SnapManager® v 1.0 for SQL Server 2000

by Gerson Finlev

Network Appliance, Inc.

October, 2003 | TR 3283

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Table of Contents

Abstract

SnapManager for Microsoft® SQL Server 2000 (SMSQL) is the newest member of the NetApp SnapManager product line. SMSQL relies on services by SQL Server 2000 and depends on SnapDrive™ servicing I/O-related requests to NetApp storage devices. SMSQL easily and rapidly backs up multiple databases in a short time, and is an excellent tool for migrating databases to virtual disks using NetApp storage. This paper will discuss the physical layout of SQL Server databases for backup and recovery when virtual disks on NetApp storage are used for storing SQL Server databases. Rules for consolidating SQL Server instances and databases will also be discussed in this technical report.

1. Introduction

This technical report describes the relationships between SQL Server's physical storage and SnapManager for SQL Server (SMSQL). We will analyze and illustrate various methodologies for the layout of the physical structure of database files on virtual disks and their implications for backup and restore, and the use of SnapMirror®.

It is strongly recommend that you read:

- Topics related to SnapDrive and administration of SQL Server can be found in *Guide to Integrating SQL Server with SnapDrive* (reference 1)

- *SnapManager 1.0 for Microsoft SQL Server: Installation and System Administration Guide* (reference 2)

Backup and recovery of a SQL Server database is tightly associated with a database's physical layout. Understanding the layout is critical when consolidating multiple SQL Server instances and databases and is an important part of any database design, especially when multiple databases share the same NetApp storage device.

This technical report covers several main ideas:

- Database storage objects are defined and different layouts are discussed, focusing on the use of SMSQL
- Database sizing and I/O load requirements
- Consolidating the storage of multiple SQL Server instances and databases
- SMSQL overview and its functions
- A checklist referencing related sections

Note: This technical report is not a replacement for *SnapManager 1.0 for Microsoft SQL Server: Installation and System Administration Guide* (reference 2).

Screen captures are used throughout this technical report to illustrate selections or a point. Some of the screen captures have been cropped to show only the important part of a screen.

Finally, a warning: Using Terminal Server (or Remote Desktop for XP) for administrating SMSQL from a remote PC is very common, but some SMSQL functions require temporarily mounting a virtual disk in a Snapshot™ copy. The drive letter with the newly mounted virtual disk will not be visible for the Terminal Server session and the SMSQL action will fail!

2. Database's Logical and Physical Layout

Designing and implementing a SQL Server databases is in most aspects intuitive, but can also be critical for future growth and performance, and important for optimal use of SMSQL. Understanding the relationship among database files, virtual disks, and NetApp filer volumes is important for optimal use of SMSQL. This section will discuss database file objects and explore various layouts of SQL Server databases, and ways to utilize virtual disks that can be supported by SMSQL.

Both system and user databases deploy the same basic structure; therefore, the given description will cover system databases[1], demonstration databases[2], and user-defined databases. The focus is not the logical structure of database files but a database's physical files and how they are accessed.

A database consists of two or three different physical file types with different content. The file types use the following default file extensions:

- **.ldf** is a transaction log. A database has at least one transaction log that contains information necessary to recover all transactions in a database.
- **.mdf** is the database's primary data file. Every database has one primary data file that keeps track of all other database files and internal database structure, in addition to storing data.
- **.ndf** is the secondary data file. A database can have zero or more secondary data files that contain only data.

Data files can be logically grouped into file groups. A file group consists of one or more physical data files.

Figures 1 to 5 illustrate different layouts of SQL Server databases, followed by a description including advantages as well as disadvantages.

2.1. Common Database Layouts

The most common layout of a database is with two files: one primary data file and one transaction log file as illustrated in figure 1. Both the data file and the log file use the same virtual disk: disk "M."

Advantages of the layout in figure 1 are:

- Only one drive letter is used for the database[3]
- Easy to administrate and to understand the relationship between files and the database
- The virtual disk can be shared by several databases when all files belonging to the databases are located in the same virtual disk[4]
- Making SnapMirror copies is simple since only one filer volume is used by the database

Disadvantages are:

- If the volume fails, the database has to be recovered from an archive of a full backup and backup transaction logs. A volume almost never fails but it can happen. Archiving backed up databases is important for disaster recovery.
- Synchronous access to the transaction log and random access to the data file are mixed in the same volume, but that is only a potential issue with environments that require high levels of I/O access. Database files can always be migrated with SMSQL.
- It is not practical with very large databases (700GB and up).

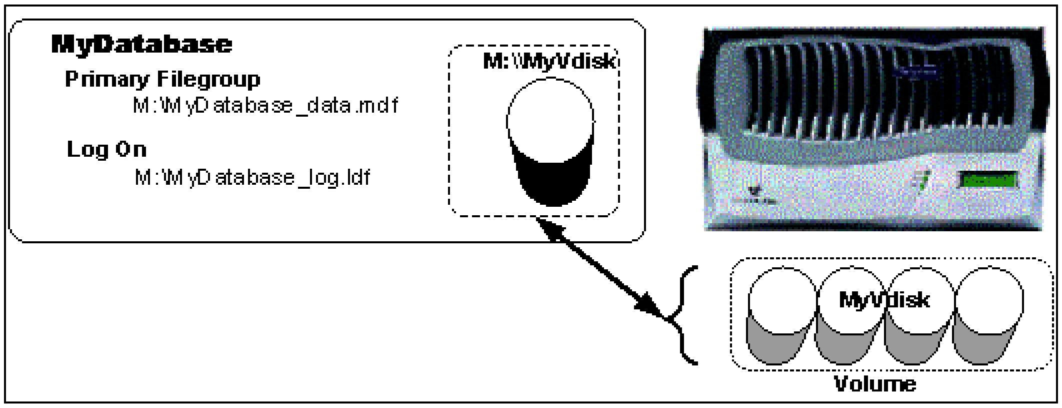


Figure 1. Basic Structure of a SQL Server Database

Another common layout of the database's files is illustrated in figure 2. It is very similar to the illustration in figure 1, except that the transaction log file and the data file are each located in separate virtual disks utilizing a single volume. The transaction log is located in the X device and the data file is located in the M device.

Advantages of the layout in figure 2 are:

- SnapMirror transactions are simple when only one filer volume is used by the database.
- I/O is distributed over two separate virtual devices so the network traffic can be divided between two distinct transport layers and volumes.
- The potential I/O throughput is greater[5].

Disadvantages are:

- Uses two drive letters.

- If the single volume fails, the database has to be recovered from archive and backed up on transaction logs.
- Not practical with very large databases (700GB and up).
- The virtual disks cannot be shared by any other database.

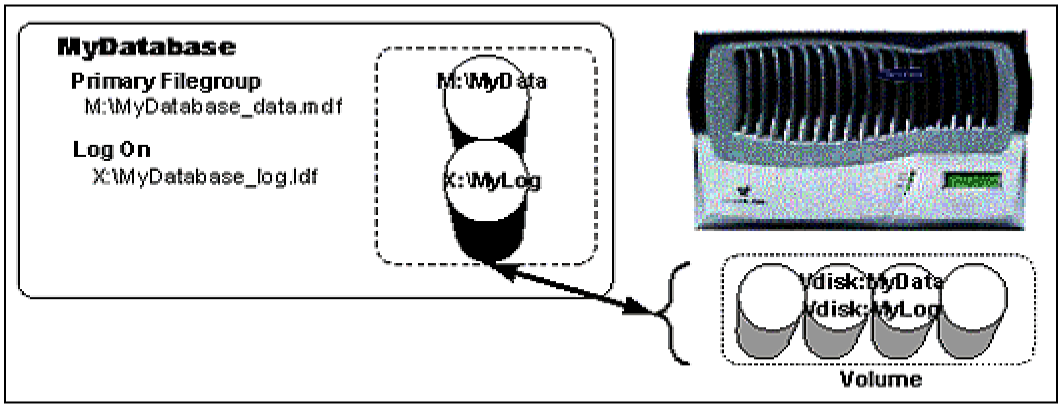


Figure 2. Database Layout with Two Virtual Disks

2.2. Large Databases with Sequential I/O Access

Large databases with sequential access can potentially increase the I/O performance when database objects (like tables or indexes) are created in file groups consisting of several data files. When the SQL Server realizes that a table or index has been created in a file group with several files, then SQL Server will extend the table space round-robin[6] over available files. In practice, this means that SQL Server will allocate 64kb round-robin over all files in the file group[7].

The database in figure 3 has three data files in `MyFilegroup` and one transaction log file. The files in file group `MyFilegroup` are all located in the M device and stored in one volume. The transaction log file[8] is stored in device X, which is located in a second volume.

Advantages of the layout in figure 3 are:

- You may achieve higher concurrent I/O access when a table is sequentially scanned[9].
- I/O is distributed over two separate volumes, so the sequential access to the transaction log is to a different volume than the access to the data files. Access to the transaction log is always sequential but access to data files is potentially random. Access to OLTP databases is generally random and access to data warehouses is mostly sequential.
- If the database becomes too large for one volume then it is fairly easy to move one of the data files to another volume, or to expand the file group with a new file located in another volume.

Disadvantages are:

- Uses two drive letters.
- The virtual disks cannot be shared by any other database.
- SnapMirror processes are more complicated.

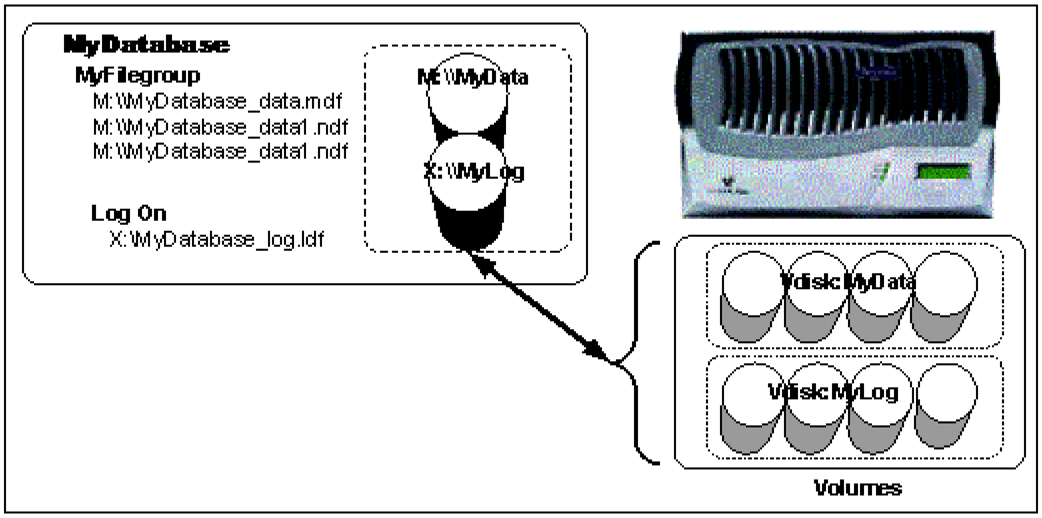


Figure 3. Database Layout Using Two Volumes

2.3. Very Large Database with High I/O Performance

Large databases with high I/O performance requirements can create large tables in different file groups with multiple data files. Multiple data files can potentially increase the physical I/O rate[10] by making it possible for SQL Server to issue concurrent I/O requests to each file in a file group[11]. Data files in a file group can be stored in one single device, Filegroup 1, or in several devices, Filegroup 2 (figure 4). Reasons for using several filer groups are:

- The database has several large tables[12].
- A table and its indexes can be created in different file groups.
- The database contains very large tables that are too large for a single volume. The data files in the file group can easily be spread over multiple volumes, which can potentially also increase the I/O performance, but that depends on the application's access pattern.

Advantages of the layout in figure 4 are:

- You will be able to achieve more concurrent I/O access when tables are sequentially scanned.
- A table and its indexes can be created in different file groups.
- I/O is distributed over two separate volumes; therefore, access to the transaction log is to a different volume from the volume that contains the data files. Access to the transaction log is always sequential but access to data files is potentially random. Access to OLTP databases is generally random and access to data warehouses is mostly sequential.
- If the database becomes too large for one volume it is fairly easy to move some of the data files to another volume or to expand the file group with new files located in another volume.

Disadvantages are:

- Creating several filer groups uses many drive letters.
- The virtual disks cannot be shared by any other database.
- Backing up the files with SnapMirror is complicated.
- The layout is complicated and rarely needed; a NetApp filer is great at load balancing I/O over available disks in a volume.

