temporary tables, and table variables
- Stores SQL Server internal objects such as work tables for intermediate results for spools or sorting and running database verifications
- Stores row versions in databases that use snapshot isolation transactions or transactions that are read committed using row versioning isolation
- Stores row versions that are generated by data modification transactions for features, such as online index operations, Multiple Active Result Sets (MARS), and AFTER triggers

tempdb Utilization and Workload Types

OLTP (On-Line Transaction Processing) is a common workload type. OLTP typically does not make heavy use of tempdb, but makes heavy use of the transaction log. Reporting and DSS (Decision Support Systems)-type workloads, also common workloads, often are heavy users of tempdb, with relatively light use of the transaction log file. Both workload types can coexist on a single storage system with multiple Windows hosts and SQL Server instances.

tempdb and Backups

Tempdb should not be included in a backup since the data it contains is temporary. Place tempdb on a LUN that is in a storage system volume in which Snapshot copies will not be created; otherwise large amounts of valuable Snapshot space could be consumed.

tempdb Space Management

To prevent tempdb from running out of space, the SQL Server `FILEGROWTH` attribute can be used. The `FILEGROWTH` attribute can be specified for any SQL database file. It handles out-of-space conditions by automatically growing the file in the LUN. Note that the LUN must have available space for the file to grow.

`FILEGROWTH` can have a negative effect on storage performance because, when data and log files are created or grown, the new file space is initialized by filling it with zeros. The file growth increment should be set to a reasonable size to avoid either 1) the file constantly growing because the increment is too small, or 2) taking too long to grow because the increment is too large. Microsoft's rule of thumb is to set the increment at 10% for tempdb databases larger than 200MB.

Monitor tempdb file size, disk IOPS, and disk latency. Adjust the size of tempdb (recall that it is recreated each time SQL Server restarts) accordingly to minimize the number of times `FILEGROWTH` triggers. Use `ALTER DATABASE` to change the size. When altered, that will become the new size each time tempdb is recreated. The goal is to eventually set the initial tempdb size so that `FILEGROWTH` rarely, if ever, triggers. Note that this goal may seem counter-intuitive because it defeats `FILEGROWTH`. Consider `FILEGROWTH` to be a fail-safe. If tempdb runs out of space the entire SQL instance might stop functioning.

tempdb and Database Verification

`DBCC CHECKDB` and related statements typically must read each allocated page from disk into memory so that it can be checked. Running `DBCC CHECKDB` when there is already a lot of activity on the system impairs DBCC performance for two reasons. First, less memory is available, and the SQL Server Database Engine is forced to spool some of `DBCC CHECKDB`'s internal data to the tempdb database. Second, `DBCC CHECKDB` tries to optimize the way that it reads data from the disk. If an intensive workload is also using the same disk, optimization is greatly reduced, resulting in slower execution.

Because the tempdb database resides on disk, the bottleneck from I/O operations as data is written to and from disk impairs performance. Regardless of system activity, running `DBCC CHECKDB` against large databases (relative to the size of available memory) causes spooling to the tempdb database.

tempdb Performance Tuning

Tempdb can use more than one data file. By using more than one data file, I/O can be distributed across multiple files. For SQL instances in which tempdb files are heavily used, a general guideline is to create one file per CPU on the host system. This is only a recommendation, and the actual number of files can be more or less than the number of CPUs in the system.

10.5 SQL Server Databases

SQL Server uses 8KB pages to store data (tables and indexes) in the data files (*.mdf, *.ndf) and performs I/O at the page level. The pages are organized within these files in one of two possible ways:

- Heaps: A table without a clustered index. Table rows are not stored in any particular sort order.
- Clustered: A table with a clustered index. Table rows are sorted in order of the clustered index.

Following are some of the key characteristics for each file type:

- Primary data files (*.mdf):

  - Every database has one primary file.
  - Resides in the primary filegroup.
  - Contains database startup information and points to the other files that comprise the database.
  - Can contain user-defined objects, such as tables and indexes.
- Optional secondary data files (*.ndf):
  - A database can have none, one, or many of this file type.
  - Can reside in the primary filegroup, or in optional secondary filegroups.
  - Contains user-defined objects such as tables and indexes.
- Transaction log files (*.ldf):
  - A database can have one to many of this file type.
  - Contains the database transaction log records.

Tables

Tables contain rows and columns of data. A database can have one to many tables. SQL Server tables are stored in one or more database partitions, which are discussed later in this technical report. Be sure not to confuse database partitions with LUN partitions, because they are completely different from each other.

Indexes

Indexes can be created for tables; they help queries run much faster. Tables without indexes are scanned when searching. Indexes can be clustered or nonclustered and the difference between the two is significant.

A clustered index takes the table column to be indexed and sorts all the data rows in the table to the order of this column. The index is integrated with the data and, because of this, clustered indexes always reside in the same data file as the table they support. A table can have either none or one clustered index.

A nonclustered index is stored outside the table in a b-tree format. The actual table rows are not sorted. Nonclustered indexes can reside in the same data file as the table they support, or they can be stored in a separate file. A table can have none, one, or many nonclustered indexes.

Page Splits, FILLFACTOR, and PAD_INDEX

When an index page is full and a new row needs to be added, the index page splits. Split processing creates a new page, and then moves approximately 50% of the rows from the original page to the new page. This creates room inside the original page so new rows can once again be added. With clustered indexes, this split processing occurs against the actual data pages.

Page split processing can negatively affect storage performance and cause file fragmentation. To reduce page splits, a fill factor can be used. The fill factor allocates extra space at the leaf level in each page. This provides extra room in the page to accommodate new rows without causing a page split. `PAD_INDEX` does the same thing as `FILLFACTOR`, but applies the fill factor to the intermediate-level index pages.

Example: Create an index with a fill factor of 80, and pad the intermediate-level index pages also.

```
USE MyDB;
GO
CREATE NONCLUSTERED INDEX IX_MyIndex
ON MyDB.MyTable(MyCol1)
WITH (FILLFACTOR = 80,
PAD_INDEX = ON,
DROP_EXISTING = ON);
GO
```

FILLFACTOR and PAD_INDEX Considerations

Users should understand the following aspects of `FILLFACTOR` and `PAD_INDEX`:

- Their use increases disk space consumption by the additional free space percentage.
- Their use can decrease database read performance since more pages must be retrieved to access the real data.
- They can be applied to new or existing indexes.
- They are only honored when the index is created or rebuilt.
- `FILLFACTOR` affects the leaf pages only.
- The default fill factor is 0.
- `PAD_INDEX` affects the intermediate pages and uses additional disk space.
- `PAD_INDEX` only has an effect if `FILLFACTOR` is greater than 0.

Transaction Log (*.LDF) Files

Each SQL Server database has a transaction log, which is a write-ahead log file. I/O to the log file is always synchronous. Database modifications are first sequentially written in the transaction log. After that the application can commit the transaction to the database or abort the transaction, in which case modifications to the database will not take place and the log records will be discarded.

Each database in SQL Server uses a Recovery Model. The Recovery Model determines the level of logging that takes place for that database. It is a dynamic database attribute so it can be changed on the fly using the `ALTER DATABASE` command. There are three different recovery models:

Simple:

- Use this model when you don’t want to take log backups. Only choose this model when a degree of data loss is acceptable, since you can only recover to the point in time when the last backup was taken.
- No log backups can be taken. SQL Server automatically manages the transaction logs, truncating them with each checkpoint.

Bulk-logged:

- If used, this model is used in conjunction with the Full Recovery Model to increase performance and reduce logging.
- Bulk operations such as `SELECT INTO` and index operations such as `CREATE INDEX` are minimally logged but all other transactions are fully logged the same as with the Full Recovery Model.
- Use `ALTER DATABASE` to dynamically change the recovery model prior to running bulk operations.
- This model requires log backups. If they are not taken, the log file(s) will fill up.
- This does not support point-in-time recovery, so recovery must be up to the last backup.

Full:

- This is the normal model used for production databases.
- It requires log backups. If they are not taken, the log file(s) will fill up.
- This model enables up-to-the-minute restores as well as restores to a specific point in time.
- The transaction log is used for the following:
  - Recovery of individual transactions
  - Roll back of all incomplete transactions
  - Recovery to the point of failure
  - To support transactional replication and standby-server solutions

Use Case

During large data loads, the fill factor can be set high to reduce the number of splits during the load. Additionally, if the database recovery model is normally Full, it can be set to Bulk-logged to greatly reduce the amount of logging. Only use the Bulk-logged recovery model if you are willing to restart the load from the beginning in the event it fails before completion. Performing these two steps can often significantly speed up the load process. After the load is successful, the recovery model can be set back to Full, and the table fill factor reduced to a more normal operating value. The new fill factor takes effect the next time the table is reorganized, which is a good step to perform after a data load.

Transaction Log Utilization and Workload Types

The amount of I/O load on the transaction log depends on the rate of change against the database. For example, OLTP (On-line Transaction Processing) is a common workload type and typically makes a lot of data changes (add, change, delete), which causes heavy use of the transaction log. On the other hand, reporting and DSS type workloads, also common workloads, do mostly reads against the database, so they are relatively light users of the transaction log. Realize that both workload types can concurrently exist, possibly running against the same database.

Multiple Transaction Log Files

Like tempdb, a database transaction log can have more than one physical file. Unlike tempdb, log file access is sequential to only one file at a time. The reason an additional physical file might be added is to increase log file space. For example, if the LUN where the log file resides is full, a second LUN can be created to add extra space, but this would do nothing for performance and the I/O would still be to one file at a time, filling the first file, then moving on to the second file. With the ability to easily expand LUNs with NetApp storage, there is no compelling reason to use more than one physical log file.

Planning and Design Considerations

- When the Full recovery model is used, the transaction log LUN must be large enough to accommodate all log records generated between log backups.
- The longer the time between log backups, the larger the transaction log grows.
- The larger the transaction log, the longer it takes to perform backups and recoveries.
- When using SnapManager for SQL Server, the SnapInfo partition/directory must be large enough to hold the maximum number of log backups that will be retained. This is determined when identifying the RTO and RPO for the database.
- The transaction log is sensitive to I/O delays, so it must be backed by enough physical spindles to support the transactional I/O demands.

Monitor and Manage

- Determine that the SnapInfo partition maintains enough space to hold all log backups. SMSQL backups can be configured to maintain the number of backups desired.
- Schedule log backups to keep the log file from filling up.
- Monitor `FILEGROWTH`. Adjust the file size as needed to prevent `FILEGROWTH` from triggering.

10.6 Filegroups

Up to this point the SQL directory structure; database files, including logs, and tables and indexes have been discussed. This section explores filegroups.

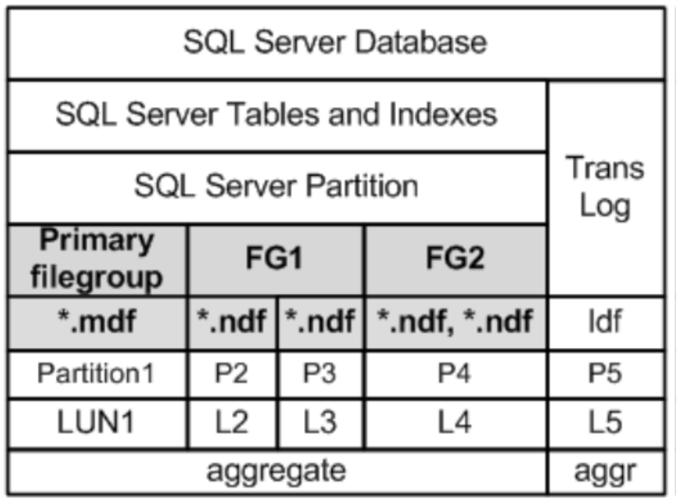
Filegroup Overview

Filegroups are a core component in the SQL Server database storage architecture. Filegroups are a logical grouping of physical files (*.mdf, *.ndf) and are used to organize database files in a way that optimizes storage space and performance. When a filegroup contains more than one file, all the files within the group should be the same size.

The following SQL Server database storage stack (Figure 12) highlights three filegroups and the files they each contain.

Note: A file cannot be defined in more than one filegroup.

Figure 12) Database filegroups and files.



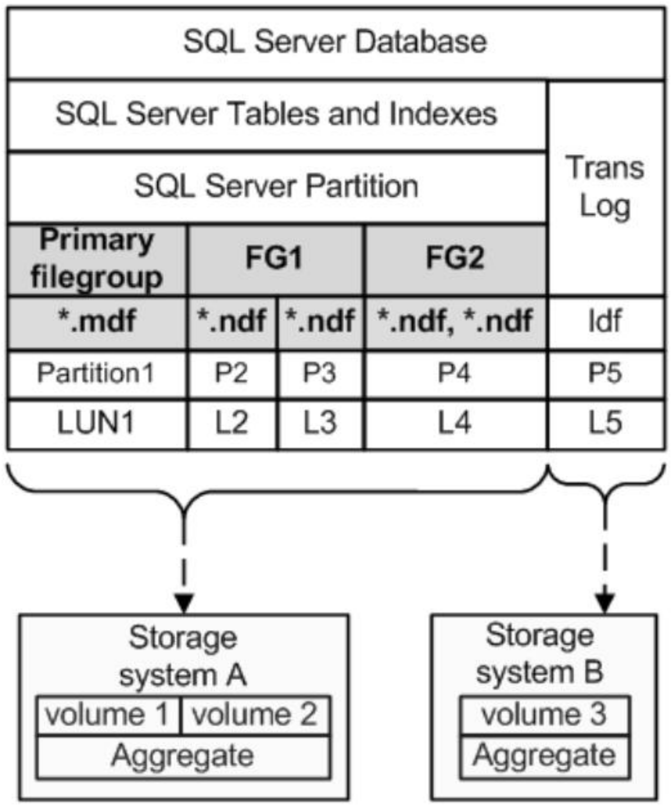
On multiprocessor servers, filegroups help take advantage of parallel I/O processing by distributing LUNs across the processors. The Microsoft recommendation on multiprocessor servers is to have at least one LUN per processor. For more information, read about the affinity I/O mask in the SQL Server Books Online.

Characteristics of the design in Figure 12 include (excluding the logs):

- Primary filegroup:
  - Contains the *.mdf file where the system databases reside
  - Resides on its own LUN partition, LUN, and storage system volume
  - Shares the same aggregate as the other storage system volumes
- Filegroup 1 (FG1):
  - Contains two secondary (*.ndf) files
  - Each file has its own partition and LUN
  - Shares a volume with filegroup 3
  - Shares the same aggregate with all the other volumes
- Filegroup 2 (FG2):
  - Contains two secondary (*.ndf) files
  - Both files share the same partition and LUN
  - Shares a volume with filegroup 4
  - Shares the same aggregate with all the other volumes

Filegroups facilitate placing database files on different storage systems. The level of granularity enables placing a single table, index, or log on a dedicated LUN, volume, or even aggregate. Figure 13 shows the same storage design shown in Figure 12; however, it uses two NetApp storage systems, with the data files placed on one storage system and the log files placed on the other.

Figure 13) Database filegroups and files.



How SQL Server Performs I/O with Multiple Files in a Filegroup

SQL Server uses a proportional fill strategy to write data across multiple files within a filegroup. When a filegroup does have multiple files and all the files in the group are the same size, I/O to all files within the group is uniform. When a new file is added to an existing filegroup that already contains populated tables, the I/O is not initially uniform. Instead, SQL Server writes more data to the new file in proportion to the other files, until space in the new file is consumed to the same extent already consumed in the other files within the group. After that time, I/O again becomes uniform across all the files in the filegroup.

Filegroup Types

There are two types of filegroups: primary and user-defined. Following are some of the characteristics of each.

- Primary
  - There is one per database, created when the database is created.
  - The SQL system tables reside here, in the *.mdf file.
  - It contains any other files not allocated to another filegroup.
- User-defined
  - Each database can have none, one, or many of these.

- It can contain one or many data files (*.ndf).
- It contains user database tables and indexes.

The Default Filegroup

One filegroup in each database can be designated as the default filegroup. This way, when a table or index is created for that database and a filegroup is not specified, it is still defined in the intended set of physical files. If a default filegroup is not specified, the primary filegroup is the default filegroup.

If the initial default filegroup (primary) is never changed and you never explicitly specify a different filegroup name, then when you create new databases, tables, and so on, they are created in the files in the primary filegroup. The net effect is that the SQL System databases and the user-defined databases will reside on the same physical files. It is a general SQL Server best practice to change the default filegroup so this does not occur. This is done for performance and availability reasons.

The `ALTER DATABASE` command can be used to specify the default filegroup. In the following example, a default filegroup is specified for database "MyDB."

Change the default filegroup:

```
ALTER DATABASE MyDB
MODIFY FILEGROUP MyDB_FG1 DEFAULT
GO
```

Note: When SnapManager for SQL Server is used to migrate the SQL Server system databases to NetApp LUNs, the primary filegroup will point to the NetApp LUNs.

Read-Only Filegroups

User-defined filegroups can be marked read-only so the data within the filegroup cannot be changed. A common use of read-only filegroups is for storing online historical data. Provided the storage requirements are maintained, further storage savings can be obtained by placing the filegroup(s) on less expensive disks.

Filegroups and Piecemeal Restores

It is good to be aware of piecemeal restores for two reasons:

1. NetApp storage systems enable SQL Server DBAs to avoid this complex procedure.
2. In SQL Server 2005, a database can be brought online after restoring just the primary filegroup, without restoring any of the other filegroups.

With Microsoft SQL Server 2005, databases with multiple filegroups can be restored and recovered in stages through a process known as piecemeal restore. Piecemeal restore involves first restoring the primary filegroup, after which the database can be brought online, and then restoring one or more secondary filegroups.

A sample use case would be to store mission-critical data in a select subset of filegroups. These could be managed differently; for example, they could reside on more expensive and highly protected storage. Less critical data could be stored in other secondary filegroups, perhaps on less expensive storage. In a recovery scenario, the mission-critical filegroups would be restored first so the database and related critical applications using that database could be brought online. Then the less critical filegroups could be restored in the background.

As noted, with NetApp Snapshot technology, piecemeal restores are probably unnecessary because a restore from a Snapshot copy is so fast that the entire database can be brought back very quickly.

Creating and Managing Filegroups

Following are examples of common filegroup-related management tasks.

Add a new filegroup to an existing database:

```
USE master
GO
ALTER DATABASE MyDB
ADD FILEGROUP MyDB_FG2
GO
```

Add a file to the filegroup just created in the preceding example:

```
USE master
GO
ALTER DATABASE MyDB
ADD FILE
( NAME = MyDBdat2,
FILENAME = 'g:\MyDB_data2\MyDBdat2.ndf',
SIZE = 25GB,
MAXSIZE = 100GB,
FILEGROWTH = 15GB )
TO FILEGROUP MyDB_FG2
```

Resize an existing file in a filegroup:

```
USE master
GO
ALTER DATABASE MyDB
MODIFY FILE
(NAME = MyDBdat2,
SIZE = 200GB)
GO
```

Remove the file just added:

```
USE master
GO
ALTER DATABASE MyDB
REMOVE FILE MyDBdat2
GO
```

Change the default filegroup:

```
ALTER DATABASE MyDB
MODIFY FILEGROUP MyDB_FG1 DEFAULT
GO
```

Make the primary filegroup the default filegroup:

```
USE master
GO
ALTER DATABASE MyDB
MODIFY FILEGROUP [PRIMARY] DEFAULT
GO
```

Delete a filegroup from a database (all files must be removed first):

```
USE master
GO
ALTER DATABASE MyDB REMOVE FILEGROUP MyDB_FG1
GO
```

Placing Indexes on Dedicated Storage

By default, all indexes reside in the same data files as the tables they support. Alternatively, nonclustered indexes can be placed in their own files. This provides the ability to place nonclustered indexes on their own LUN or even volume, which may be desirable for performance purposes. Following is an example of how to place an index in its own file in a filegroup.

In this example a new filegroup is added to the “MyDB” database. This filegroup contains a new file with an extension of .idx. That extension name helps identify it as a file that contains indexes. The index is then created on that file.

To place an index on its own file, perform the following steps:

1. To use a dedicated LUN, present it to the host and create a partition. NetApp recommends using SnapDrive for this step because it automatically handles creating the LUN and mapping it to the host, as well as creating a properly aligned partition and formatting it. SnapDrive supports volume mount points as well as drive letters.
2. Create a new filegroup called MyDB_EX_fg1:

```
USE master
GO
ALTER DATABASE MyDB
ADD FILEGROUP MyDB_IX_fg1
GO
```

3. Add the new file to the filegroup just created:

```
ALTER DATABASE MyDB
ADD FILE
( NAME = MyDB_IX_fg1,
FILENAME = 'g:\MyDB_ix_fg1\MyDB_IX1.idx',
SIZE = 5MB,
MAXSIZE = 10MB,
FILEGROWTH = 1MB )
TO FILEGROUP MyDB_IX_fg1
```

4. Create a new index on the ErrorLog table, and place it in the new filegroup:

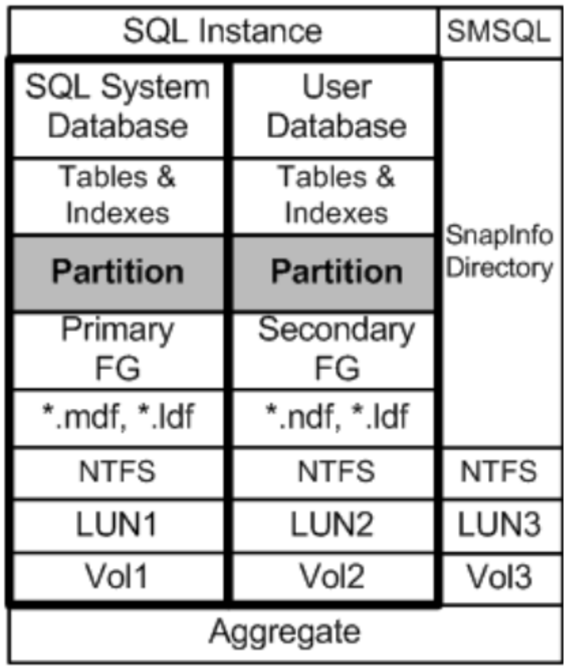
```
CREATE NONCLUSTERED INDEX IX_username
ON dbo.ErrorLog (UserName)
ON MyDB_IX_fg1;
Go
```

The new index now resides in its own file, rather than in the same file in which the table resides.

10.7 An Introduction to SQL Server Table and Index Partitioning

This section examines SQL Server table partitions, how they can be used to enhance applications, and their storage implications.

Figure 14) Table and index partitions in the SQL Server database storage stack.



Starting in SQL Server 2005, partitions form the base physical organizational unit for tables and indexes; every table and index page is stored in a partition. A table or index with a single partition, as illustrated in Figure 14, is equivalent to the organizational structure of tables and indexes in earlier versions of SQL Server.

Partitions allow the data rows stored in one table to be split and stored in multiple smaller tables, called partitions. After a table or index is partitioned, you still use the same name to reference it. For example, the Employee table is still referenced by Employee, whether it has one or multiple partitions.

There are two types of table partitioning: vertical and horizontal. Vertical partitioning splits a table by columns so that different columns reside in different partitions. Horizontal partitioning splits a table by rows, so that different rows reside in different partitions. This paper discusses horizontal partitioning only.

Note: Partitioned tables and indexes are only available in the Enterprise and Developer editions of SQL Server 2005.

Why Use Partitions?

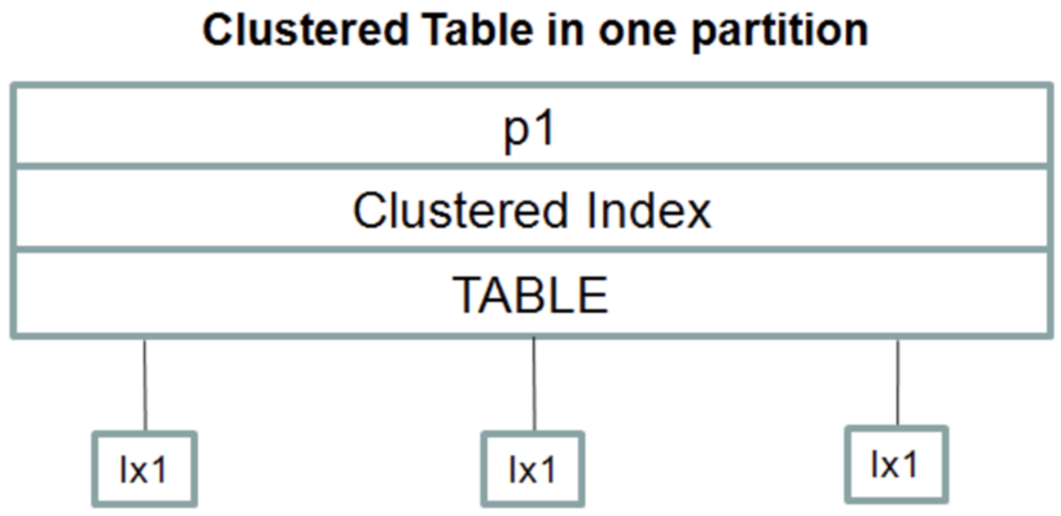
Table partitioning has two significant qualities:

- Partitions offer significant performance improvements in retrieving data from disks.
- Partitions add a level of intelligence to how data rows are distributed across the filegroups that comprise a table by automatically inserting rows into the appropriate partition, based on predefined data ranges for each partition.

Consider the following two designs. They show the same table—one without partitions and one with partitions. The table has a clustered index and three nonclustered indexes. The first example, Figure 15, shows the default—one partition.