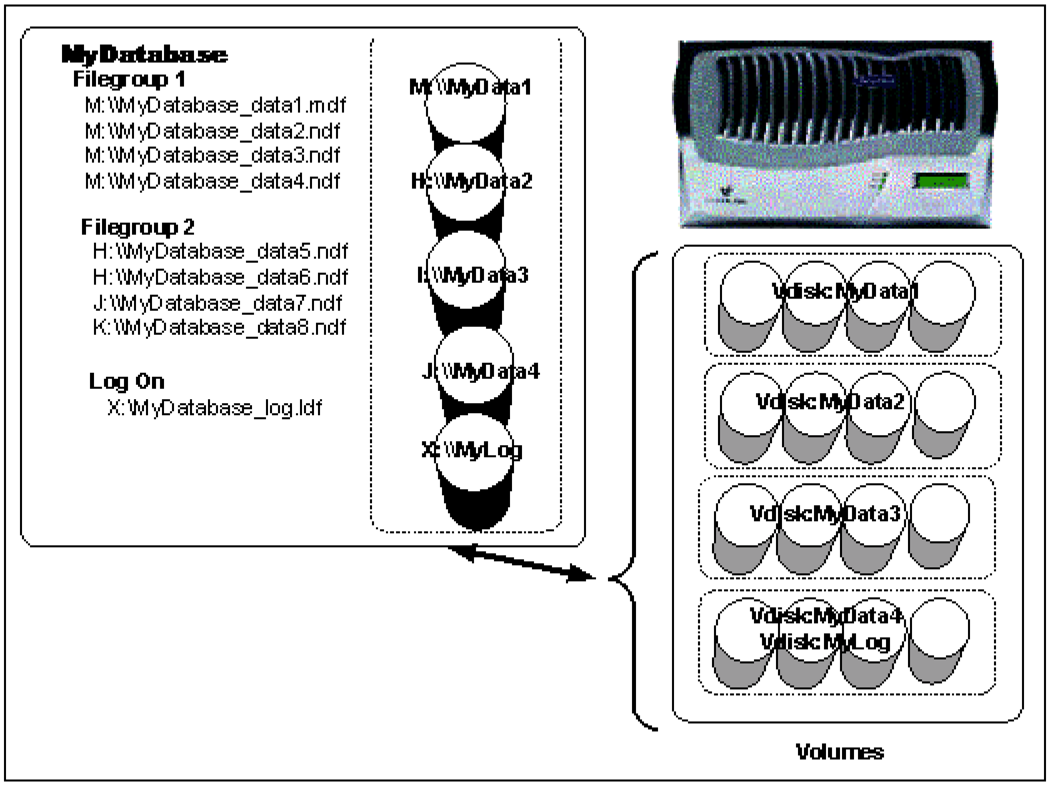


Figure 4. Database layout with multiple filer-groups

2.4. Several Databases on One Windows Server

We have until now illustrated possible layouts of a single database. In this section we will discuss how to deploy several databases controlled by one or more SQL Server instances running on a single Windows platform. Each Windows[13] platform has a fixed number of available drive letters—around 20 that all databases filers have to be allocated across. If there are more databases than available drive letters, then the databases have to be grouped together according to some criteria and stored in one virtual disk. Deciding on the physical layout for multiple databases spread over several virtual disks can make the design complicated.

Figure 5 illustrates two basic concepts when several databases are controlled from a single server.

The first concept is illustrated by databases MyDatabase and MyDB; all files belonging to those two databases are located in one single device, M. A device can be shared by many databases as long as all files belonging to the databases are located in the same virtual disk. The second concept is illustrated by database MyLargeDB. When database files are spread over several virtual disks then those virtual disks cannot be used by any other database. This is because of the use of Snapshot™ and SFSR[14]; if MyDatabase was located in the J drive then SMSQL could not restore MyDatabase without MyLargeDB becoming corrupted (figure 5).

When several databases share a single virtual disk then SMSQL can quickly back up all databases in the volume[15] and can quickly restore all databases in the virtual disk; section 8 describes how to restore a single database out of many databases in a virtual disk. It is often quicker to restore all databases in a virtual disk than restoring the corrupted database, but that depends greatly on the size of the databases. Restoring one of the databases on a virtual disk takes about the same time as using SQL Server level restore, but a full backup with SQL Server takes much longer than when SMSQL is used for full backup.

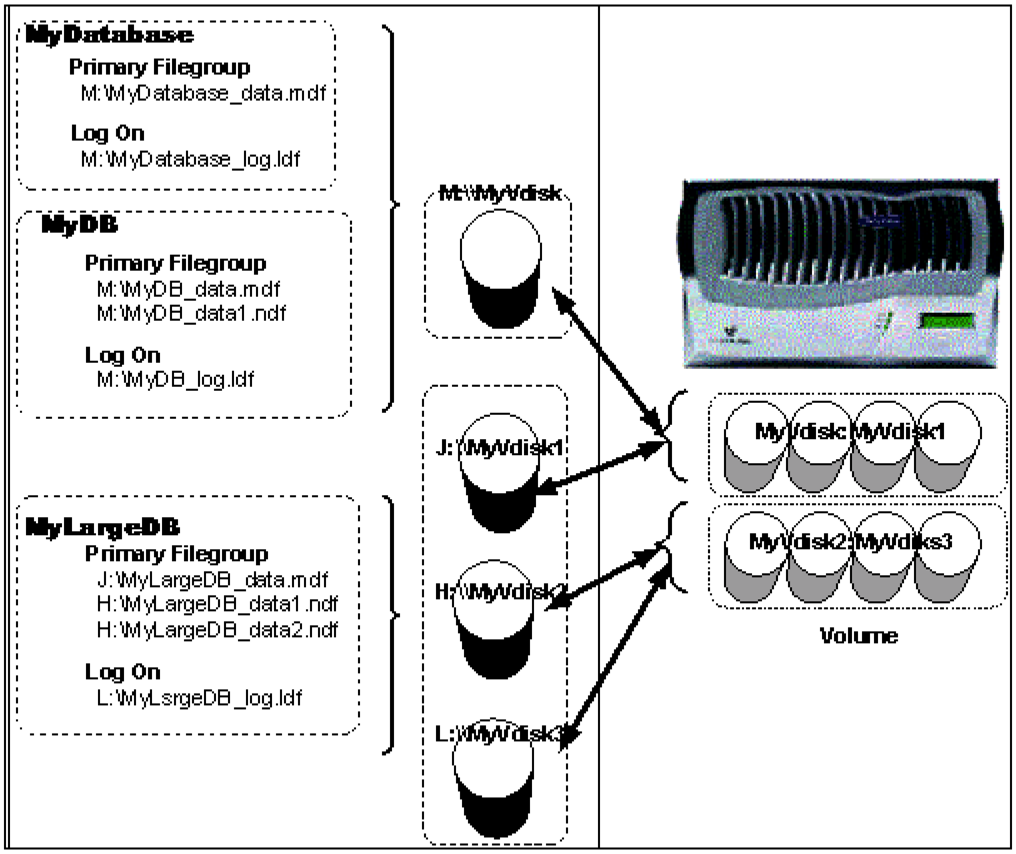


Figure 5. Multiple Databases

When a server has to handle more databases than there are available drive letters[16], you should sort all databases according to size, acceptable downtime, and how critical the database is.

Size: The larger a database, the greater the advantage of using SMSQL for backup and restore. Grouping the largest database with the smallest databases makes it possible to restore all databases if the large database has to be restored, or to restore a single smaller database if that has to be restored, as described in section 8.

Acceptable downtime: Knowing the acceptable downtime makes it possible for the DBA to decide the order in which databases have to be restored. By selecting only the database(s) that have to be restored first (see figure 33), the selected database(s) will be restored and be online first, while the other databases will be in a "loading" state (figure 34). The DBA can apply transaction logs and take databases online in the required order.

Critical: Databases that are critical can be located in their own virtual disk. This might require that more of the remaining databases will be located together in the other virtual disks, but this is a tradeoff that has to be made.

3. Sizing

In addition to the space required for the virtual disk to store SQL Server databases, free space is required to store data that changed between Snapshot copies of the active file system of the virtual disk(s). Providing additional space ensures that multiple Snapshot backups can be created of database(s). General considerations when creating a NetApp volume are:

- To be able to create Snapshot backups, the volume has to be at least twice the size the virtual disks the volume stores.

The volume has to have enough free space to support the number of Snapshot copies to be kept online.

The following data assists in estimating volume size and virtual disk size:

- *Initial database size* is the starting point for any sizing project.
- *Database growth rate* is important for sizing the virtual disk(s) correctly so the virtual disk and the volume(s) don't need to be expanded too soon and too often.
- *Change rate of current database* is important for estimating the Snapshot space requirements.
- *Free volume space requirements for Snapshot copies* are at least equal to the size of the virtual disk plus changed file blocks.
- It is recommended to use the default *RAID group size*.
- It is also recommended that you operate with extra free volume space. Maintaining extra free volume space will decrease possible volume fragmentation rate and will also prevent sudden "out of space" events. Also, when a volume's free space is very limited I/O performance could suffer.
- Depending on the access pattern, a single disk drive can support about 150 I/O operations per second (IOPS) with current disk technology. Therefore, it is important to understand the I/O rate during peak loads to estimate the number of disk drives needed to support the I/O load. A RAID group size of eight disk drives (seven data and one parity) can support about 1,050 IOPS (7*150 IOPS).

The following is an example of sizing a database with the following requirements:

- Initial database size is 100GB.
- Database growth rate is estimated to be 10% per month.
- Change rate of current database is estimated to be 15% per month.
- Backup requirement is 4 backups (Snapshot duplicates) per day, with a total of 12 Snapshot copies (3 days).
- Default RAID group size is eight drives, 72GB drive size.
- The administrator wants to expand the virtual disk (and volume) only every six months.
- The growth and change rates are estimates, so extra space has been requested. Also, the customer realizes that a volume has to operate with some free space to decrease the possible fragmentation rate, or I/O performance could suffer. Therefore, an extra 20% free space per disk drive will be allocated as a free-space buffer
- The average I/O rate is about 1.5MB per second and the peak rate is about 3MB per second.

The result of using above estimates is:

1. The database size after six months will be about 180GB.
2. About 27GB of the database will change every month after six months, which is equal to 0.15GB every four hours.
3. The minimum space requirement after six months will be (180GB * 2) + (0.15GB *12) = 362GB
4. A 72GB disk drive has 68GB of usable file space, and since 20% is allocated as extra free space, only 55GB will be allocated per disk drive. Hence, six disk drives are needed for data and one disk drive for parity, for a total of seven disk drives. However, it is important for performance reasons to always configure complete RAID groups; therefore, the volume will be created with eight disk drives. If the rate of growth is consistent over time, the volume will have to be expanded with another RAID group after six or seven months.

5. The estimated peak load was 3MB per second, which is equal to about 770 IOPS. Since seven data disk drives can support 1050 IOPS, a RAID group size of eight will be sufficient to support both space requirements and I/O load requirements.

The above estimate doesn't account for the space required for the database's transaction log. Sizing space for a database's transaction log requires an understanding of the following factors:

- The rate of transactions that modify a database record
- The size of the transaction
- The frequency of the transaction log backup
- The active part of the transaction log will be backed up, and not the total size of the transaction log

The key to sizing requirements correctly for database's transaction log is to monitor usage over time.

3.1. System Databases

Except for tempdb all system databases are fairly small. Tempdb is a work area for all SQL Server's temporary storage requirements. The size of tempdb depends exclusively on the operations executed by database applications, the number of databases, and the number of users. Tempdb can become very large. It is very hard to predict how big tempdb can become, but the I/O load can be very high and, since it is recreated every time SQL Server starts up, it is never reused. The tempdb should be located on a volume with no Snapshot copies.

3.2. SnapInfo

SnapInfo contains metadata, streamed full backup of system databases, streamed full backup of user-defined databases that share a virtual disk with a system database, and backed up transaction logs. Space requirements for SnapInfo depend on all above factors, including the number of databases, the size of streamed databases, number of backups to be kept online, and the active size of a database's transaction log since the last time the transaction log was backed up.

Metadata per backup set is less than 10kb.

The size of each backed up system database is less than 15MB.

The size of a backed up transaction log depends on the number of new transactions received since the last time the transaction log was backed up and on the database's recovery model. The full recovery model is the most common recovery model and will generate one record per transaction, and the backed up transaction logs will be approximately the same size as the active part of a transaction log. Bulk-logged recovery model is not used that often, but the size of a backed-up transaction log when the recovery mode is bulk-logged can become much larger compared to full recovery mode. Bulk operations will create minimal logging, and bulk changed database pages will be marked for inclusion in the next backed-up transaction log. Therefore, when a bulk-logged database's transaction log is backed up, all bulk marked pages will be written to the backup. Perfmon can monitor databases' used and free space including the active part of the transaction log.

Full backup (Snapshot) of user-defined databases will only add a set of metadata to SnapInfo.

Two excellent resources for capacity planning and performance tuning are references 4 and 5.

4. General Consolidation Principles

It quite common for companies to add new Windows 2000 servers whenever new SQL Server applications are required for a business purpose. It is fairly easy to add new hardware to network infrastructure, to install the application, and to make the new environment available to end users. This works well for some companies but can create administrative and maintenance issues for others.

Many companies begin to think about consolidation when they discover that it is hard to maintain environments with multiple servers and locally attached storage, which can become costly and complicated. This section will describe important considerations when consolidating SQL Server environments.