Example 1

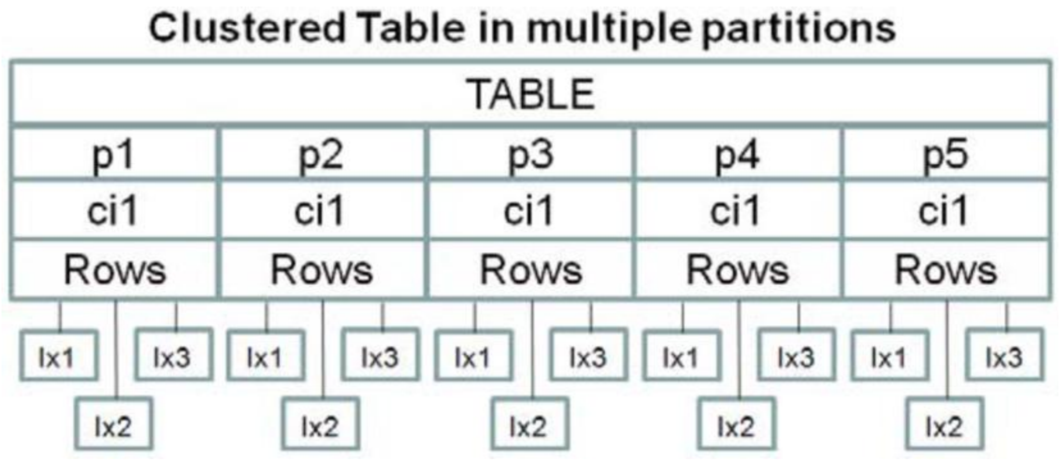
Figure 15) A table and its indexes residing in a single partition.



Example 2 shows the same table, but with five partitions. The indexes are also partitioned.

Example 2

Figure 16) A table and its indexes residing in five different partitions.



The second example, in Figure 16, shows the same table as in Figure 15, except it is using multiple partitions. In this example, each partition would hold approximately one-fifth the data as compared to Example 1. With a 1,000,000-row table, each partition would contain just 200,000 rows.

Information Lifecycle Management and Sliding Windows

One use for partitions is in information lifecycle and sliding windows solutions, both of which have very similar processing. Because partitions can be aligned with filegroups, and because filegroups can be aligned with LUNs, they enable business application–driven data migration at the LUN level through a process of splitting, switching, and merging partitions.

Partitions and Performance

As you review the following information, remember that each partition can reside on its own LUN. Many operations occur at the partition level. Query performance can be improved, because only the partitions containing the data ranges are searched.

Referring to Example 1 and Example 2 in this section, without the use of partitions, table scans, index reorgs, new index creation, and other table-level operations must operate against all one million rows. With partitioned tables, those same operations process just one fifth of the rows, so they operate much faster.

Loading data from an OLTP to an OLAP (On-Line Analytical Processing) system is much faster, taking seconds rather than minutes or hours in some cases, since just the partitions containing the required source data are read, rather than the entire table.

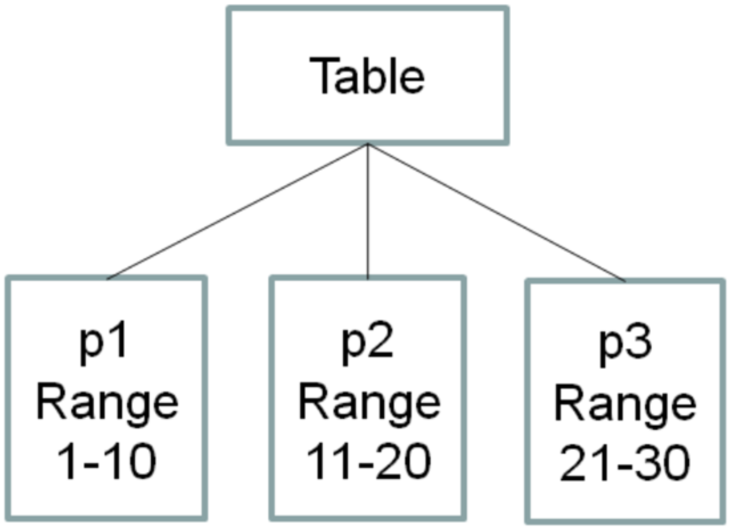
When adding new partitions to existing tables, the data can initially be loaded into an empty offline table, minimizing the effect on the production systems. After the load completes, the new table can be "switched" into the online table, which is a very fast metadata operation.

Table partitions can eliminate the need for partitioned views. Partitioned views are less efficient and require more administrative overhead.

Range Left or Range Right

This aspect of partitioning can relate directly to the number of LUNs required for the table, depending on the partitioning strategy that has been deployed. Partitions have boundaries, or ranges that determine which table rows belong in which partition. Boundaries are defined using the `CREATE PARTITION FUNCTION` command. Boundaries are enforced by way of a partitioning key, which is just a column in the table, and, optionally, table constraints. Refer to Figure 17. It shows a table with three partitions (p1, p2, and p3) with ranges of 1–10, 11–20, and 21–30.

Figure 17) A table with three partitions.



As each row is written to the table, the partitioning key is evaluated and the row is written to the appropriate partition. The question is: Where do rows that are outside the range of all the partitions get written? These rows must be accommodated. The answer is that one extra partition actually exists. The extra partition does not have a boundary value. Depending on the nature of the data, the extra partition is placed either to the left or to the right of all the other partitions. Placement is determined by using Range Left or Range Right. An example of using Range Right is included in the command examples that follow.

Creating Partitions

After completing the partition design, which includes identifying the partitioning key and determining how many partitions, filegroups, and LUNs are needed, you can create the partitions. Following is an example to demonstrate the commands used to create partitions. The LUNs are already mounted and ready to use.

These steps also are the basic flow when building partitions for a sliding-window scenario.

1. Create the filegroups. Commands for doing this were demonstrated earlier. This example uses four filegroups, with one LUN in each filegroup.
2. Create the partition function. This specifies the partitioning key and range values for each partition.

In the following example, four partitions are created.

```
Use MyDB
Go
CREATE PARTITION FUNCTION pf1_MyRange (int)
AS RANGE RIGHT FOR VALUES (1, 10, 20);
Go
```

The data are distributed across the partitions as follows:

- p1 – range values <= 1
- p2 – range values >1 and <= 10
- p3 – range values >10 and <= 20
- p4 – range values >20

3. Create the Partition Scheme. This maps each partition previously created to a filegroup.

```
Use MyDB
Go
CREATE PARTITION SCHEME ps1_MyRange
AS PARTITION pf1_MyRange
TO (MyDB_fg1, MyDB_fg2, MyDB_fg3, MyDB_fg4);
Go
```

This creates the following mapping:

- p1 to MyDB_fg1
- p2 to MyDB_fg2
- p3 to MyDB_fg3
- p4 to MyDB_fg4

4. Create the partitioned table and optional index(es).

```
Use MyDB
Go
CREATE TABLE pt_Table (col1 int, col2 int, col3 int))
ON ps1_MyRange (col1);
GO
CREATE NONCLUSTERED INDEX MyDB_IX1
ON MyDB.pt_Table (col1)
ON ps1_MyRange (col1);
GO
```

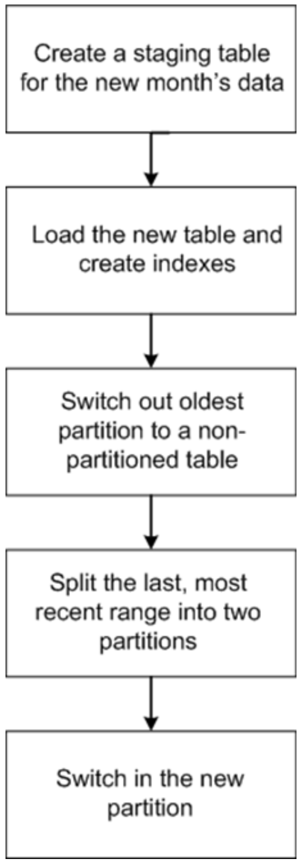
5. Load the table. This is a standard table load.
6. Other remaining steps might include creating clustered indexes and foreign key references.

Managing Partitions

Microsoft SQL Server provides a complete set of functions for managing partitions. The business requirements determine the exact partition management processes. Figure 18 shows a high-level example of the steps that occur for a specific scenario.

In this scenario, 12 monthly partitions are being maintained. It is time to add the next month of data. The oldest partition needs to be swapped out, and the new partition needs to be added in order to maintain just 12 partitions.

Figure 18) High-level step flow for adding a new partition and removing the oldest partition.


Create a staging table for the new month's data
Load the new table and create indexes
Switch out oldest partition to a non-partitioned table
Split the last, most recent range into two partitions
Switch in the new partition

Partitioning Summary

Partitioning introduces additional levels of complexity, including in the storage management area, and improperly designed partitions can actually impair query performance. But for certain business needs, the compelling features offered by partitioning outweigh these complexities. Also, as of the date of this writing, SQL Server 2008 will include partitioning wizards to further reduce complexity. For complete details on SQL Server partitions, refer to the SQL Server Books Online.

10.8 General Best Practice Recommendations

This section provides general recommendations for creating database storage designs on NetApp storage systems. Note that working with storage and database administrators is a subjective process, and sometimes best practices have exceptions.

The following information is based on systems that use the following technologies:

- Microsoft Windows Server

- NetApp SnapDrive for Windows
- NetApp SnapManager for SQL Server
- NetApp storage system

Keep It Simple

Keep the design as simple as possible while taking advantage of the appropriate technology features. Verify that the design supports and meets all the business requirements.

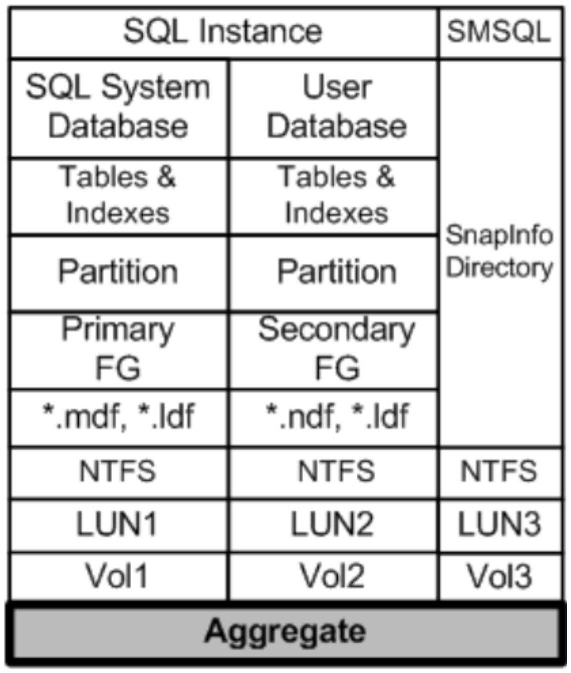
RAID Type

Use RAID-DP. When many spindles are available, put 16 spindles in each RAID-DP group.

Use One Aggregate

Aggregates are the lowest level in the storage stack. They are containers of physical disks from which volumes are carved out.

Figure 19) Aggregates highlighted in the SQL Server database storage stack.



Aggregates can be dedicated to specific databases or shared by all databases. Figure 19 shows one aggregate containing three volumes. The first volume is used by system databases, the second volume is used by user databases, and the third volume is used by SnapInfo.

NetApp recommends using one large aggregate for all SQL Server databases. There are two reasons for this:

- One aggregate makes the I/O abilities of all spindles available to all files.
- One aggregate enables the most efficient use of disk space.

NetApp has performed various tests using both approaches. Data and log separation as well as DSS and OLTP workload types were tested on both shared and dedicated aggregates. The conclusion is that one large aggregate yields significant performance benefits and is easier for administrators to manage. Also, as physical disks become larger and larger, efficient space management using multiple aggregates becomes even more challenging. For example, significant disk space is wasted in an aggregate containing high-capacity drives dedicated just to a high-utilization log file.

The prevailing reason for using more than one aggregate is high availability, for example, one for data and another for logs. With more than one aggregate, if one of the aggregates fails (which is highly unlikely), the odds are increased that there will be less data loss because the other aggregate is still available.

For environments requiring extremely high availability, NetApp SyncMirror® software can be used to create and maintain a local mirror of the complete aggregate.