Figure 6 shows a common infrastructure with a number of servers, with each server supporting one SQL Server instance and application. Each server and its resources are independent[17] from each other and have to be administrated separately. The administration of each server includes:

- Performance monitoring and tuning
- Capacity planning
- Maintenance
- System-level backup
- Database backup

Not only is it complicated and time-consuming to administer many servers but available resources on one server can not be utilized by another server, such as:

- Idle processors
- Memory
- Free files and disk space

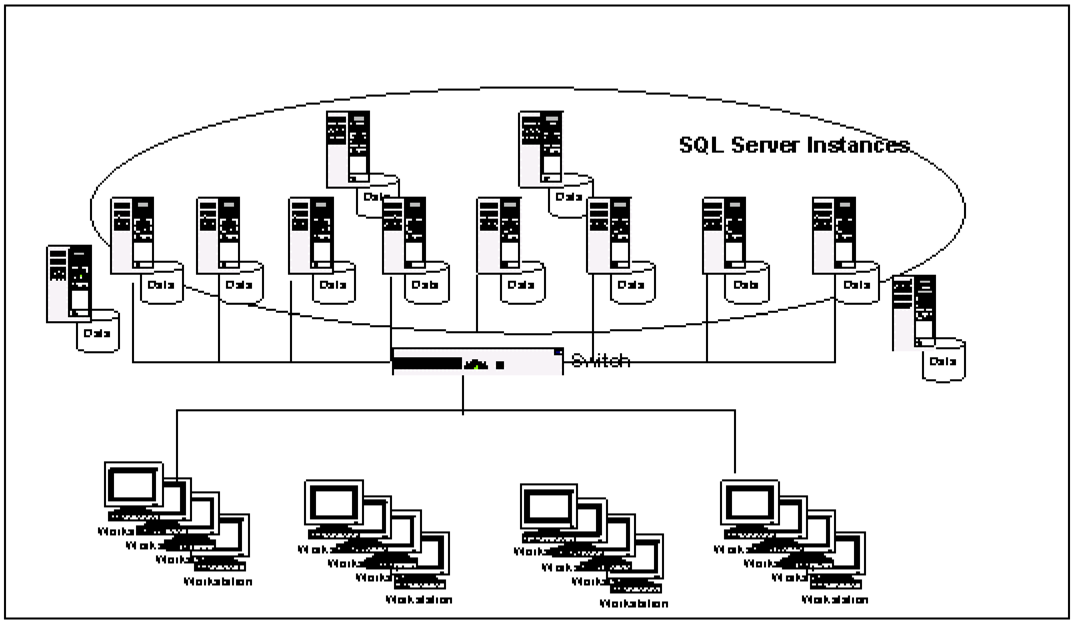


Figure 6. Before Consolidating

Consolidating the environment illustrated in figure 6 makes it easier to administrate and to share resources among applications. Consolidation projects involves different but related projects, storage consolidation (figure 7), and server consolidation (figure 8).

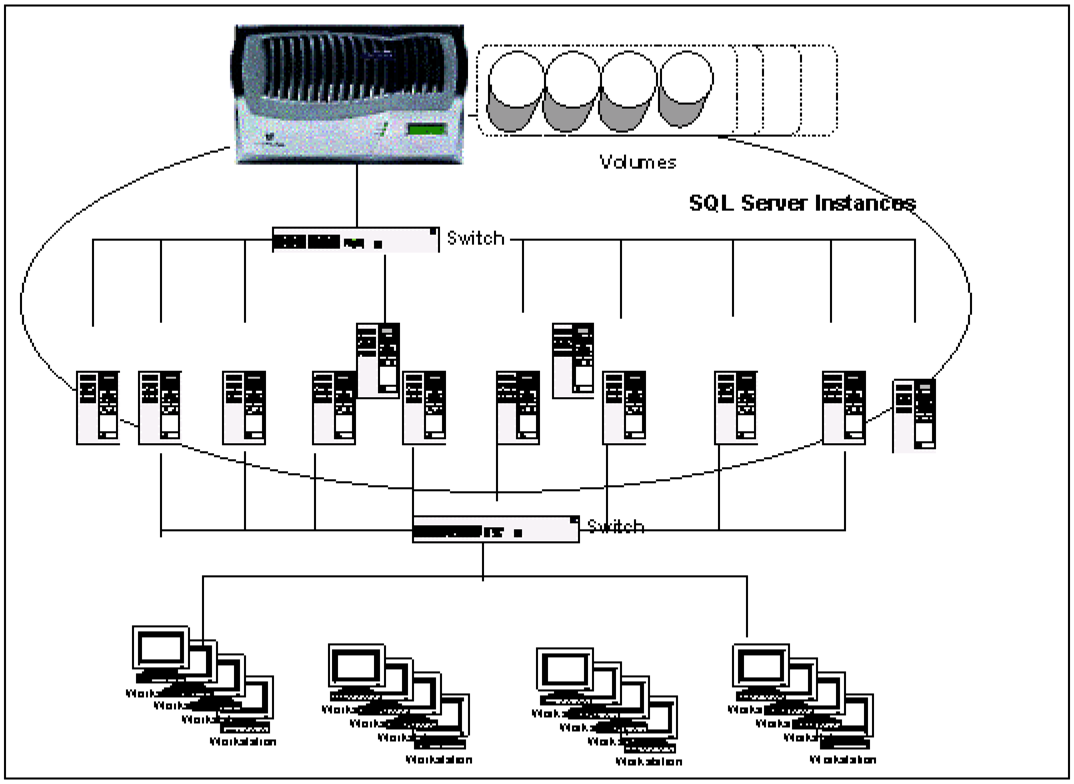


Figure 7. Storage Consolidation

Figure 7 illustrates a situation where the disk storage has been consolidated, making it much easier to share storage resources and do capacity planning among SQL Servers. Consolidating server storage works well with limited numbers of servers, such as when there is a one-to-one relationship between a server and a NetApp volume. Even though it is possible for servers to share a NetApp volume, it is absolutely not recommended with SMSQL. Backups are volume-level Snapshot copies, which makes it complicated when a single volume is shared among servers; *one volume per SQL Server platform is recommended with SMSQL*.

It is advisable when the environment consists of many servers to consolidate them so that it is possible to create a simple relationship between one server and one volume.

Figure 8 illustrates the final consolidation that consists of two SQL Server platforms and a cluster of filers.

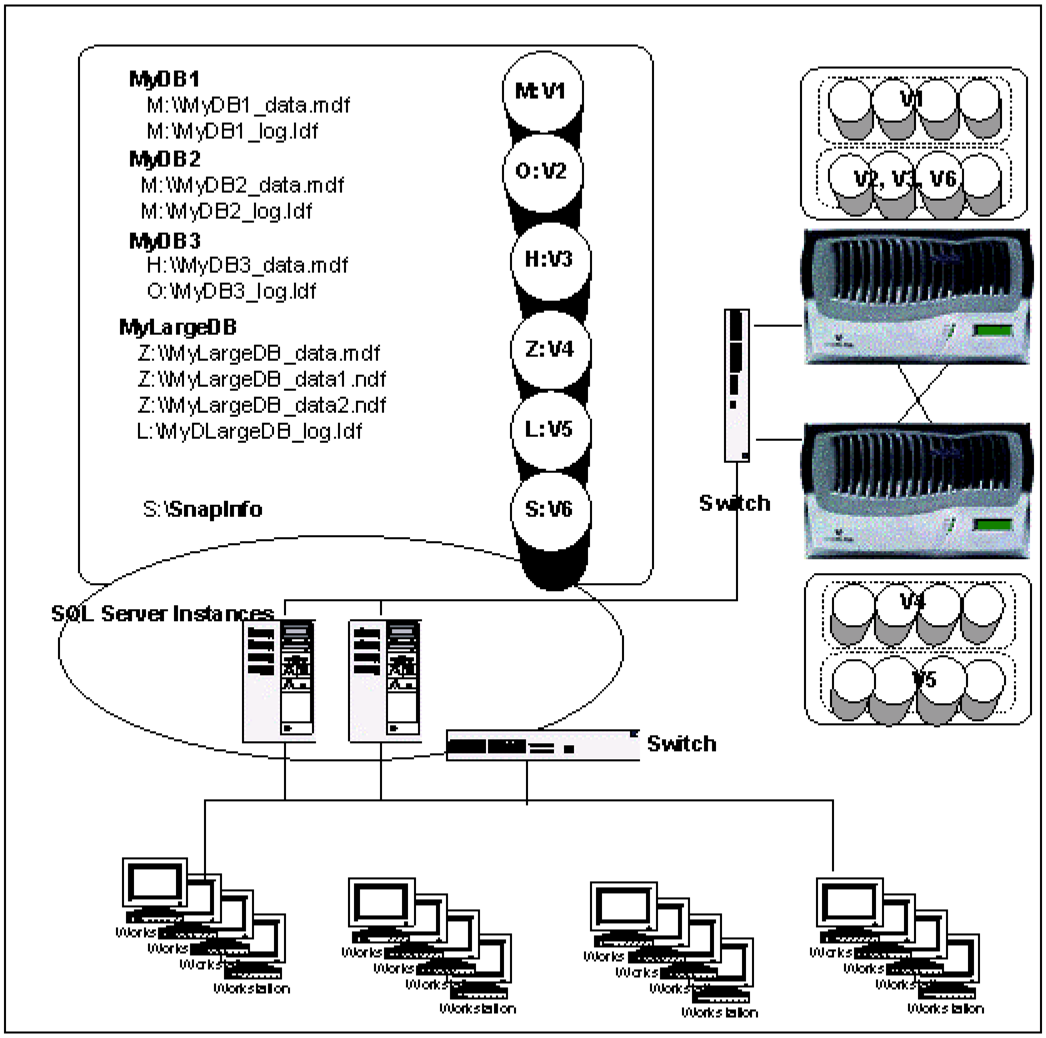


Figure 8. Complete Consolidation

Figure 8 exemplifies the total consolidation of the environment in figure 6. All locally attached disk storage has been consolidated into a cluster of filers, and the many smaller servers have been consolidated into two larger servers.

Figure 8 also illustrates important points discussed in section 2:

- A virtual disk can be shared when all files belonging to the databases are located in one virtual disk (MyDB1 and MyDB2).
- When a database's files are located in different virtual disks, the virtual disks cannot be shared with other databases' files (MyDB3 and MYLargeDB).
- SnapInfo[18] cannot be located in a virtual disk with any transaction log file. We recommend that SnapInfo be located in its own virtual disk.
- There is a mapping between a virtual disk and a drive letter; a Windows server has a maximum number of usable drive letters, usually about 20[19].

The advantages of the final consolidation are:

- Much easier to administer
- Much easier to monitor performance
- Much easier to share resources
- Much lower administration work

5. SnapMirror Requirements

SnapMirror is important for disaster recovery services and database cloning for reporting, testing, or development.

Figure 9 illustrates a SQL Server environment with one database stored in a volume that has an active SnapMirror backup. Figure 9 also illustrates one reason to use multiple SnapInfo directories—one SnapInfo for the mirrored database and another for unmirrored databases. SnapDrive will detect if a volume is mirrored during Snapshot creation, and will request that the NetApp filer update the mirror. It is also possible to update a mirror more frequently using rolling Snapshot backups (described in reference 2). Rolling Snapshot consists of two alternating Snapshot copies that are created when executing an up_SnapMirror command from SnapDrive or from its command line interface.

Two copies will be replicated: a full backup and another when the SnapInfo directory is updated. The setting "*Create Snapshot of SnapInfo drive after backup*" is on by default[20], but check to be certain it hasn't been changed.

Some points to think about when using SnapMirror:

It is recommended that all files belonging to the mirrored database(s) are located in the same volume. It is technically possible to locate databases files in different volumes and mirror each volume, but there is a slight possibility that the newest set of Snapshot backups are inconsistent in the sense that not all volume replications have completed before the disaster, in which case the administrator has to restore from an older full backup set.

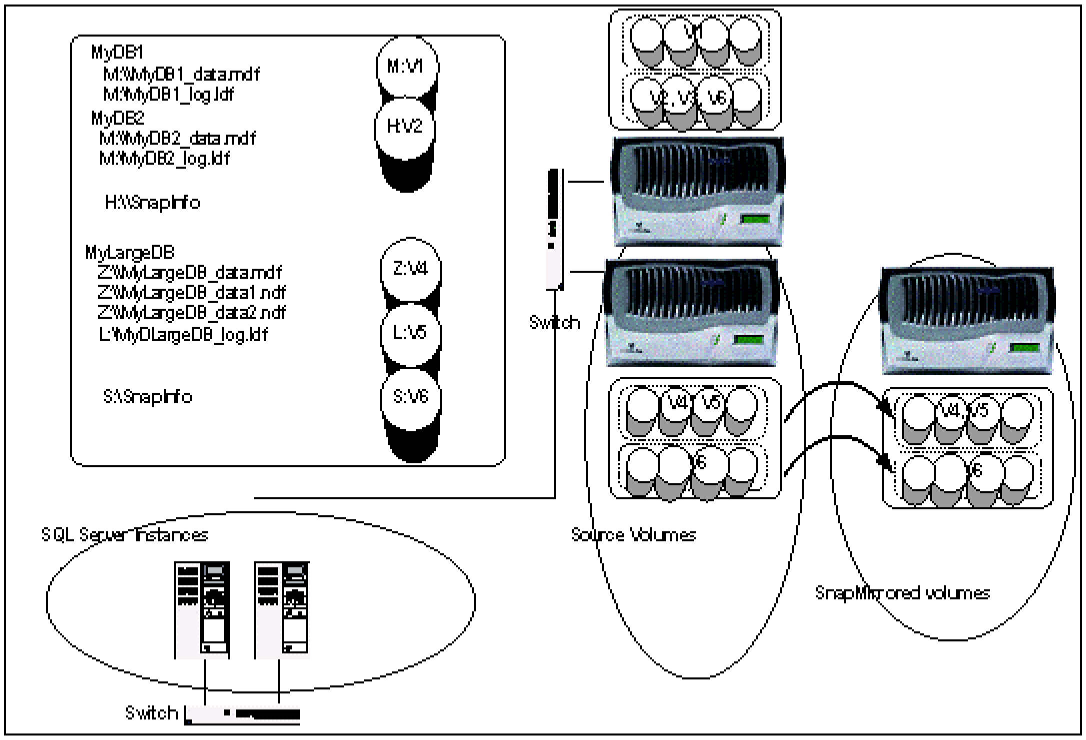


Figure 9. SnapMirror

6. SMSQL v1.0 Overview

The previous sections discussed possible valid database layouts with SMSQL. This section will discuss relationships among SMSQL, SnapDrive, and SQL Server 2000, and the maximum configurations supported by SMSQL V 1.0 will be defined.