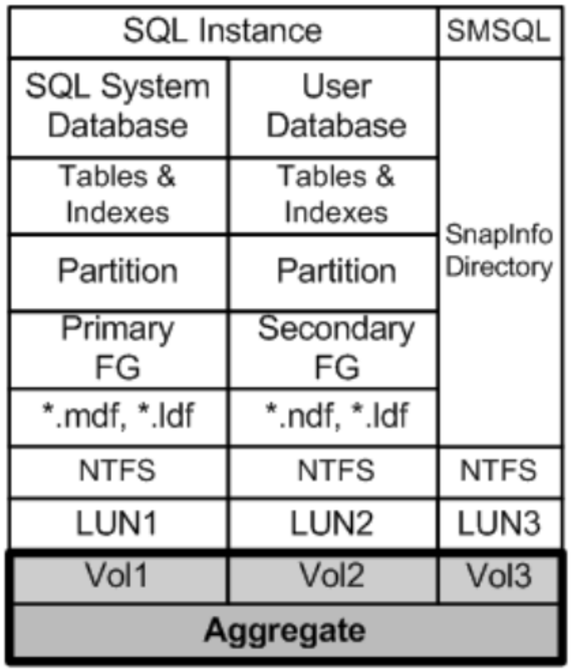
When creating and sizing aggregates, take into consideration:

- The total size of all the databases using the aggregate
- The number of database Snapshot copies that will be retained
- The number of log backups that will be retained
- The rate of data change and the duration of Snapshot copy retention
- The I/O load generated by all users accessing all the databases that will share the same aggregate
- The projected storage space growth
- The projected user count growth
- Plans for adding new databases to the aggregate
- Any other nondatabase files that may use the same aggregate
- When creating aggregates, let the NetApp storage system select which physical disks will be in each aggregate

Storage System Volumes

NetApp FlexVol volumes are created and reside inside aggregates. Many volumes can be created in a single aggregate, and each volume can be expanded or shrunk. Figure 20 shows an aggregate containing three volumes.

Figure 20) One aggregate with three volumes.



Snapshot Copies

SnapDrive takes Snapshot copies at the volume level. SnapManager for SQL Server also takes Snapshot copies at the volume level. With volume-level Snapshot copies, all the data in the volume is included in the Snapshot copy, even when only some of the data in the volume is pertinent to the specific database being backed up. For example, in Figure 20, if SQL DB1 is backed up, the LUNs for SQL DB2 will also be included in the Snapshot copy because both databases share Vol1 since they reside in the same volume. Up to 255 Snapshot copies can be created per volume.

SnapMirror

Like Snapshot copies, SnapMirror also operates at the volume level as well as at the qtree level. All the data in a source volume are mirrored to the target volume. Using one volume per database provides the most granular control over SnapMirror frequency and also provides the most efficient use of bandwidth between the SnapMirror source and the destination volumes. When addressing HA implementations, it is more common to have one volume for logs and a separate volume for all the data LUNs.

Volume Design Considerations

Before a database volume design can be created, the backup and recovery requirements must be defined. They provide the “specs” needed for the volume design process. Following are best practice considerations to apply during the volume design process:

- Place the SQL Server system databases on a dedicated volume to provide separation from the user databases.
- Place tempdb on a dedicated volume so that it is not part of a Snapshot copy.
- When using SnapManager for SQL Server, place the SnapInfo directory on a dedicated volume.
- It is common to take transaction log backups more frequently than database backups, so place the transaction log on a separate volume from the data files so independent backup schedules can be created for each.
- If each database has a unique backup requirement:
    a. Either create separate FlexVol volumes for each database and transaction log, OR
    b. Place databases with similar backup and recovery requirements on the same FlexVol volume. This can be a good option in cases of many small databases in a SQL Server instance.
- Storage administration overhead can increase as more volumes are used.
- Do not share a volume with more than one Windows server. The exception is when using Microsoft Cluster Services (Windows Server 2003) or Windows Failover Cluster (Windows Server 2008).

Note: Refer to the SQL Server Books Online and the NetApp SnapManager for SQL 2.1 Installation and Administration Guide for complete details.

Windows Volume Mount Points

NetApp storage solutions and Microsoft SQL Server 2005 support mount points. Mount points are directories on a volume that can be used to “mount” a different volume. Mounted volumes can be accessed by referencing the path of the mount point. Mount points eliminate the Windows 26-drive-letter limit and offer greater application transparency when moving data between LUNs, moving LUNs between hosts, and unmounting and mounting LUNs on the same host. This is because the underlying volumes can be moved around without changing the mount point path name.

Volume Mount Point Names

When using volume mount points, a general recommendation is to give the volume label the same name as the mount point name. The reason is illustrated in the following example. Consider a database with four volumes. Two volumes use drive letters, and two use mount points.

- Drive M: for the logs
- Drive S: for SnapInfo
- The data files use two mount points: mydb_data1 and mydb_data2. These are mounted in drive M: in the directory named mount.

Figure 21) Screenshot of database mydb data, log, and SnapInfo file paths with no labels.

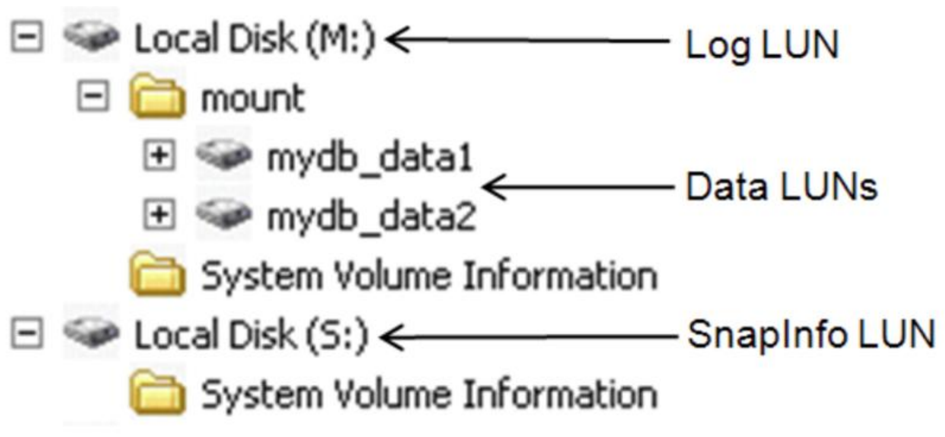
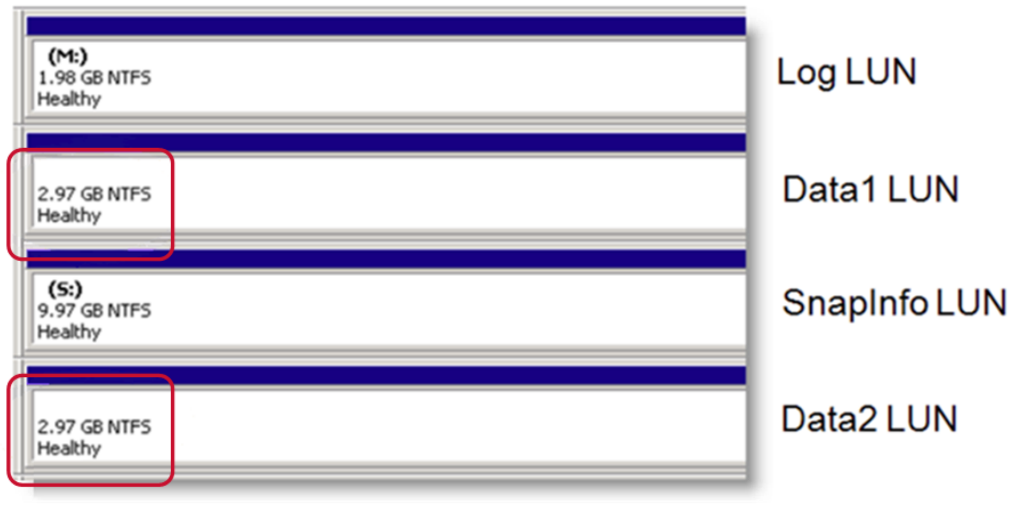


Figure 22) Screenshot of database mydb data, log, and SnapInfo LUNs with no labels.



Notice that it is not possible to identify how each volume is used. This can be resolved by using volume labels, as shown next.

Figure 23) Screenshot of database mydb data, log, and SnapInfo file paths with labels.

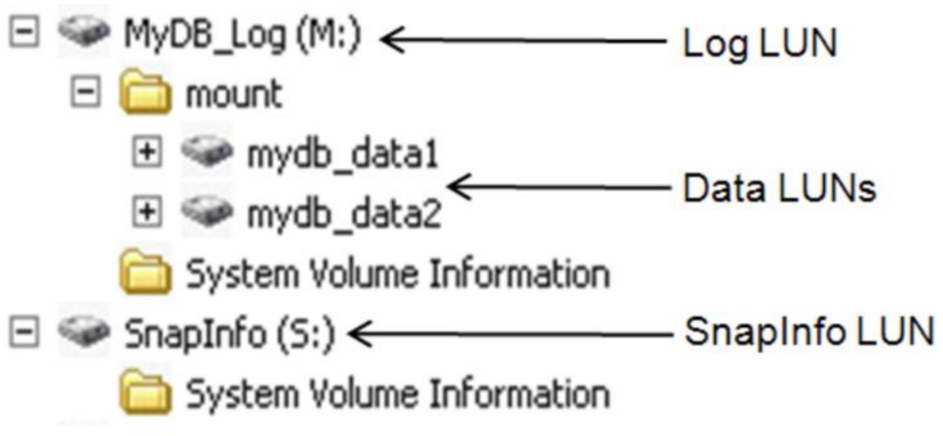
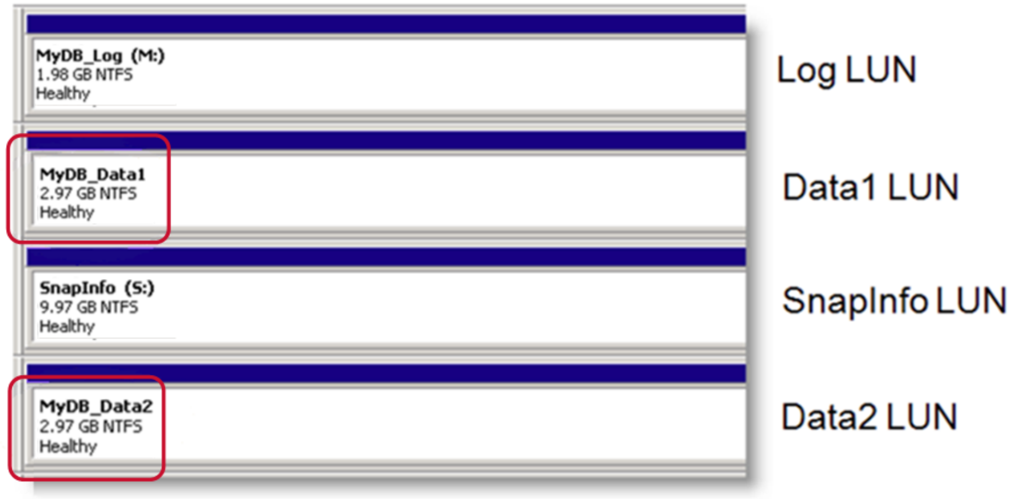


Figure 24) Screenshot of database mydb data, log, and SnapInfo LUNs with labels.



With the labels added as shown in Figure 23 and Figure 24, the purpose of each volume is clear. This is particularly useful when volumes are dismounted, then mounted on a different host or back to the same host, because the mount points must be reestablished.

Changing the volume name can be done by right-clicking the volume, in Windows Explorer for example; selecting properties; and typing in the new name. This is just a label, and changing the name does not affect the volume mount point path name used by applications.

Mount Points and SnapManager for SQL Server

NetApp SnapManager for SQL Server supports the use of mount points. Refer to the SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide for more information.

11 Database Storage Designs

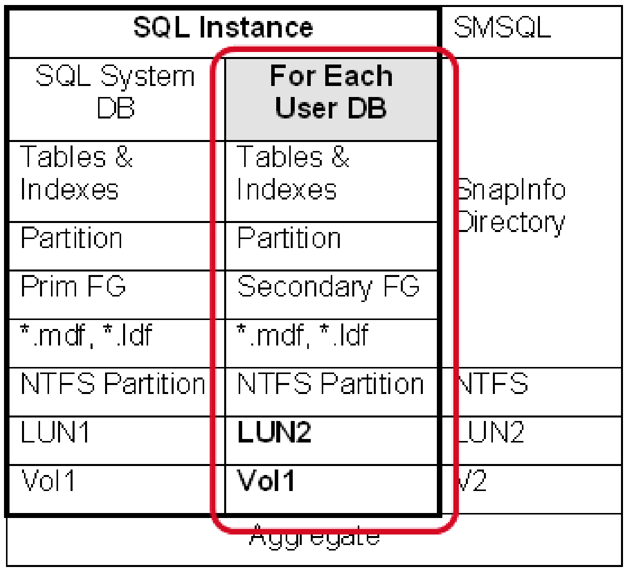
This section provides examples of SQL Server designs for NetApp storage, and takes into consideration environments that use SnapManager for SQL Server.

11.1 Design Example 1—Basic

Figure 25 shows a simple storage design.

- It does not use SQL Server partitions beyond the default configuration.
- There is one aggregate for the SQL instance.
- It uses a dedicated vol/LUN for the SQL Server system databases including tempdb.
- It uses a dedicated vol/LUN for each user database.
- It uses a dedicated vol/LUN for the SMSQL SnapInfo directory.