6.1. SMSQL, SnapDrive, and SQL Server 2000

SMSQL is dependent on services provided by SnapDrive and SQL Server. SMSQL executes user requests that have to be coordinated with SnapDrive and SQL Server 2000. See figure 10.

1. Full backup, backup, and truncation of transaction logs are synchronized between SMSQL and SQL Server. SQL Server will quiesce database(s) and dump metadata to SnapInfo when a Snapshot copy is created. Backup and truncation of transaction logs are requested by SMSQL but done by SQL Server, and are dumped to SnapInfo by SQL Server. Database(s) are verified by SQL Server after the virtual disk containing the database(s) has been restored as a writable Snapshot file by SnapDrive.

2. SMSQL will request that SnapDrive create a Snapshot backup when a full backup of a database(s) is done. A copy of backed up transaction logs will be created if that option is turned on. SMSQL will dump metadata describing backup sets into the SnapInfo directory.

3. If a volume is mirrored, SnapDrive will update the mirror every time a Snapshot copy is created. SnapDrive will restore virtual disks in Snapshot backups as required by database restore operations, or it will restore a virtual disk as a writable copy for database verification.

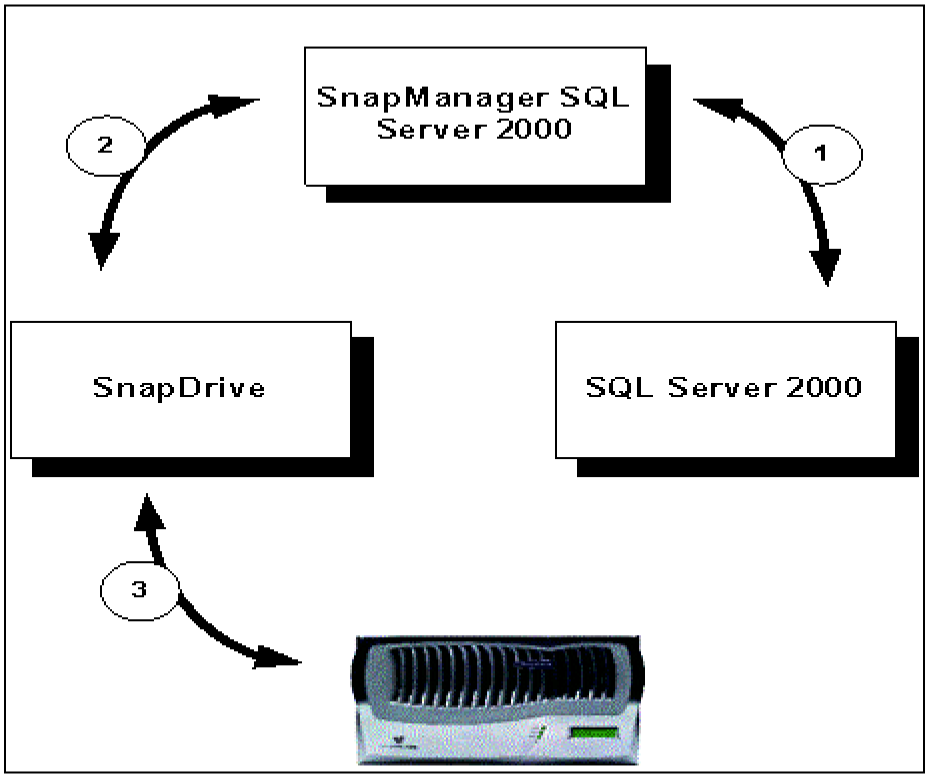


Figure 10. SMSQL Overview

6.2. Maximum Configuration

SMSQL has been designed to manage a maximum number of SQL Server instances, databases, files, and file groups, which are:

- Maximum number of SQL Server 2000 instances per server: 4
- Maximum of 253 databases per server
- Maximum of 253 file groups per server
- Maximum of 503 database files per server
- Maximum number of databases stored in one single virtual disk: 64
- Maximum volume that can be used by a single database: 4
- Total number of volumes that that can be used by a server: 8
- 10 file groups per database

7. Database Migration

The configuration wizard will move databases and their files to NetApp storage. The configuration wizard has to be executed even when databases are already located on virtual disk before SMSQL is installed. The configuration wizard will set up SnapInfo[23] and verify settings to ensure that all SMSQL functions will work correctly. Even if no database has to be moved to NetApp storage, the configuration wizard has to be executed to create the mapping between databases and SnapInfo.

Sections 2 and 4 discussed important background information for migrating databases. If the project is part of a consolidation project, section 4 contains important information related to consolidation. If SnapMirror will be used then section 4.1 is very relevant.

7.1. Migration Basics

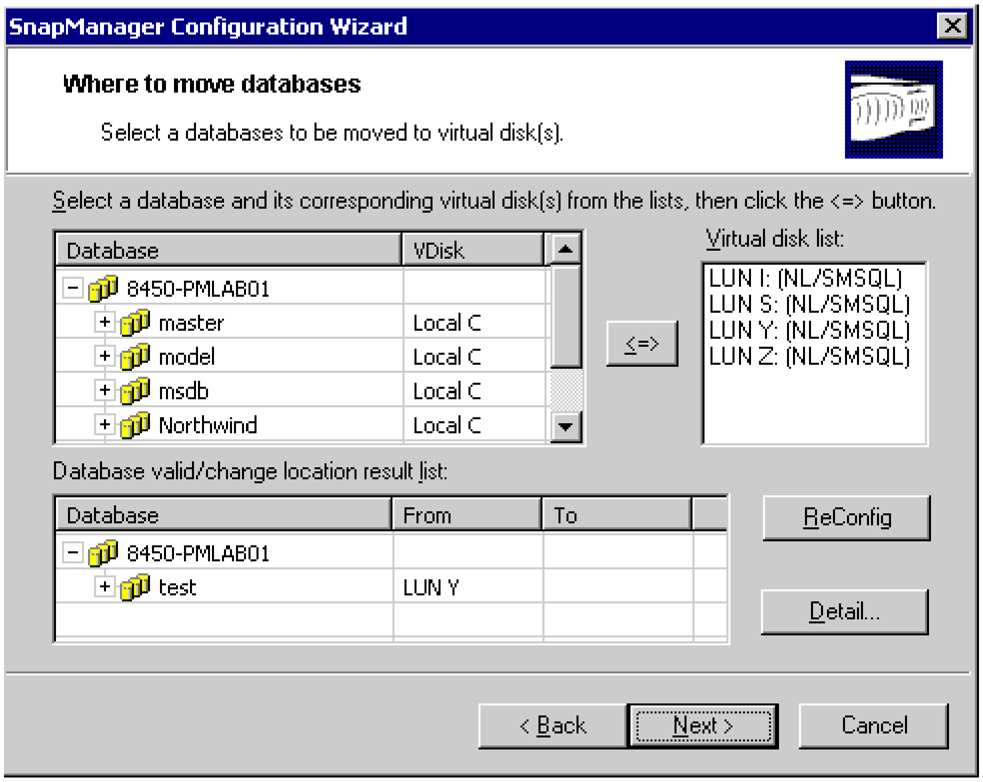
The configuration wizard's first central screen is "where to move databases"; figure 11. The screen consists of three panels:

1. The top left panel shows SQL Server instances, and under each instance it shows databases that can be moved to virtual disks but still be located on other than virtual disks. The panel shows one SQL Server instance (8450-PMLAB01) and the number of databases located on the C drive.

2. The top right panel shows virtual disks that are available for moving databases to, or can be used by, SnapInfo. The panel shows four virtual disks: I, S, Y, and Z.

3. The bottom panel shows SQL Server instances with databases stored on virtual disks. The database "test" is located on the Y drive. The From and To columns can easily be misunderstood. Think of From as the current location from which the database will be moved only if To shows a different drive letter. The "test" database was already on Y when the screen was opened, and 'test' will not be moved.

Figure 11 illustrates one point that was discussed in section 2. A virtual disk can be shared by several databases when all files belonging to the databases are located in the one virtual disk. All files belonging to "test" are located in Y, so other databases can be added to Y. That is why it is shown as an available drive in the top right panel.


SnapManager Configuration Wizard

Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
8450-PMLAB01	
master	Local C
model	Local C
msdb	Local C
Northwind	Local C

Virtual disk list:

LUN I: (NL/SMSQL)
LUN S: (NL/SMSQL)
LUN Y: (NL/SMSQL)
LUN Z: (NL/SMSQL)

Database valid/change location result list:

Database	From	To
8450-PMLAB01		
test	LUN Y	

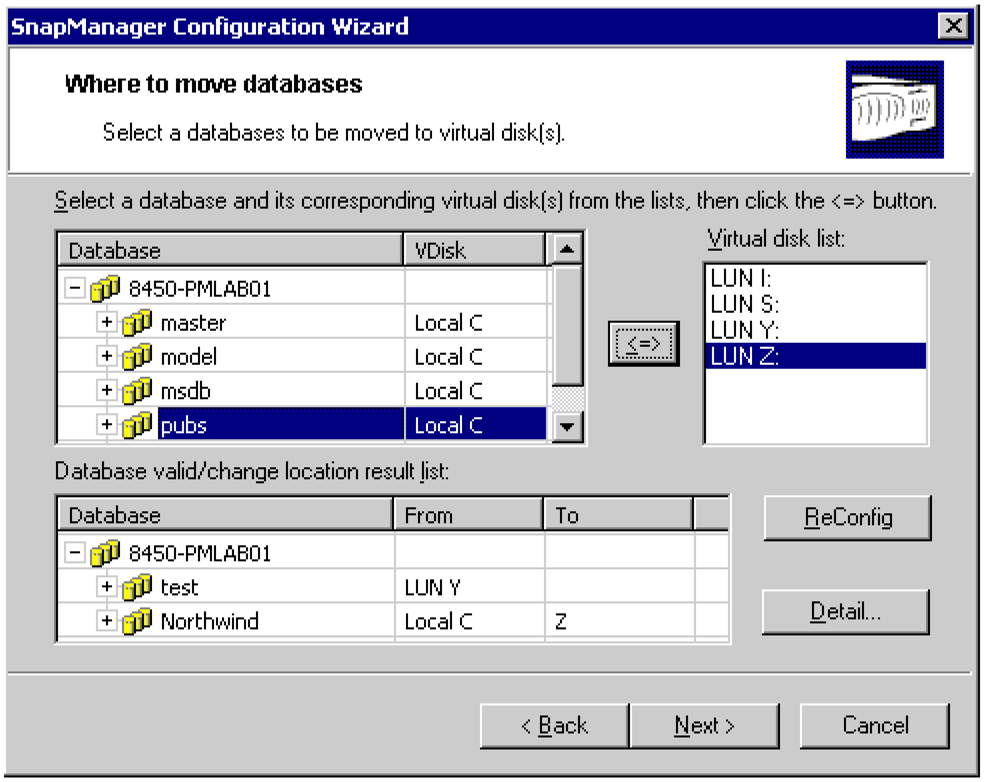
Figure 11. Where to Move Databases

7.2. Migrating Databases to One Virtual Disk

Migrating a database to a single virtual disk is easy (figure 12):

1. Highlight the database that you want to move.
2. Highlight the virtual disk that the database will move to.
3. Click on '<-->' and the database will be moved to the lower panel

A database's files can be spread over several virtual disks as discussed in section 2. Section 7.3 will illustrate how to move a database to a number of virtual disks.



SnapManager Configuration Wizard

Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
8450-PMLAB01	
master	Local C
model	Local C
msdb	Local C
pubs	Local C

Virtual disk list:

LUN I:
LUN S:
LUN Y:
LUN Z:

Database valid/change location result list:

Database	From	To
8450-PMLAB01		
test	LUN Y	
Northwind	Local C	Z

Figure 12. Where to Move Databases

7.3. Migrating One Database to Multiple Virtual Disks

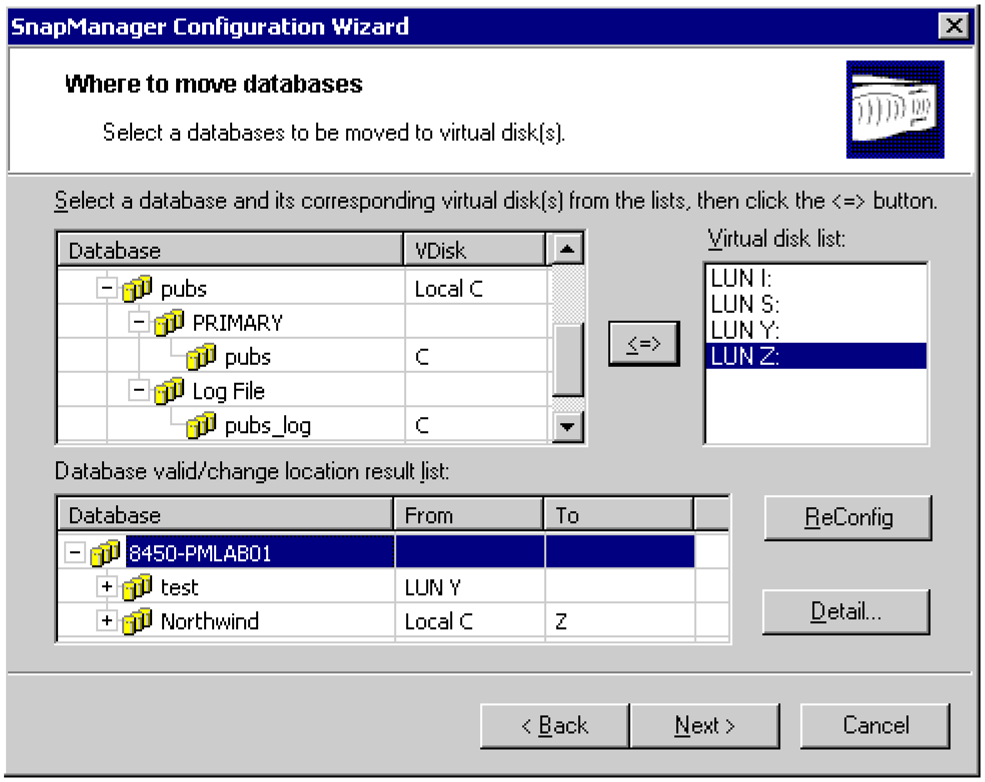
Migrating a database's filer to different virtual disks is nearly identical to migrating to a single virtual disk. But let us first discuss a few reasons for spreading a database over multiple virtual disks. A common issue when traditional local disks are used for storing database files is load balancing the disk I/O over available disk drives. When a filer is used for storing databases, Data ONTAP™ will automatically load balance I/O evenly over available disks[24]. Most databases stored on NetApp filers utilize just one virtual disk, but there are exceptions and reasons for using multiple virtual disks using different NetApp volumes:

- The transaction log file and data file are traditionally located in different devices, primarily for safety reasons. If the volume with the data files has a double disk failure, then the transaction log file is still available. Double disk failure happens very rarely and the transaction log is only useful if the full backup and backed up transaction logs have been archived.