Figure 25) Basic SQL Server database design for NetApp storage system.



This configuration can be scaled for multiple user-defined databases per instance by replicating the circled area. It can also be scaled for multiple SQL Server instances on the same server by replicating the highlighted box.

The same aggregate can be used as the user databases and the SQL instances are scaled out. In fact the same aggregate can be used for all the SQL Server hosts connected to the storage system.

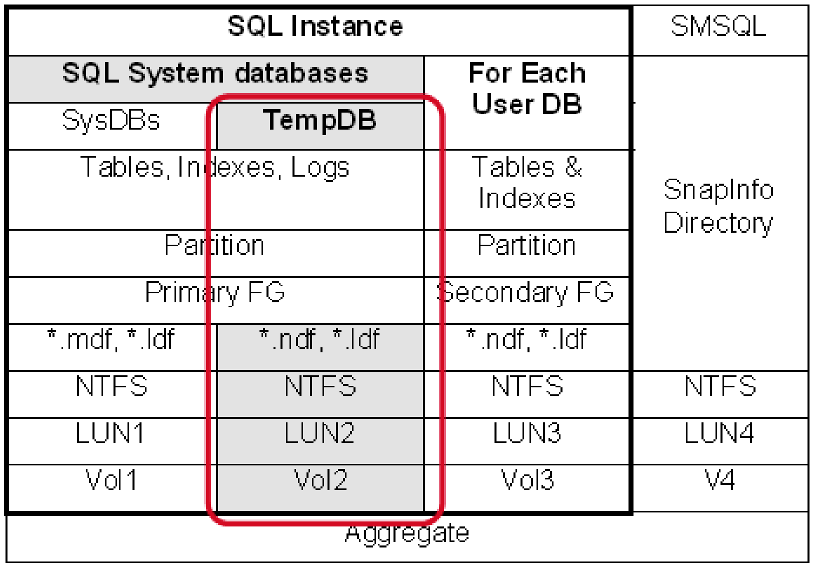
For SnapManager for SQL Server, there can be one SnapInfo directory for all the databases, which is implied in this design, or one per database. The SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide provides the information needed to determine what is appropriate for your specific design.

11.2 Design Example 2—Separate tempdb

The design example in Figure 26 is identical to design example 1 in Figure 25 except that tempdb has been placed on its own volume. Isolating tempdb onto its own volume makes it possible to keep it out of Snapshot copies. It also provides more granular control of which disks it resides on and how many LUNs it is comprised of. Spreading tempdb across multiple LUNs can help improve its ability to handle higher I/O requirements.

After the storage is configured, tempdb can be moved to its own LUNs using NetApp SnapManager for SQL Server or Microsoft SQL Server Management Studio.

Figure 26) SQL Server database design for NetApp storage systems—tempdb on dedicated volume.

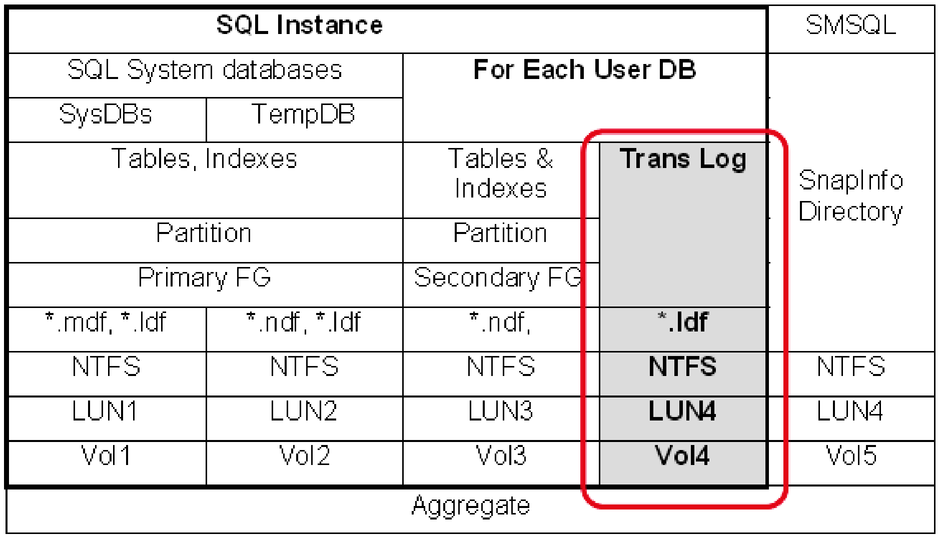


11.3 Design Example 3—Separate Transaction Log

The design shown in Figure 27 builds on the previous two. In this design, the user database log has been separated into its own volume. This provides more granular control of which disks it resides on and how many LUNs are included in it. Spreading the log across multiple LUNs can help improve its ability to handle higher I/O requirements.

This also allows the log file to be managed independently of the data file(s). Remember that Snapshot copies occur at the volume level. Using SnapManager for SQL Server, one schedule can create database Snapshot backups every hour and a different schedule can create log backups every 10 minutes.

Figure 27) SQL Server database design for NetApp storage systems—transaction log on dedicated volume.



When using SnapManager for SQL Server, a variation of this design can be used. Rather than having the transaction logs on a completely dedicated LUN, they can instead be placed in the root of the SnapInfo LUN. This design would then change from having five volumes to having four volumes. This is a good design because, taking the separate SnapManager for SQL Server schedules described in this section, SnapInfo would get backed up each time the logs are backed up.

11.4 Design Example 4—Multiple Filegroups

The user databases in the example detailed in Figure 28 use six different filegroups. The SQL instance has one aggregate, four volumes, and nine LUNs.

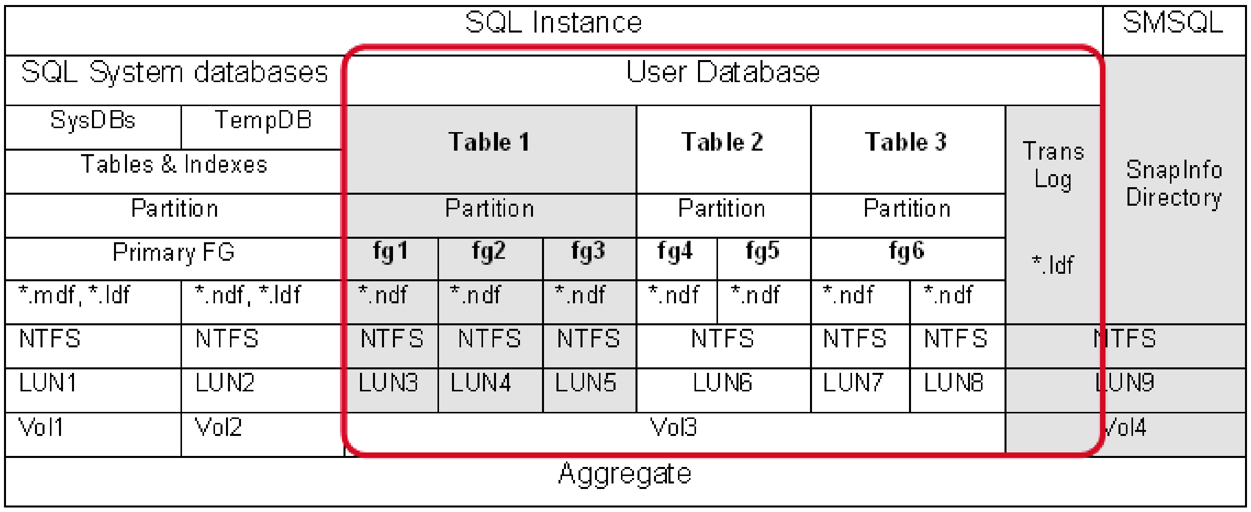
This shares the following design characteristics of Example #3:

- Tempdb is on its own LUN.
- The user database transaction log file is separate from the data files.

The differences in this design are as follows:

- The user database transaction log shares the same LUN as the SnapInfo directory.
- The user database has three tables.

Figure 28) SQL Server database design for NetApp storage systems with multiple filegroups.



Because all the tables share the same volume (Vol3), all the data is captured in a single Snapshot copy. As before, the log file is separated from the data, but now SnapManager for SQL Server's SnapInfo directory is also included in Vol4 Snapshot copies.

Let's look at each of the three tables.

Table 1

- This table uses three filegroups.
- Each filegroup contains one file and each of those files resides on its own LUN.

Table 2

- This example is included to demonstrate that this configuration can be done, but it is an unlikely configuration.
- This table uses two filegroups.
- Each filegroup contains its own file, but all the files share the same LUN.

Table 3

- This table has only one filegroup.

- The filegroup contains two files.
- Each file resides on its own LUN.

With all three tables, the only realistic level of management is at the table level. Moving individual filegroups, files, or LUNs does not make much sense.

11.5 Design Example 5—Table Partitions

This example (Figure 29) uses table partitions. Table 1 and Table 2 are identical to Table 1 and Table 2 in Example 4, except that Table 1 has been partitioned.

Figure 29) SQL Server database design for NetApp storage systems—table and index partitions.

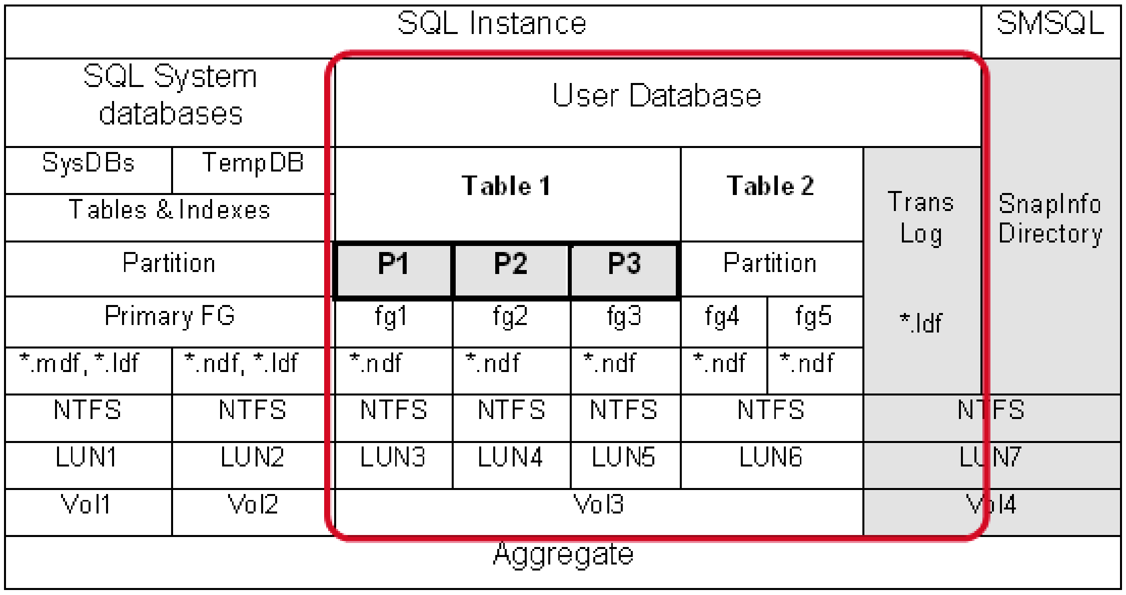


Table 1

- In this design, the SQL Server engine is distributing the rows between the three partitions based on a range value.
- Each partition can be split off and used as an independent table without corrupting Table 1.
- Because each partition is storage aligned (each partition has a dedicated LUN), it can be migrated to different types of storage based on the business rules implied by the partition ranges without concern about corrupting the table. For example, P3 could be moved off to a SATA volume, and this could be done with the database online.
- As discussed earlier in the partitioning section, each partition can have its own indexes, and nonclustered indexes can be located on their own LUNs.

12 Conclusion

Microsoft SQL Server 2005 is an appropriate product for many business-critical applications. There are many different ways to design the underlying storage. This technical report has introduced how SQL Server stores its data on disk and has explored various options that can be used to design a storage solution on NetApp storage systems. To be successful, spend the time to identify and understand required service-level agreements. Then, use the SQL Server and NetApp storage features that have been discussed in this report to create a storage and data management design that meets these requirements.

Once a solution has been designed and implemented, run tests to establish a baseline. Make certain the environment performs as expected, and save the results from the baseline tests for future reference. They can be very useful if the performance characteristics of the environment change. After deployment, monitor both SQL Server and the NetApp storage.

NetApp has proven data protection and disaster recovery tools for Microsoft SQL Server. SnapManager for SQL Server backup and restore capabilities and SnapMirror disaster recovery features are just two of the NetApp solutions that help provide a solid and robust solution for protecting and recovering SQL Server data.

Appendixes

SMSQL Best Practices Recap

Best Practice: Volume Mount Point

Make one of the transaction log LUNs the volume mount point root because it will be backed up on a regular basis. This makes certain that your volume mount points are preserved in a backup set and can be restored if necessary.