- The access to the transaction log is sequential and the access to data file(s) is mostly random. Therefore, in a case of poor performance because the volume's disk drives have high utilization but low throughput, performance can be improved by moving the transaction log to another virtual disk located in another volume and thereby decreasing the disk's utilization.

- A volume has to be large enough to contain a database(s) physical size, growth, and Snapshots backups. There is no exact point when a database becomes too big for one volume, but it is appropriate to start planning and to analyze the I/O performance and space requirements when the database size is around 700GB to estimate if moving to two volumes is appropriate.

- If the database consists of several very large tables with different access patterns, locating each table in different file groups and locating each file group in different volumes will potentially increase I/O throughput.


SnapManager Configuration Wizard
Where to move databases
Select a databases to be moved to virtual disk(s).
Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.
Virtual disk list:
<=>
Database valid/change location result list:
ReConfig
Detail...
< Back
Next >
Cancel

Database	VDisk
pubs	Local C
PRIMARY	
pubs	C
Log File	
pubs_log	C

LUN I:
LUN S:
LUN Y:
LUN Z:

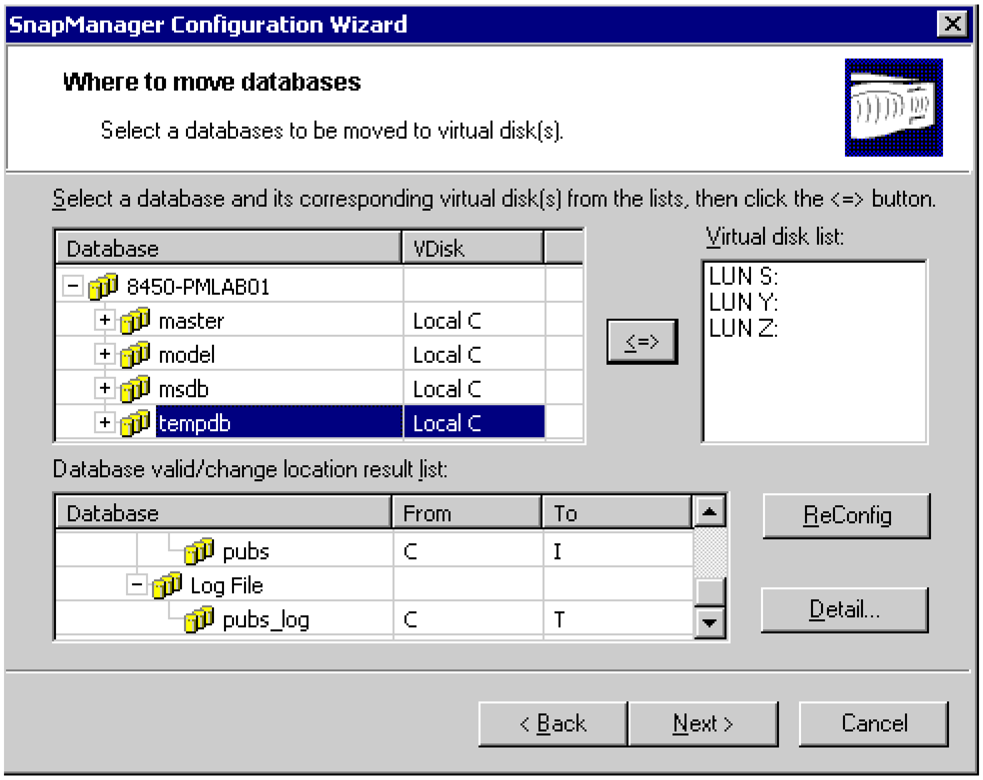
Database	From	To
8450-PMLAB01		
test	LUN Y	
Northwind	Local C	Z

Figure 13. Where to Move Databases

Migrating a database to multiple virtual disks requires two steps per database file. Figure 13 shows the expanded pubs database with its two files, `pubs` and `pubs log`, located on the C drive. The following steps illustrate how to move the `pubs` data file to the I drive and the `pubs_log` file to the T drive:

1. Expand pubs database so that all file groups and files are visible (figure 13).
2. Highlight the pubs data file.
3. Highlight the I drive.
4. Click on <--> and the pubs database with the data file will be moved down to the lower panel.
5. Highlight the `pubs_log` file.
6. Highlight the T drive.
7. Click on <--> and all of the pubs databases will be moved down to the lower panel (figure 14).

Figure 14 illustrates an important point that was discussed in section 2. When a database occupies more than one virtual disk, those virtual disks cannot be shared by any other database. Figure 14 shows that the I and T drives have been removed from the available drive list (top right panel).



SnapManager Configuration Wizard

Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
8450-PMLAB01	
master	Local C
model	Local C
msdb	Local C
tempdb	Local C

Virtual disk list:

LUN S:
LUN Y:
LUN Z:

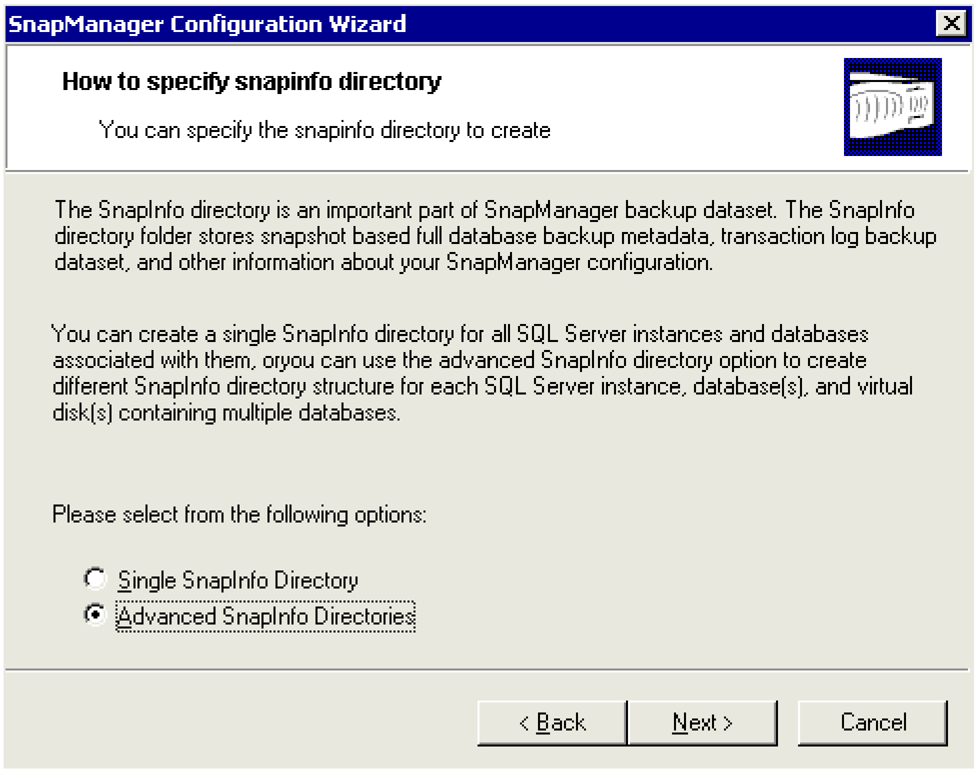
Database valid/change location result list:

Database	From	To
pubs	C	I
Log File		
pubs_log	C	T

Figure 14. Where to Move Databases

7.4. SnapInfo

SnapInfo is a directory structure that contains metadata related to backup sets, copies of backed up transaction logs, and streaming data when streaming backup is used. The default selection is to use a single SnapInfo directory for all databases managed by SMSQL, but it is possible to configure multiple SnapInfo directories by selecting the *Advanced SnapInfo Directories*; figure 15.



How to specify snapinfo directory

You can specify the snapinfo directory to create

The SnapInfo directory is an important part of SnapManager backup dataset. The SnapInfo directory folder stores snapshot based full database backup metadata, transaction log backup dataset, and other information about your SnapManager configuration.

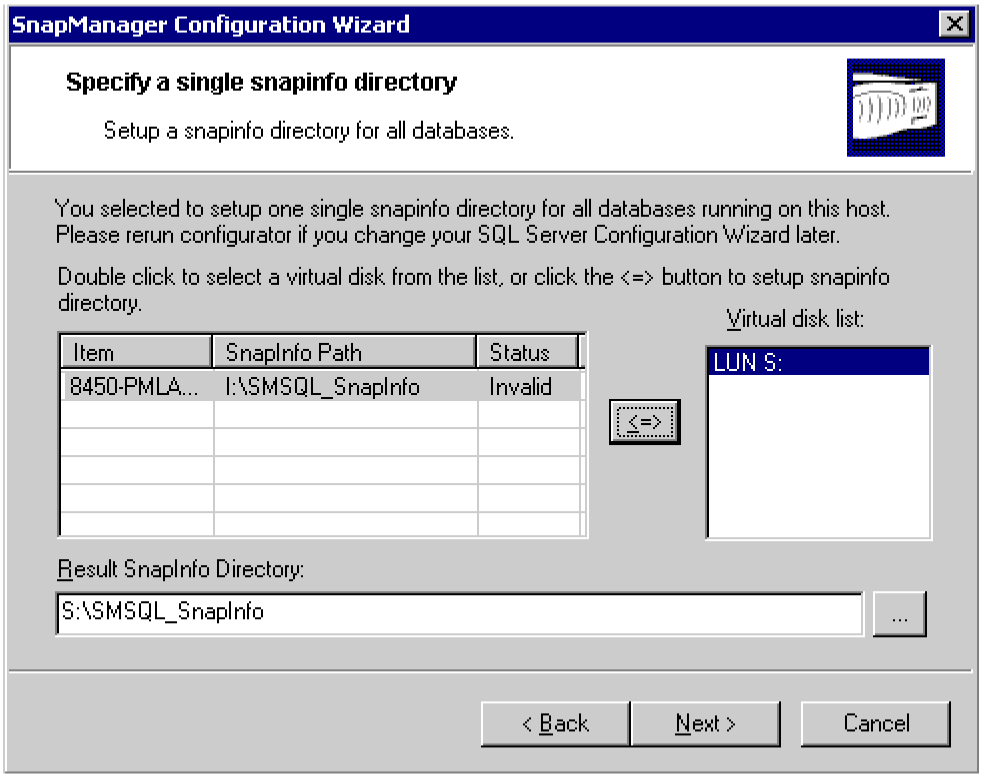
You can create a single SnapInfo directory for all SQL Server instances and databases associated with them, oryou can use the advanced SnapInfo directory option to create different SnapInfo directory structure for each SQL Server instance, database(s), and virtual disk(s) containing multiple databases.

Please select from the following options:

- ○ Single SnapInfo Directory
- ● Advanced SnapInfo Directories

Figure 15. Specify SnapInfo Type

The default selection *Single SnapInfo Directory* will display a single screen (figure 16), whereas the advanced selection will display three different screens. The advanced selection is flexible so the administrator can configure the SMSQLs according to the local environment; such as using different SnapInfos for databases in a volume that is being mirrored by SnapMirror, and databases using another volume that is not mirrored.



Specify a single snapinfo directory

Setup a snapinfo directory for all databases.

You selected to setup one single snapinfo directory for all databases running on this host. Please rerun configurator if you change your SQL Server Configuration Wizard later.

Double click to select a virtual disk from the list, or click the <=> button to setup snapinfo directory.