Also note that if databases reside on a LUN, do not add mount points to that LUN. If you must complete a restore of a database residing on a LUN with volume mount points, the restore operation removes any mount points that were created after the backup, disrupting access to the data on the mounted volumes referenced by these volume mount points. This is true only of mount points that exist on volumes that hold database files.

Best Practice: Always Place System Databases on a Dedicated LUN and FlexVol Volume

We can take a backup when the user database is present in a LUN in which system databases are also present. This is because we can restore the Snapshot copy for a user database, and SnapManager reverts to the conventional stream-based backup, dumping all data, “block for block,” to the snapinfo LUN.

Keeping system databases on a dedicated LUN and FlexVol volume separated from your user databases has the benefit of keeping deltas between Snapshot copies smaller since they do not hold changes to tempdb. Tempdb is recreated every time SQL Server is restarted and provides a scratch pad space in which SQL Server manages the creation of indexes and various sorting activities. The data in tempdb is thus transient and does not require backup support. Since the size and data associated with tempdb can become very large and volatile in some instances, it should be located on a dedicated FlexVol volume with other system databases on which Snapshot copies do not occur. This also helps preserve bandwidth and space when SnapMirror is used.

Best Practice: Backup Policy

Choose a backup retention level based on your SMSQL backup creation and verification schedule. If a Snapshot copy deletion occurs, make sure that a minimum of one verified backup remains on the volume. Otherwise, you run a higher risk of not having a usable backup set to restore from in case of a disaster. SMSQL executes defined deletion policies on a per-volume basis. If a backup set is spanned across multiple volumes, you must set similar or identical policies across those volumes. If this practice is not followed, you might end up with mismatching backup sets on different volumes. This can cause a backup set to be rendered useless and unable to be restored.

Best Practice: Fractional Space Policy

The threshold value for the policy “deletion of backup sets” should always be less than that of “dismount databases.”

Best Practice: Fractional Space Reservation

When a LUN is fully space reserved (fractional space reservation set to 100%), write operations to that LUN are protected against failure caused by an out-of space condition due to Snapshot copy disk space consumption. If you do require a fractional space reservation policy, work with your NetApp Support representative. The representative can help implement a fractional space reservation policy that fits your environment.

The fractional space reservation policy is set to 100% (disabled) by default. In order to leverage fractional space reservation policies, you must set the fractional reserve parameter to less than 100%.

Best Practice: Guidelines for Estimating the Size of the Volume

When you create a volume and you want to reserve a percentage of the space for overwrites, use the following calculation to determine the size of the volume:

(1 + fractional reserve) × LUN size + amount of data in Snapshot copies

Best Practice: Minimum Backup Frequency

All backup and recovery processes should be created according to each environment's unique requirements. When in doubt, the general recommendation is to back up the database every hour and the transaction log every 30 minutes during production time and possibly less often during off-peak hours.

Best Practice: Always Archive or Mirror Snapshot Copies

NetApp strongly recommends archiving or mirroring backup sets as soon as possible. Disasters do occur, and if the storage device containing the databases and backup Snapshot images is adversely affected, it will not be possible to restore the databases from the primary storage system. Archive media can be tape, another FAS system, a NearStore device, or even local SATA drives available in the FAS3000 series. The archive process is described in further detail in SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide.

Best Practice: Tips for Creating LUNs

1. When specifying a UNC path to a share of a volume to create a LUN, use IP addresses instead of host names. This is particularly important with iSCSI, because host-to-IP name resolution issues can interfere with the locating and mounting of iSCSI LUNs during the boot process.
2. Use SnapDrive to create LUNs for use with Windows to avoid complexity.
3. Calculate disk space requirements to accommodate data growth, Snapshot copies, and space reservations.
4. Leave automatic Snapshot copy scheduling off as configured by SnapDrive.

Best Practice: Run Database Consistency Checker (DBCC Checkdb) to Validate Database Before and After Migration

For usual operations, also try to execute the verification process during off-peak hours to prevent excessive load on the production system. If possible, off-load the entire verification process to another SQL Server instance running on another server.

Best Practice: Avoid Creating Snapshot Copies on Volumes with Active Snapshot-Writable LUNs

NetApp does not recommend creating a Snapshot copy of a volume that contains an active Snapshot copy because this creates a “busy Snapshot” issue.

Avoid scheduling SMSQL Snapshot copies when:

- Database verifications are running
- Archiving a Snapshot-backed LUN to tape or other media

Best Practice: When Using Supplemental Rolling Snapshot Copies

- Thoroughly read the “SMSQL Installation and Configuration Guide” section “Minimizing your exposure to loss of data.”
- Configure rolling Snapshot copies to occur only between SMSQL backup and SnapMirror processes. This prevents overlapping Snapshot schedules.

Best Practice: Use Alternative Methods for Remote Management

NetApp recommends avoiding Terminal Server for server management when possible. When using Terminal Server with Windows 2003 (not Windows 2000), you may use a remote desktop to connect if you use a command line parameter/console, shown here:

```
%SystemRoot%\System32\mstsc.exe /console
```

If you are using Windows 2000, NetApp advises using either NetMeeting remote control or RealVNC (available at www.realvnc.com/download.html).

Consolidation Methodologies

Consolidation of SQL Server databases, servers, and storage can provide great ROI benefits and optimize hardware, software, and administrative resources. This section identifies general approaches and methodologies in consolidating SQL Server environments. Much of the information presented in preceding sections is put into practice in a simulated customer deployment scenario.

Phase 1: Analysis of Existing SQL Server Instances

This section highlights some of the key planning phases that should be undertaken in the beginning stages of a SQL Server consolidation project by focusing on a fictitious company, “ACME Corp.” The company has a total of 11 databases running on 4 dedicated SQL Servers on direct-attached storage (DAS) and wants to consolidate all instances into a 2-node cluster configuration with NetApp storage.

The analysis phase begins with collecting information about existing databases in terms of bottom line criticality to the business. Table 8 illustrates the results of collecting information that is both technical and business related. The High Availability Index is a value used to depict the business-critical characterization of each database. This index is also used as a deciding factor in determining which backup and SnapMirror policies are implemented for each database. The High Availability Index works on a five-level designation, with 1 being the most critical and 5 signifying the lowest level of criticality.

Table 8) ACME's current SQL Server information.

SQL Server	Database	High Availability Index
SQL Server 2	DB3	2
	DB4	1
	DB5	2

SQL Server	Database	High Availability Index
SQL Server 3 DB 7 1	DB6	2
	DB7	1
	DB8	3
SQL Server 4 DB 10 3	DB9	3
	DB10	3
	DB11	1

Table 8 shows a completed table of information describing all databases intended for migration to the new environment, along with the ranking of importance to the business. Once the databases have been identified and ranked, the next phase of determining key technical metrics concerned with size, change rate, growth, and throughput can proceed.

It can sometimes be difficult to get data describing I/O throughput during peak loads (holidays, end of quarter, or other peak times), but this data can be very valuable when sizing storage devices correctly for current and future loads at peak times. Table 9 shows the type of data that can be very helpful when sizing the target storage device.

Table 9) ACME's I/O throughputs at peak load.

Database	Database Size	Read per Second (MB)	Write per Second (MB)	Change Rate per Month	Growth Rate per Month
DB 1	270GB	5	1	15%	5%
DB 2	8GB	8	1.5	10%	8%
DB 3	170MB	20	6	12%	6%
DB 4	2GB	0.5	0.1	5%	40%
DB 5	270MB	1.5	0.2	20%	10%
DB 6	43GB	1	0.5	10%	5%
DB 7	500GB	8	2	5%	1%
DB 8	320GB	5	1.2	50%	10%
DB 9	77MB	4	2	5%	5%
DB 10	100MB	2	3	10%	10%
DB 11	3112GB	20	10	10%	5%
Totals	4.872TB	75MB/Sec	27.5MB/sec		

Growth rate in terms of space can also be calculated based on projected system growth from a user growth capacity and business perspective. Things like database groom tasks, “data churn,” Snapshot copy and transaction log retention should also be taken into consideration.

If you are unsure how to calculate the total number of disks required, contact NetApp for an official SQL Server and NetApp storage sizing request. This helps enable the right number of spindles to be designated for your organization’s unique workload characteristics.

ACME’s Requirements

New business guidelines for ACME have dictated that all four SQL Servers be consolidated onto two newly purchased two-unit servers in an active-active cluster with a storage back end composed of a

FAS3050 system using eight full shelves of Fibre Channel disks. A single aggregate model has been selected consisting of a total of 112 144GB drives using RAID-DP with a group size of 16 disks.

The resulting configuration takes the following things into consideration:

- DB1, DB2, DB7, and DB11 have been designated as requiring the highest level of availability and require a full backup every 30 minutes and a transaction log backup every 15 minutes, with immediate SnapMirror replication to a disaster recovery site.
- DB3, DB5, and DB6 do not have as stringent requirements for recovery and can have their transaction logs backed up every 30 minutes, with a full backup every hour. Off-site SnapMirror replication is required for DR.
- DB8, DB9, and DB10 have lower backup and recovery requirements and can be backed up every hour and will not use SnapMirror.
- All databases must keep at least three Snapshot copies and additional transaction logs online at all times.
- Management has not budgeted for all databases to be mirrored and has allocated limited space to the DR site.

ACME's Requirements

New business guidelines for ACME have dictated that all four SQL Servers be consolidated onto two newly purchased two-unit servers in an active-active cluster with a storage back end composed of a FAS3050 system using eight full shelves of Fibre Channel disks. A single aggregate model has been selected consisting of a total of 112 144GB drives using RAID-DP with a group size of 16 disks.

The resulting configuration takes the following things into consideration:

- DB1, DB2, DB7, and DB11 have been designated as requiring the highest level of availability and require a full backup every 30 minutes and a transaction log backup every 15 minutes, with immediate SnapMirror replication to a disaster recovery site.
- DB3, DB5, and DB6 do not have as stringent requirements for recovery and can have their transaction logs backed up every 30 minutes, with a full backup every hour. Off-site SnapMirror replication is required for DR.
- DB8, DB9, and DB10 have lower backup and recovery requirements and can be backed up every hour and will not use SnapMirror.
- All databases must keep at least three Snapshot copies and additional transaction logs online at all times.
- Management has not budgeted for all databases to be mirrored and has allocated limited space to the DR site.

Monitoring SnapManager Events in SCOM

We can use Microsoft System Center Operations Manager (SCOM) to monitor SnapManager-specific tasks like backing up a SQL database using SnapManager for SQL (SMSQL) or restoring a specific virtual machine using SnapManager for Hyper-V (SMHV) or a LUN management task in SnapDrive. These applications log such events in the Windows event viewer. Porting them to SCOM saves us the trouble of logging on to the host every now and then to check the status of such backup tasks. We usually tend to schedule backups and leave them. SCOM can monitor and alert administrators about NetApp backup applications, be it a failed backup job in SMSQL or just a SnapDrive service restart.

1. Log on to SCOM. Open the Authoring Pane and select Monitors > Create a Monitor > Unit Monitor.
2. Go to Windows Events > Simple event detection and choose Manual Reset. Here choose the default management pack.
3. Type the name for the monitor. We can call it SnapDrive – service restart since we intend to generate an alert when SnapDrive service is restarted. Under Monitoring Target, select Windows Server 2008

R2 Full Operating System. This is because we have installed SnapDrive for Windows on a Windows 2008 R2 host.

4. Now click Next and select the Log Name where your application writes event logs. The SnapManager suite of products and SnapDrive for Windows write events in the Application Log. Therefore choose Application.
5. This is the window in which you enter all possible event IDs you want SCOM to monitor. The SnapDrive service restart event creates the event ID 100 in the Windows application log, so we enter this value here.
   Unfortunately, SCOM doesn't provide a way to upload all event IDs that you want to monitor at the same time. So, we must click Insert and add the event IDs one at a time.
6. Now we must configure health conditions, so that if the Event Raised condition occurs, then the status is Warning, otherwise it is Healthy. Click Next.
7. We need an alert when this event is generated. Click the Generate Alerts for this Monitor check box. Select Information under the Severity option and click Create.
8. Verify that this event is monitored under the target Windows 2008 R2 Full Operating System.
9. Generate the alert using Logevent.exe (the Windows event logging utility). This creates a test alert in the Windows Application Log. Click on the Alerts pane and verify that the SnapDrive restart event alert has been generated and is categorized as Severity type – information.
10. Select New Subscriptions by right-clicking Administration> Notifications > Subscriptions. Complete the wizard by entering SMTP details. This generates e-mail notifications for the events to which you have subscribed.
    In this way, you can monitor any event generated by SnapDrive for Windows, SnapManager for SQL, SnapManager for SharePoint, SnapManager for Exchange, and SnapManager for Hyper-V, among others in SCOM.