Item	SnapInfo Path	Status
8450-PMLA...	I:\SMSQL_SnapInfo	Invalid

Virtual disk list:

LUN S:

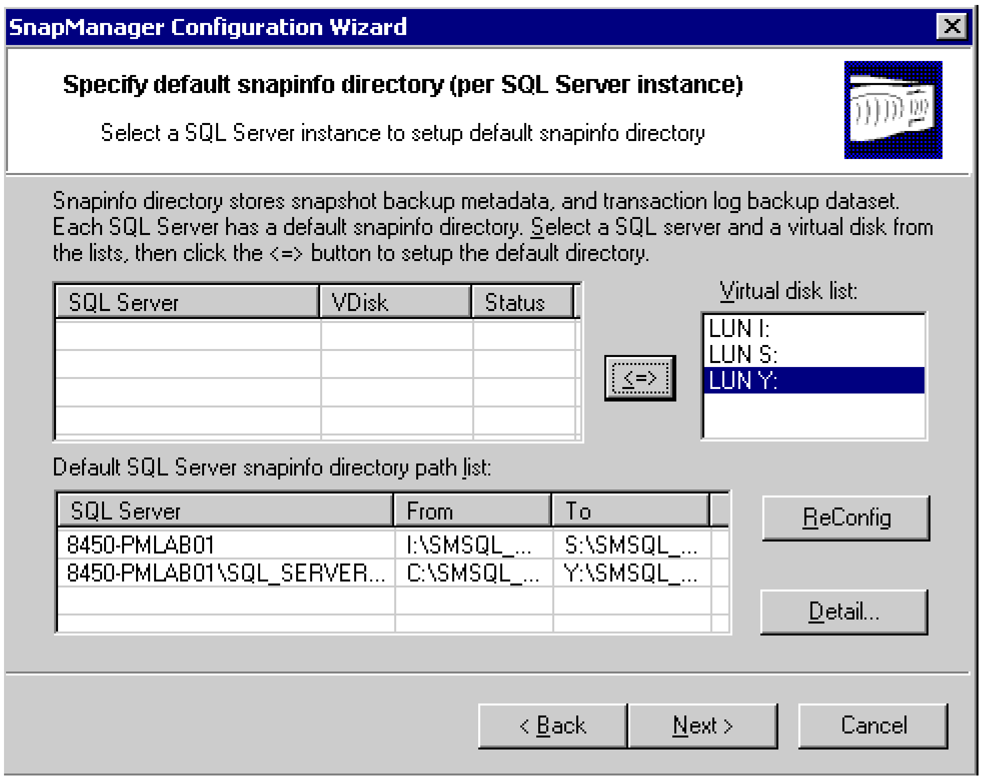
Result SnapInfo Directory:

S:\SMSQL_SnapInfo

Figure 16. Specify Single SnapInfo Directory

The first screen displayed when the advanced selection is activated makes it possible to select different SnapInfo directories for each SQL Server instance. The example in figure 17 includes two SQL Server instances:

- SQL Server instance 8450-PMLAB01 has been configured with a SnapInfo directory on the S drive.
- SQL Server instance `SQL_Server_2` has been configured with a SnapInfo directory on the Y drive.



SnapManager Configuration Wizard

Specify default snapinfo directory (per SQL Server instance)

Select a SQL Server instance to setup default snapinfo directory

Snapinfo directory stores snapshot backup metadata, and transaction log backup dataset. Each SQL Server has a default snapinfo directory. Select a SQL server and a virtual disk from the lists, then click the <=> button to setup the default directory.

SQL Server	VDisk	Status

Virtual disk list:

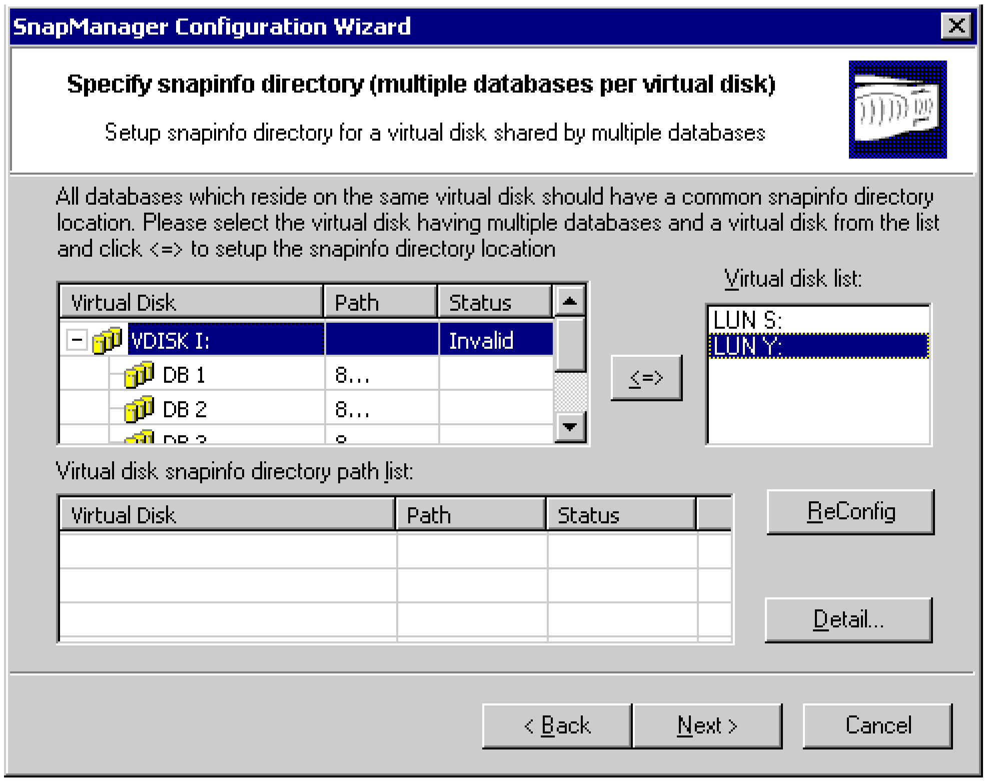
- LUN I:
- LUN S:
- LUN Y:

Default SQL Server snapinfo directory path list:

SQL Server	From	To
8450-PMLAB01	I:\SMSQL_...	S:\SMSQL_...
8450-PMLAB01\SQL_SERVER...	C:\SMSQL_...	Y:\SMSQL_...

Figure 17. SnapInfo per SQL Server Instance

The next screen configures SnapInfo for virtual disks that contain multiple databases. Figure 18 shows virtual disk I contains three databases and will be configured with SnapInfo on the Y drive.



SnapManager Configuration Wizard

Specify snapinfo directory (multiple databases per virtual disk)

Setup snapinfo directory for a virtual disk shared by multiple databases

All databases which reside on the same virtual disk should have a common snapinfo directory location. Please select the virtual disk having multiple databases and a virtual disk from the list and click <=> to setup the snapinfo directory location

Virtual Disk	Path	Status
VDISK I:		Invalid
DB 1	8...	
DB 2	8...	
DB 3	8	

Virtual disk list:

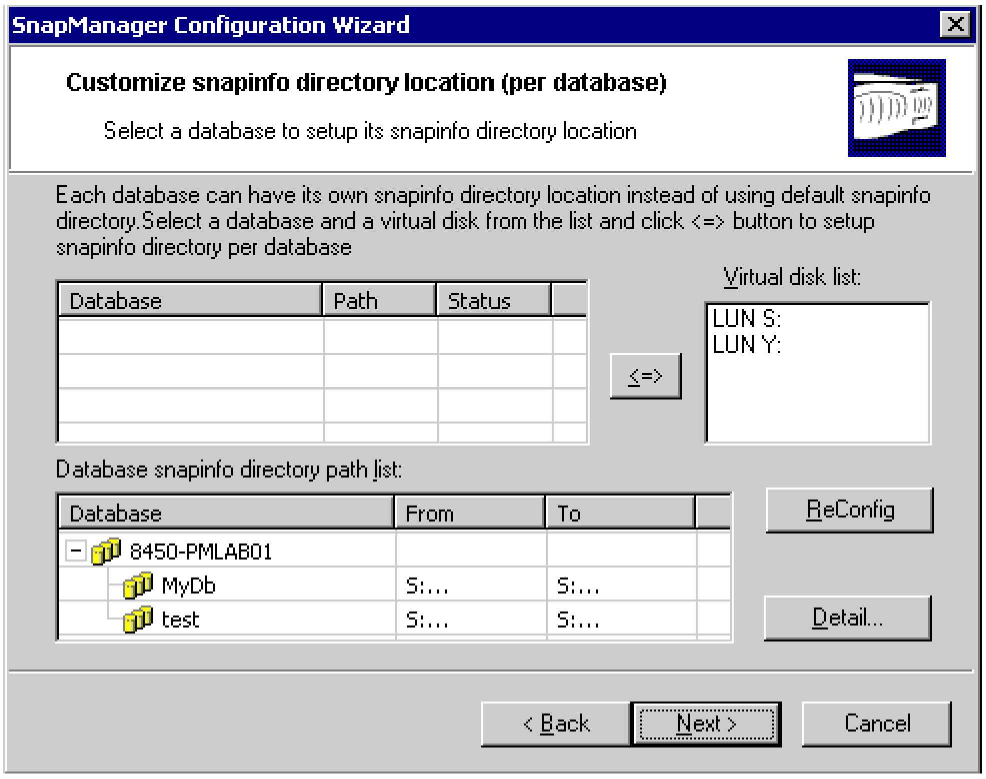
LUN S:
LUN Y:

Virtual disk snapinfo directory path list:

Virtual Disk	Path	Status

Figure 18. SnapInfo per Virtual Disk

The third SnapInfo setup screen is shown in figure 19. This process makes it possible to configure SnapInfo directories for each database that doesn't share any virtual disk with any other database.



SnapManager Configuration Wizard

Customize snapinfo directory location (per database)

Select a database to setup its snapinfo directory location

Each database can have its own snapinfo directory location instead of using default snapinfo directory.Select a database and a virtual disk from the list and click <=> button to setup snapinfo directory per database

Database	Path	Status

Virtual disk list:

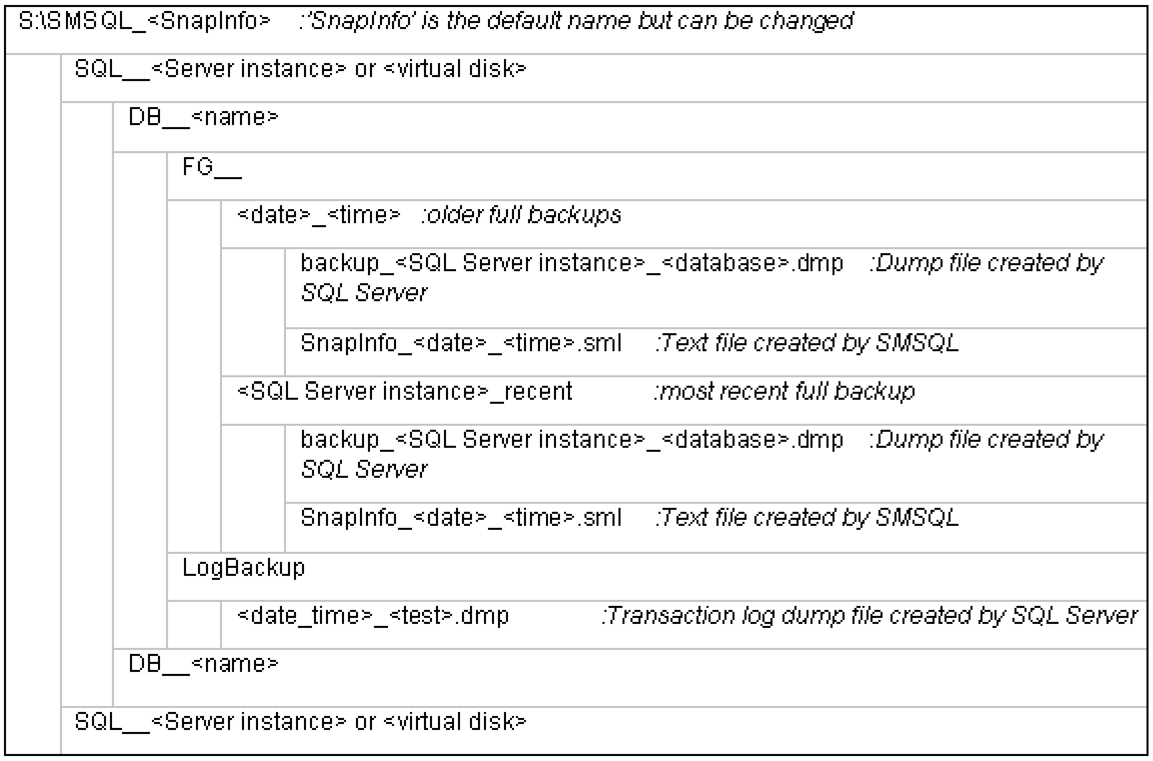
LUN S:
LUN Y:

Database snapinfo directory path list:

Database	From	To
8450-PMLAB01		
MyDb	S:...	S:...
test	S:...	S:...

Figure 19. SnapInfo per Database

What SnapInfo contains and its structure are shown in figure 20. The structure is fixed but most of the names depend on the environment. The top level is the SnapInfo directory, the second level is either the name of the SQL Server instance or the name of the virtual disk, followed by the database names under which backup information is located. The main information is under each database's subdirectories: FG__ (if a full backup has been created) and LogBackup (if the transaction log has been backed up). The information in figure 20 is important to know if a disaster occurs and the database has to be restored from the archive. See reference 2.