Perfmon

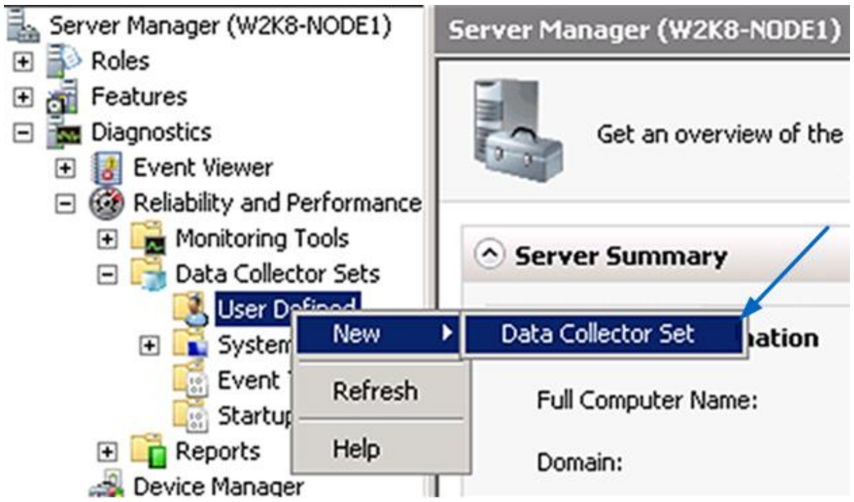
PerfMon is a tool that is part of the Windows Server system. This tool helps you easily monitor SQL Server to gather I/O read and write activity required for the database sizer tool.

Instructions for Running Perfmon

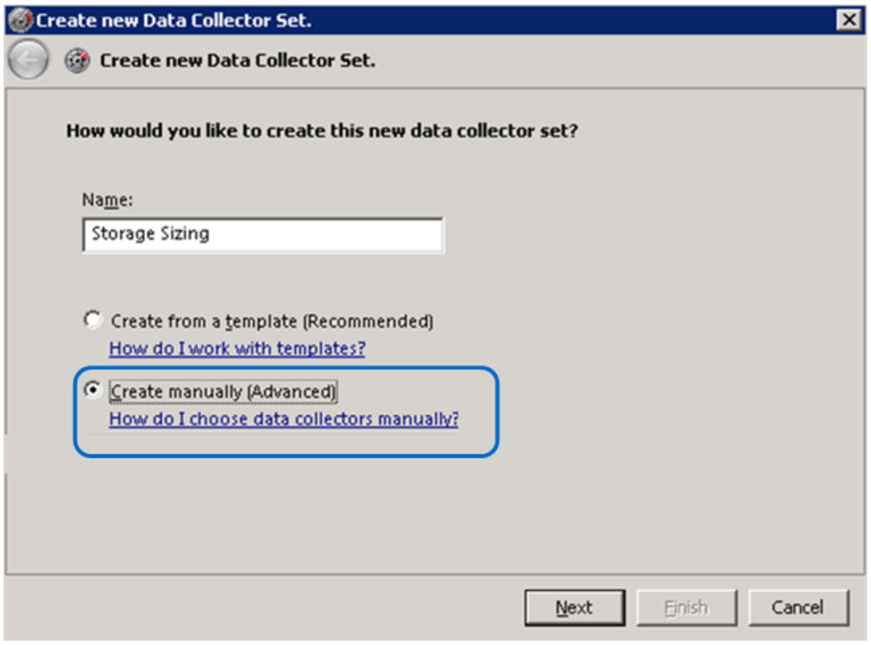
Start the PerfMon tool. The PerfMon tool can be used to monitor the SQL Server instance from the local instance or from a remote system. The remote system does not require SQL Server to be installed on it. However, both operating systems should have the same operating system service packs installed.

Create a data collection set from Server Manager. Open the Diagnostics Section, Reliability and Performance subsection.

Figure 30) Create data collection set.



1. Create a data collection manually.


Create new Data Collector Set.
Create new Data Collector Set.

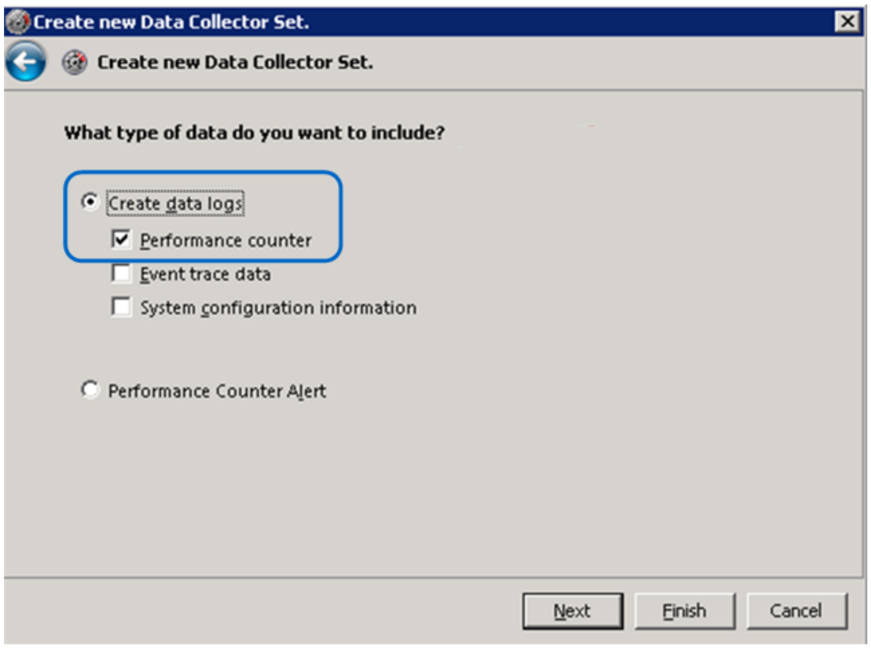
How would you like to create this new data collector set?

Name: Storage Sizing

( ) Create from a template (Recommended)
How do I work with templates?

(●) Create manually (Advanced)
How do I choose data collectors manually?

2. Create the data logs.


Create new Data Collector Set.
Create new Data Collector Set.
Next
Finish
Cancel

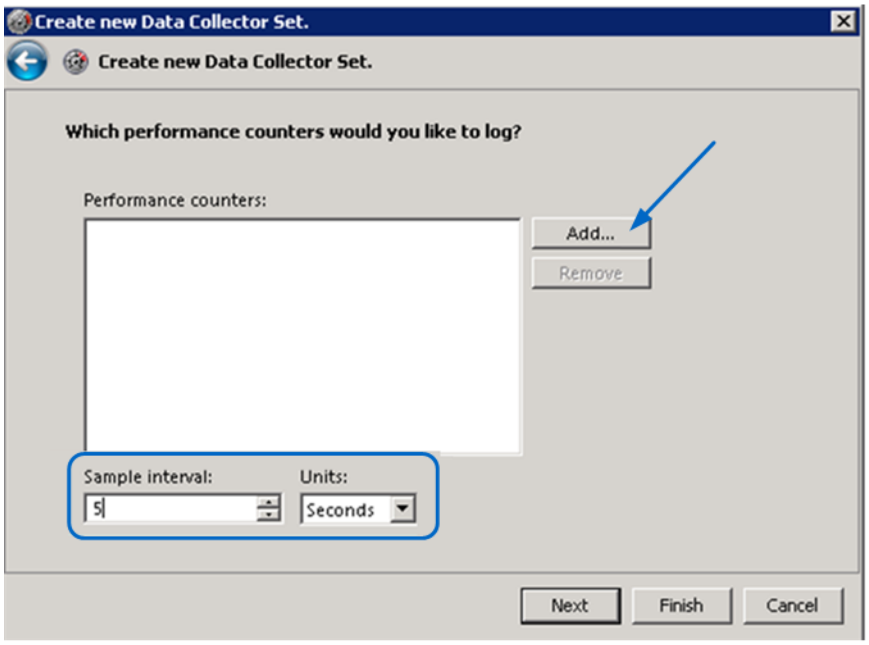
What type of data do you want to include?

[✓ Create data logs]

- [✓] Performance counter
- [ ] Event trace data
- [ ] System configuration information

Performance Counter Alert

3. Select the Sample interval and then add the performance counters


Create new Data Collector Set.
Create new Data Collector Set.
Add...
Remove
Next
Finish
Cancel

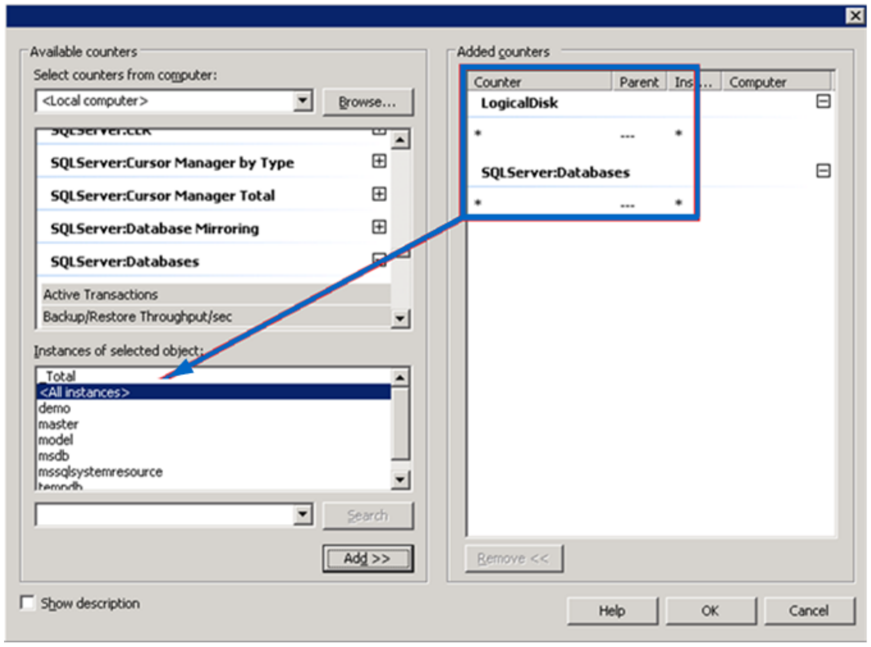
Which performance counters would you like to log?

Performance counters:

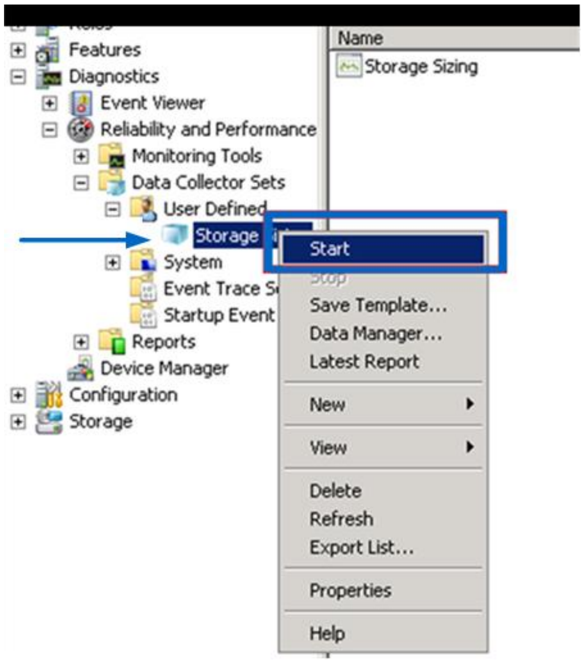
Sample interval: 5

Units: Seconds

4. Select the LogicalDisk and SQLServer:Databases counters for all SQL server instances you are trying to size.



5. Start the data collection. Data will collect into the `%systemdrive%\PerfLogs` directory in a .BLG binary format.



Note: When monitoring, attempt to capture data during peak usage so that you can capture the desired performance for the future system.

Use the `Relog.exe` that is built into Windows 7, 2008, or 2008R2 to convert the .BLG file into a CSV file type. The Relog tool can be downloaded from http://technet.microsoft.com/en-us/library/bb490958.aspx for Windows Server 2003.

Using Excel, open the CSV file and convert the Disk Read Bytes/sec. Divide by 1,024 to get your reads into MB/sec. Then convert the Disk Write Bytes /sec. Divide by 1,024 to get writes into MB/sec. Also divide AVG. Disk Bytes/Transfer by 1,024 to get I/O size in KB.