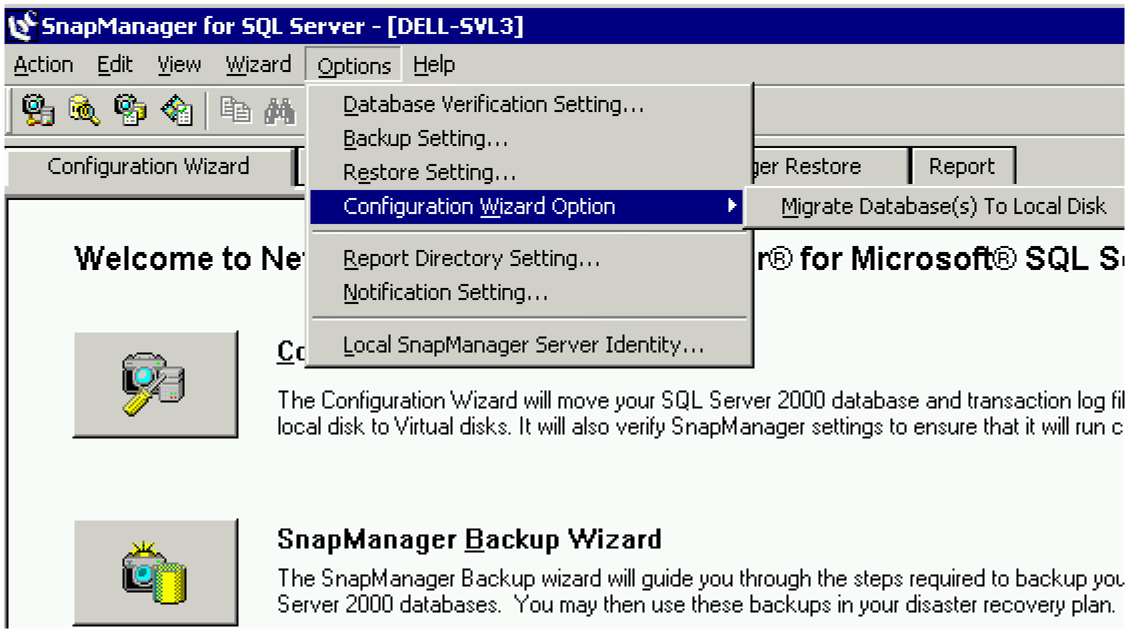


```
S:\SMSQL_<SnapInfo>    :'SnapInfo' is the default name but can be changed
    SQL__<Server instance> or <virtual disk>
        DB__<name>
            FG__
                <date>_<time>   :older full backups
                    backup_<SQL Server instance>_<database>.dmp    :Dump file created by SQL Server
                    SnapInfo_<date>_<time>.sml    :Text file created by SMSQL
                <SQL Server instance>_recent    :most recent full backup
                    backup_<SQL Server instance>_<database>.dmp    :Dump file created by SQL Server
                    SnapInfo_<date>_<time>.sml    :Text file created by SMSQL
            LogBackup
                <date_time>_<test>.dmp    :Transaction log dump file created by SQL Server
        DB__<name>
    SQL__<Server instance> or <virtual disk>
```

Figure 20. Structure of SnapInfo

7.5. Migrating Back to Local Disk

Migrating databases from local disks to virtual disks is, by default, the expected action, but it is possible to turn on an option that makes it workable to migrate databases back to local disks. The option is turned on by selecting Configuration Wizard Option from the Tools menu and then selecting Migrating Database(s) Back to Local Disk; see figure 21. After the option has been set, local disks will be visible and selectable; see figure 22.


SnapManager for SQL Server - [DELL-SVL3]
Action Edit View Wizard Options Help
Database Verification Setting...
Backup Setting...
Restore Setting...
Configuration Wizard Option
Migrate Database(s) To Local Disk
Report Directory Setting...
Notification Setting...
Local SnapManager Server Identity...
Configuration Wizard
ger Restore
Report

Welcome to Ne r® for Microsoft® SQL S

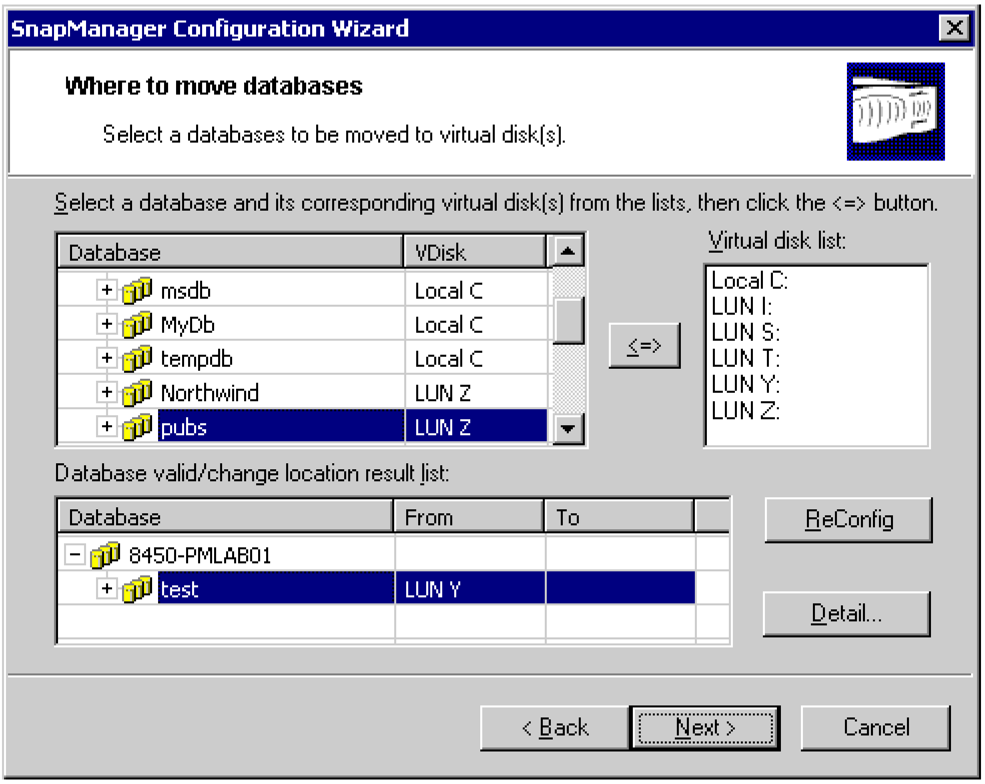
Co

The Configuration Wizard will move your SQL Server 2000 database and transaction log fil local disk to Virtual disks. It will also verify SnapManager settings to ensure that it will run c

SnapManager Backup Wizard

The SnapManager Backup wizard will guide you through the steps required to backup you Server 2000 databases. You may then use these backups in your disaster recovery plan.

Figure 21. Migrating Back to Local Disk Option



Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
msdb	Local C
MyDb	Local C
tempdb	Local C
Northwind	LUN Z
pubs	LUN Z

Virtual disk list:

Local C:
LUN I:
LUN S:
LUN T:
LUN Y:
LUN Z:

Database valid/change location result list:

Database	From	To
8450-PMLAB01		
test	LUN Y	

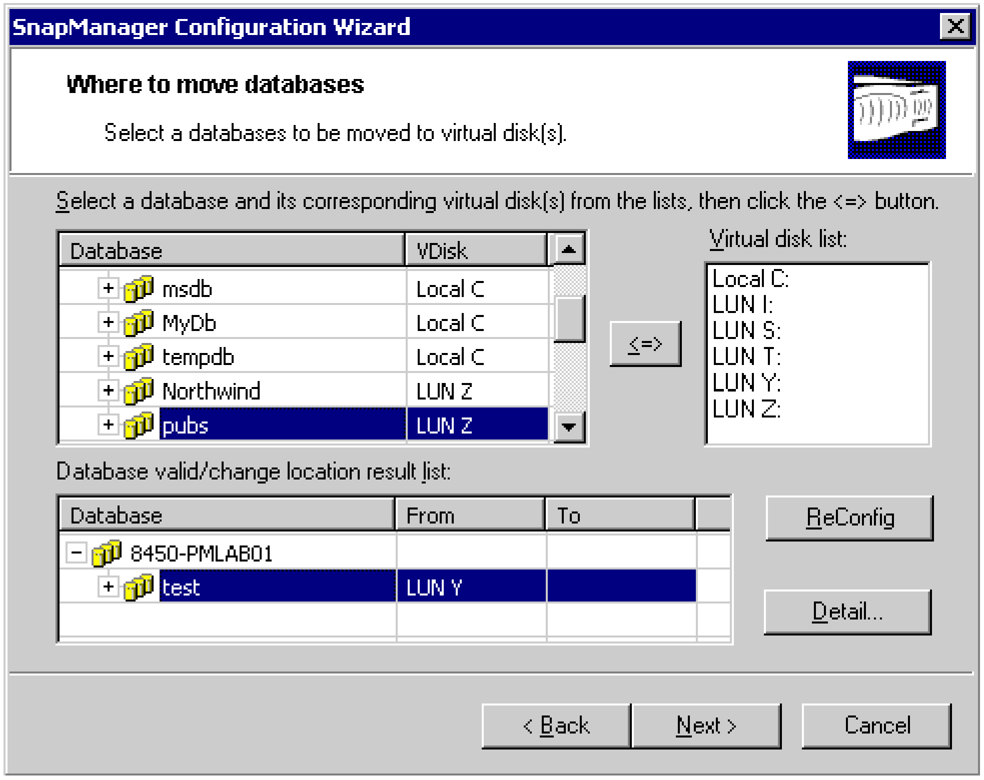
Figure 22. Migrate Back to Local Disk

Migrating databases back to local disks[25] can be performed only when they are present in the upper left panel, and each migrating database has to end up in the bottom panel. Therefore, a database that is already located on a virtual disk and is located in the bottom panel is reconfigured by first moving it to the top left panel. The following example will migrate pubs and Northwind from the Z (figure 22) drive to the C drive.

1. Highlight pubs in the lower panel and click on ReConfig.

2. Highlight Northwind in the lower panel and click on ReConfig.

3. Both databases are now located in the upper left panel with Z as the location (figure 23).



SnapManager Configuration Wizard

Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
msdb	Local C
MyDb	Local C
tempdb	Local C
Northwind	LUN Z
pubs	LUN Z

Virtual disk list:

Local C:
LUN I:
LUN S:
LUN T:
LUN Y:
LUN Z:

Database valid/change location result list:

Database	From	To
8450-PMLAB01		
test	LUN Y	

1.

Figure 23. Migrating Back to Local Disk

4. Highlight pubs in the left panel.

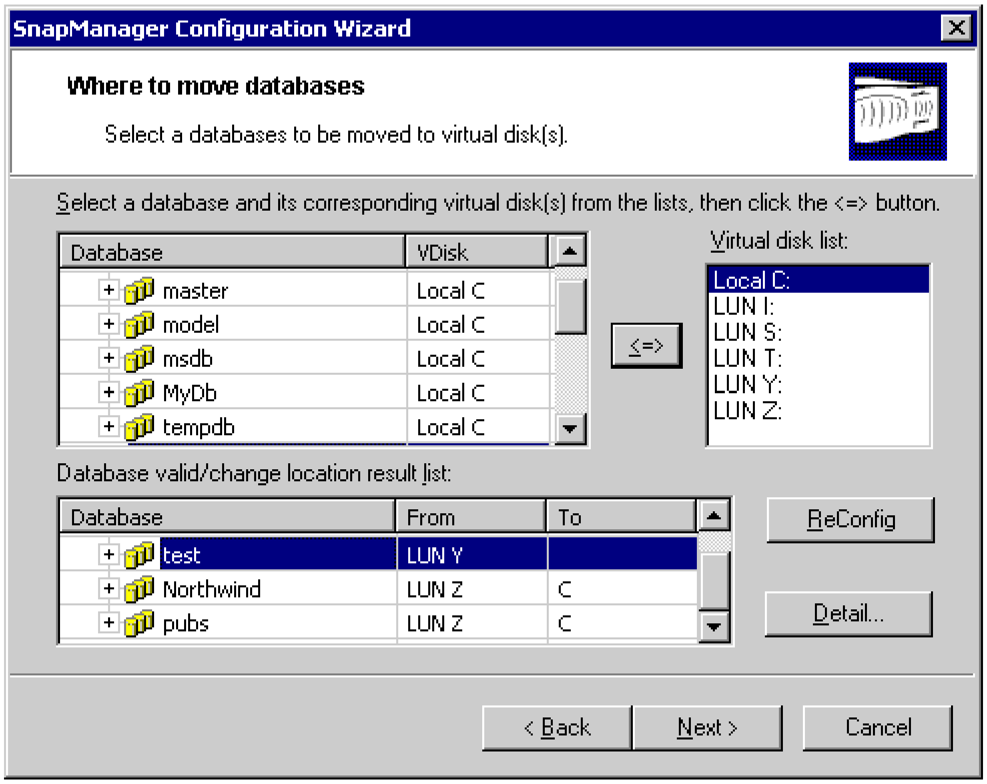
5. Highlight Local C in the right panel.