Phase II: Designing and Migrating to the New Environment

All databases were migrated to the FAS3050, with rules applied in the following areas:

- **Size and growth.** All user database LUNs have been sized by a factor of 2× + delta and have been calculated with additional room for growth for the next 12 months. DB11, sized at just over 3TB, has been split into multiple data files and spread evenly over four LUNs. For the critical database, you should size the volume using the formula
  (1 + fractional reserve) × LUN size + amount of data in Snapshot copies.
  Also total space reserved for overwrites is calculated using the formula
  Total Space Protected for Overwrites = Fractional Reserve Setting × Sum of Data in all LUNs.
- **High availability.** The base aggregate has been constructed using RAID-DP, and dual paths with MPIO have been used for connectivity. MSCS has been implemented for server-level failure. The threshold value for the policy “deletion of backup sets” is set less than that of “dismount databases,” to prevent the database from going offline.
- **Backup and recovery.** All databases that did not have conflicting requirements from a performance perspective have been logically grouped in FlexVol volumes by backup and recovery requirements. When possible, databases were also grouped in FlexVol volumes for granular control over SnapMirror replication.
- **Performance.** Some databases with more stringent performance requirements have had data and log files separated into different FlexVol volumes for optimal performance.

Table 10) Recommended database placement for ACME migration.

Instance	Database	Size	Database Type	H/A Index	LUN Mount (Device)	FlexVol Volume	SnapMirror
Quorum disk	Quorum data	100MB			Q	1	No
Node 1	System DBs	10GB		1	U	2	No
	DB 1	270GB	OLTP	1	E	3	Yes
	DB 2	8GB	OLTP	2	F	4	No
	DB 3	170MB	OLTP	1	G	3	Yes
	DB 4	2GB	OLTP	2	H	5	No
	DB 5	270MB	OLTP	2	I,J	5,6	No
	DB 6	43GB	OLTP	1	K,L	7,8	Yes
	DB 7	500GB	OLTP	3	Y	9	No
	DB 8	320GB	OLTP	3	N	9	No
	DB 9	77MB	OLTP	3	O OR C:\mntpto\db1	9	No
SMSQL	SnapInfo			1	P	10	Yes
Node 2	System DBs	150GB		1	W	11	No
	DB 10	100MB	DSS	2	S	11	Yes

Instance	Database	Size	Database Type	H/A Index	LUN Mount (Device)	FlexVol Volume	SnapMirror
	DB 11	3112GB	DSS	2	M, R, T, V OR C:\Mntpt1\db1 C:\Mntpt2\db2 C:\Mntpt3\db3 C:\Mntpt4\db4	12,13	Yes
SMSQL	SnapInfo			1	Y	14	Yes

Key Observations

- Cluster Server 1 groups databases on FlexVol volumes based on a backup policy of transaction log and database backups every 15 minutes and 30 minutes, respectively. Databases are also grouped by FlexVol volume for different SnapMirror policies and frequency.
- DB 5, DB 6, and DB 11 are all using two FlexVol volumes each to separate log and data files for optimal data locality because performance requirements for these databases are more stringent.
- The system databases for both instances have dedicated FlexVol volumes. The instance on node 2 is hosting the largest database, DB 11. The system database volume is sized at 150GB to allow ample room for growth of tempdb for the large query sorting and indexing that occurs in the DSS database DB 11.
- DB7, DB8, and DB9 are rated 3 on the high-availability index and are not configured for SnapMirror. They also all reside on the same volume to prevent them from being backed up with other schedules or unnecessarily mirrored.
- Snapinfo has also been designated for mirroring so that both metadata and transaction log backups are transmitted to the SnapMirror site.
- Because the cluster is active-active and must allow takeover, no drive letter is duplicated on either active node.
- We can use volume mount points and not drive letters to break down a huge database into separate LUNs, avoiding the issue of running out of drive letters.

The ACME example illustrates a successful implementation using optimal SQL Server file placement and volume and LUN layout while staying within the confines of drive letter limitations and SMSQL backup requirements. All files, LUNs and aggregates, and FlexVol volumes were implemented to provide just the right level of granular control over performance, backup, and disaster recovery policies.

Load Generation Scripts

T-SQL Code:

```
SET NOCOUNT ON
GO

CREATE TABLE tab1 (c1 int,c2 char(100),c3 char(100),c4 char(100),c5 char(100))
GO

declare @i as bigint
set @i = 1
while @i < 10000000
begin
  insert into tab1 values(@i,'abc','def','ghi','jkl')
  set @i = @i + 1
end
GO

SET NOCOUNT OFF
GO
```

OStress Command Line:

```
C:\rml\ostress.exe -SBTC-PPE-BLADE4\SQL_2K5_DR -E -dsqldrdb -n30 -i"C:\s1.sql"
```

Here the preceding T-SQL Script is saved in a file named `s1.sql` on the C: drive.

Recovering from Database Attach Failures on the DR Site

In case of issues with attaching the database at the DR site due to the checkpointing activity, follow these procedures. More about checkpoints can be found at http://msdn2.microsoft.com/en-us/library/ms189573.aspx.

Proposed Workaround

The following steps must be performed for working around checkpoint-related errors:

1. Verify that the LUN containing the data (.MDF) file is recovered and that the same drive letter is assigned to it as on the primary site. Change the extension of the .MDF file to .MDF.old.
2. Now create a new LUN and assign it the same drive letter as the LUN that contained the TLOG file (.LDF) on the primary site.
3. Create a new database with the same name as the one on the primary site. This new database does not have to be the same size as the one on the primary site; however, it must contain the same number of data and log files (with the same filenames) in the same drives as the original database.
4. Stop the SQL Server instance that you are working on.
5. Change the extension of the .MDF file of the newly created database to .MDF.new and the extension of the .MDF.old file to .MDF.
6. Restart the SQL Server instance.
7. Note that immediately the new database starts up in the SUSPECT mode and registers SQL Server Error 5173.
8. Run the following command from a new query window in the SQL Server Management Studio:

```
ALTER DATABASE <name of the database being recovered> SET EMERGENCY.
```

For example: `ALTER DATABASE sqldrdb SET EMERGENCY`.

This command brings the database into the EMERGENCY mode.

9. Now detach the newly created database.
10. Run the following command from a new query window in the SQL Server Management Studio:

```
CREATE DATABASE <name of the database being recovered>
ON (FILENAME = '<Complete path to the .MDF file being used>')
       FOR ATTACH_REBUILD_LOG
GO
```

For example:

```
      CREATE DATABASE SQLDRDB
             ON (FILENAME = 'F:\Microsoft SQL Server\MSSQL.1\MSSQL\Data\sqldrdb.mdf')
             FOR ATTACH_REBUILD_LOG
     GO
```

The preceding command creates a new TLOG file on the same drive as the one having the MDF file in the previous steps.

11. Run `DBCC CHECKDB` on the present database to determine whether there is any data corruption.

Tests and Results

This workaround was tested on the same testbed as that of business case 1, and we were able to bring the database online on the DR site.

Note: This workaround may result in data loss subject to the recovery.

Glossary

Aggregate. A manageable unit of RAID-protected storage consisting of one or two plexes that can contain one traditional volume or multiple FlexVol volumes.

Database Checkpoint. Forces all dirty pages for the current database to be written to disk. Dirty pages are data or log pages modified after entering the buffer cache but whose modifications have not yet been written to disk.

Data ONTAP. Data ONTAP 7G is a highly optimized, scalable operating system that supports mixed NAS and SAN environments. It includes a patented file system, multiprotocol data access, and advanced storage virtualization capabilities. Data ONTAP 7G is NetApp's premier operating system for general-purpose enterprise computing environments. It is the default software platform that is shipped with all FAS and V-Series storage systems.

FlexVol. A FlexVol volume is a logical file system of user data, metadata, and Snapshot copies that is loosely coupled to its containing aggregate. All FlexVol volumes share the underlying aggregate's disk array, RAID group, and plex configurations. Multiple FlexVol volumes can be contained within the same aggregate, sharing its disks, RAID groups, and plexes. FlexVol volumes can be modified and sized independently of their containing aggregate.

Logical Unit Number (LUN). From the storage system, a LUN is a logical representation of a physical unit of storage. It is a collection of, or a part of, physical or virtual disks configured as a single disk. When you create a LUN, it is automatically striped across many physical disks. Data ONTAP manages LUNs at the block level, so it cannot interpret the file system or the data in a LUN. From the host, LUNs appear as local disks on the host that you can format and manage to store data.

Partitioned Views. A partitioned view joins horizontally partitioned data from a set of member tables across one or more servers, making the data appear as if it is from one table. Microsoft SQL Server 2005 distinguishes between local and distributed partitioned views. In a local partitioned view, all participating tables and the view reside on the same instance of SQL Server. In a distributed partitioned view, at least one of the participating tables resides on a different (remote) server. In addition, SQL Server 2005 differentiates between partitioned views that are updatable and views that are read-only copies of the underlying tables. In SQL Server 2005, the preferred method for partitioning data locally is through partitioned tables.

Proportional Fill. SQL Server uses a proportional fill strategy across all the files within each filegroup and writes an amount of data proportional to the free space in the file. This enables the new file to be used immediately. In this way, all files generally become full at about the same time. However, transaction log files cannot be part of a filegroup; they are separate from one another. As the transaction log grows, the first log file fills, then the second, and so on, by using a fill-and-go strategy instead of a proportional fill strategy. Therefore, when a log file is added, it cannot be used by the transaction log until the other files have been filled.

Qtree. Qtrees are logically defined file systems used to partition data within a volume or assign quotas to limit the amount of storage space a user is allowed to use. Every file or directory is in a qtree, since the root of a volume is also a qtree. Other qtrees can be created as special directories in the root of a volume using the `qtree` console command.

SnapDrive. SnapDrive for Windows software integrates with the Windows Volume Manager so that storage systems can serve as storage devices for application data in Windows Server environments.

SnapDrive manages LUNs on a storage system, making this storage available as local disks on Windows hosts.

SnapMirror. This Data ONTAP feature enables you to periodically make Snapshot copies of data on one volume or qtree, replicate that data to a partner volume or qtree, usually on another storage system, and archive one or more iterations of that data as Snapshot copies. Replication on the partner volume or qtree promotes quick availability and restoration of data from the point of the last Snapshot copy, should the storage system containing the original volume or qtree be disabled.

Snapshot Copy. An online, read-only copy of an entire file system that protects against accidental deletions or modifications of files without duplicating file contents. Snapshot copies enable users to restore files and to back up data to tape while the NetApp storage system is in use.

WAFL. Write Anywhere File Layout. The WAFL® file system was designed for the NetApp storage system to optimize write performance. The storage system uses the WAFL blocks–based file system to manage file access and storage system performance. WAFL is a UNIX®-compatible file system optimized for network file access. In many ways WAFL is similar to other UNIX file systems such as the Berkeley Fast File System (FFS) and TransArc's Episode file system. WAFL is a block-based file system that uses inodes to describe files. It uses 4KB blocks with no fragments.

References

This section lists useful resources to assist in planning and managing your SQL Server storage environment.

- NetApp Storage Systems
  www.netapp.com/us/products/storage-systems/
- Data ONTAP Documentation
  http://now.netapp.com/NOW/knowledge/docs/ontap/ontap_index.shtml

Additional Documentation Available from the NetApp Support Site (Formerly NOW)

- NetApp SnapDrive for Windows Installation and Administration Guide
- SnapManager for Microsoft SQL Server (SMSQL) Installation and Administration Guide
- Microsoft SQL Server Customer Advisory Team - resources for complex enterprise SQL Server implementations
- http://sqlcat.com/
- Microsoft SQL Server Storage Engine Blog
  http://blogs.msdn.com/sqlserverstorageengine/default.aspx
- MSDN—Product documentation including SQL Server Books Online
  http://msdn.microsoft.com/en-us/library/bb545450.aspx
- SQL Server Best Practices
  http://technet.microsoft.com/en-us/sqlserver/bb671430.aspx
- SQL Server Hardware and Software Requirements
  http://technet.microsoft.com/en-us/library/ms143506.aspx

Microsoft SQL Server

- Description of disaster recovery options for Microsoft SQL Server
- INF: Disaster Recovery Articles for Microsoft SQL Server
- Disaster Recovery
- Introduction to Backup and Restore Strategies in SQL Server

- You cannot restore system database backups to a different build of SQL Server

SnapManager for SQL

- NetApp SnapManager for SQL 2.1 Installation and Administration Guide
- Best Practices Guide: Microsoft SQL Server 2000/2005 and NetApp Solutions

SnapDrive for Windows

- SnapDrive for Windows 5.0 Installation and Administration Guide
- SnapDrive for Windows Best Practices Guide

Data ONTAP

- Data ONTAP System Administration Guide
- Data ONTAP Storage Management Guide

NetApp SnapMirror

- SnapMirror How-To Guide
- SnapMirror Best Practices Guide
- SMSQL Best Practices Guide
- Database Layout with Data ONTAP 7G

Acknowledgements

The author would like to acknowledge the help of Mukesh Gupta, Prateek Caire, Venudhar Polluri, Sourav Chakraborty, Sukh Singh, Ross Mistry, and Wyman Ferguson in developing this document .

Version History

Version	Date	Document Version History
Version 1.0	January 2012	Initial release.
Version 1.0.1	October 2012	Updated for changes resulting from Microsoft SQL Server 2012 release.
Version 1.0.2	March 2013	Added best practices for Availability Groups.