6. Click on <-->.

7. Repeat steps 4 to 6 with Northwind.

8. Pubs and Northwind have moved down to the lower panel; see figure 24. Notice that the "From" is Z and "To" is C.

9. Click Next and continue until the process is finished.



SnapManager Configuration Wizard

Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
master	Local C
model	Local C
msdb	Local C
MyDb	Local C
tempdb	Local C

Virtual disk list:

- Local C:
- LUN I:
- LUN S:
- LUN T:
- LUN Y:
- LUN Z:

Database valid/change location result list:

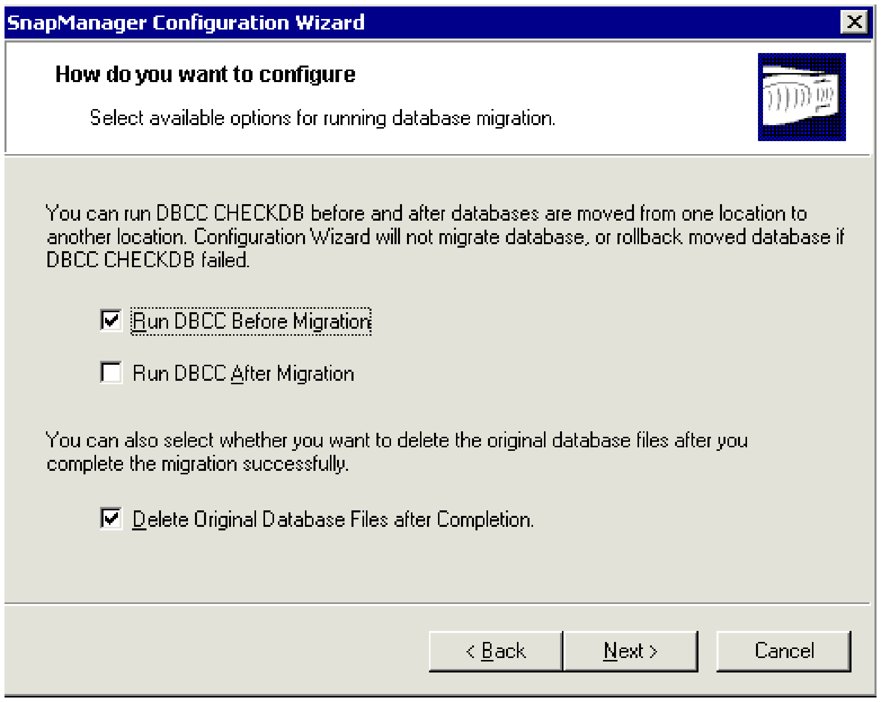
Database	From	To
test	LUN Y	
Northwind	LUN Z	C
pubs	LUN Z	C

Figure 24. Migrating Back to Local Disk

8. Database Consistency: DBCC CHECKDB

It is strongly recommended that you check a database's consistency before it is migrated. There is nothing more frustrating than discovering that a backed up database is inconsistent only because the original source was inconsistent.

There are a couple of screens related to verifying databases for consistency. The first screen is displayed during the initial setup by configuration wizard; see figure 25. The default is to run DBCC before the database is migrated, which is the recommended option. Verifying a database takes resources away from the production environment, therefore, offloading verification to another remote SQL Server instance is recommended.



SnapManager Configuration Wizard

How do you want to configure

Select available options for running database migration.

You can run DBCC CHECKDB before and after databases are moved from one location to another location. Configuration Wizard will not migrate database, or rollback moved database if DBCC CHECKDB failed.

[✓] Run DBCC Before Migration

[ ] Run DBCC After Migration

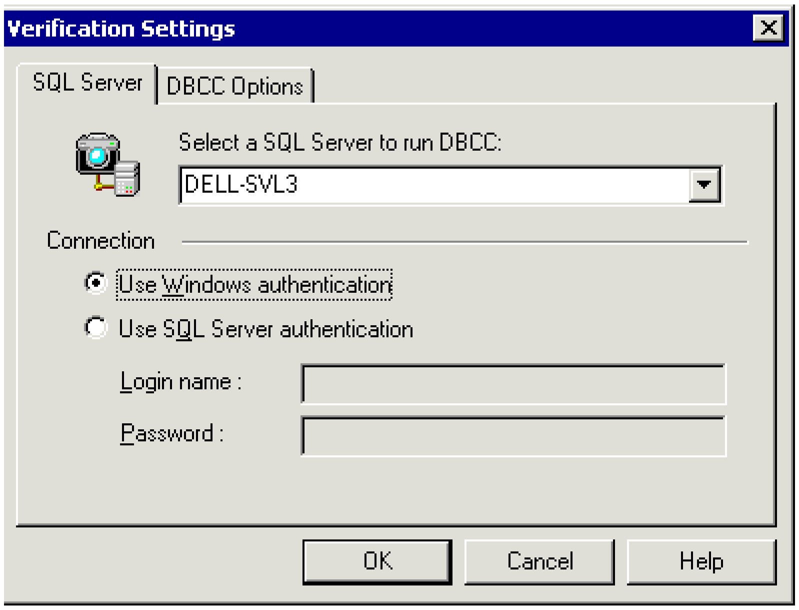
You can also select whether you want to delete the original database files after you complete the migration successfully.

[✓] Delete Original Database Files after Completion.

Figure 25. Wizard's DBCC Options

General verification settings used during backup and recovery setting are defined in SMSQL's options menu:

1. Highlight the Options menu.
2. Select Database Verification Setting; figure 26 will be displayed.



Verification Settings

Select a SQL Server to run DBCC: DELL-SVL3

Connection

[✓ Use Windows authentication] Use SQL Server authentication

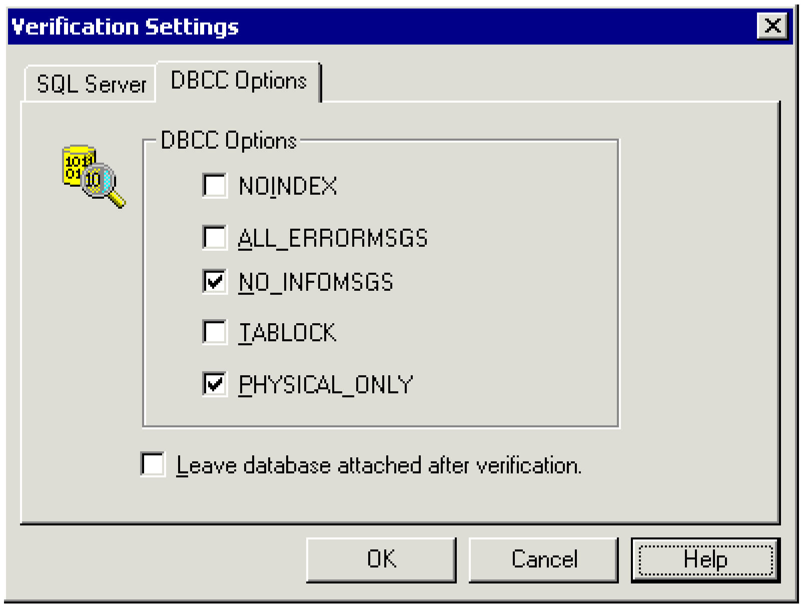
Login name :

Password :

Figure 26. DBCC Verification Settings

3. Browse to the SQL Server that will run DBCC that can be a remote server.

4. Open tab: DBCC Options (figure 27).



DBCC Options
- [ ] NOINDEX
- [ ] ALL_ERRORMSGS
- [x] NO_INFOMSGS
- [ ] TABLOCK
- [x] PHYSICAL_ONLY

- [ ] Leave database attached after verification.

Figure 27. DBCC CHECKDB Options

5. DBCC's default verification options are PHYSICAL_ONLY and NO_INFORMSG.

Verifying a database takes resources and time but PHYSICAL_ONLY takes the least time. Even so, it is not a quick operation. Other possible verification options are shown in figure 27.

Possible DBCC CHECKDB options are:

- NOINDEX does not check table indexes on user tables.

- TABLOCK causes DBCC CHRCKDB to obtain shared table locks. TABLOCK will cause DBCC to run faster on a database under heavy load, but decreases the concurrency available on the database while DBCC CEHECDB is running. A table lock will allow the transaction to read table objects but not to insert or change data before DBCC has finished checking that table.

- PHYSICAL_ONLY limits the checking on the integrity of the physical structure of all pages, the record header, and the id between the pages' ID and index ID. Also it detects torn pages.

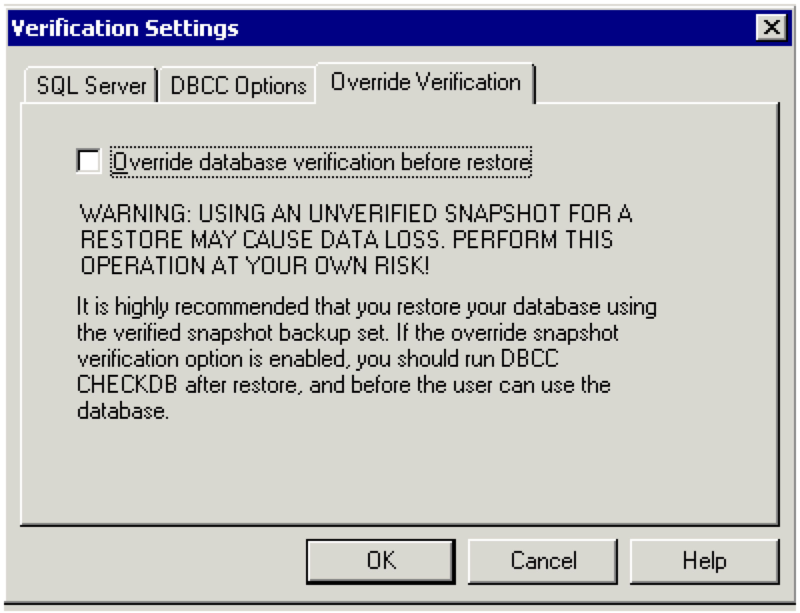
SMSQL will by default prevent a backed up database from being restored without checking its consistency, which in an emergency can take too much time to be functional. Therefore, SMSQL has an override option for an emergency, but it is strongly recommended to check the database's consistency as soon as possible. The override option is available when:

The SnapManager Restore tab is activated:

1. Highlight the Options menu.

2. Select Database Verification Setting; figure 28 will be displayed.

Notice the warning on the screen in figure28!



☐ Override database verification before restore

WARNING: USING AN UNVERIFIED SNAPSHOT FOR A RESTORE MAY CAUSE DATA LOSS. PERFORM THIS OPERATION AT YOUR OWN RISK!

It is highly recommended that you restore your database using the verified snapshot backup set. If the override snapshot verification option is enabled, you should run DBCC CHECKDB after restore, and before the user can use the database.

Figure 28. DBCC Override Option

8.1. MSCS and Checking Database Consistency

MSCS is described in references 1 and 2. It is important to understand that a virtual SQL Server can not be used by SMSQL to verify a backed up database. A virtual server is only able to access databases that are located in storage, that are a resource for MSCS, and that are in SQL Server's dependency list, which a mounted writable Snapshot backup is not[26]. Use a remote SQL Server instance or install another local SQL Server instance on each node to run verification.

9. Backup Databases

The methodology used to back up databases has indirectly been decided when the databases' physical layouts were designed and implemented. Full backup[27] of user-defined databases through SMSQL is implemented using volume-level Snapshot copies, which means that all databases utilizing virtual disks located in the same volume will be backed up together. Databases sharing a virtual disk will be backed up together with all other databases sharing the same volume, except if the virtual disk contains a system database. In this case, full backup is streaming-based and is done per database, independent of other databases using the virtual disk or volume. This is not recommended since streaming-based full backup is slower than Snapshot technology-based full backup and streaming based backup take much more space in the related SnapInfo directory.

Backup can be done with the help of the backup wizard or the SnapManager Backup tab. The crucial screen is the same with both tools and is shown in figure 29. This screen has a couple of interesting points:

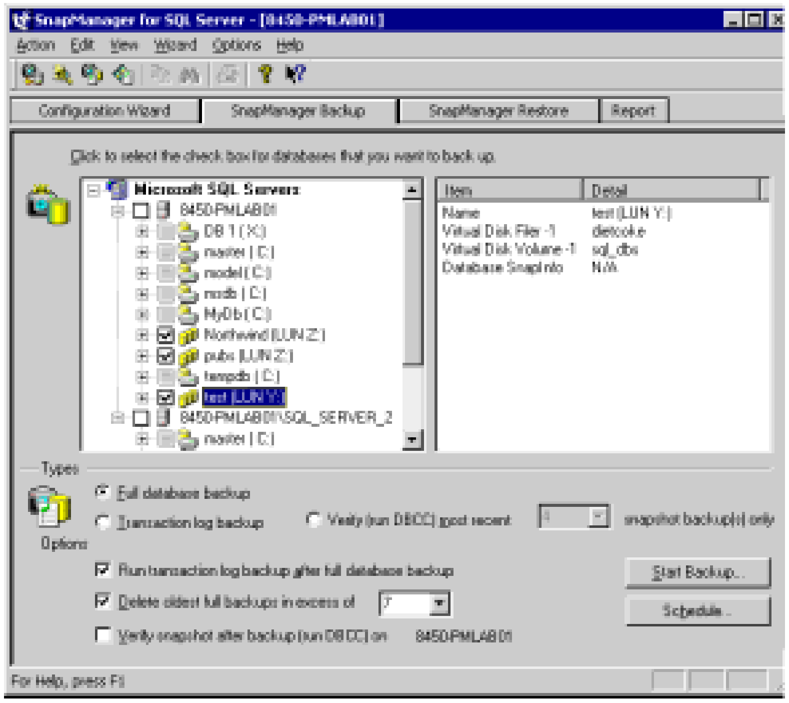
- Only databases located on virtual disks can be backed up by SMSQL.
- Databases using virtual disks located in the same filer volume can only be backed up together (pubs, Northwind, and test in figure 29). If you check or uncheck one of the databases, the action affects all databases utilizing the same volume.
- A transaction log is logically truncated when it is backed up and not when a full backup is created.
- Full backup can be followed by backup of the transaction log.
- Full backups use Snapshot and are quick.

- Transaction log backup is streaming-based and is done by SQL Server.
- Backup of system databases is streaming-based; see figure 30.
- Streaming-based backup is used when a used database is sharing a virtual disk with a system database.
- Backups can automatically be scheduled (set up through the Schedule button). The SQL Server Agent has to be running if it is scheduling backup jobs[28].

All backup sets take space in their SnapInfo directory for metadata, backed up transaction logs, and streaming-based backed up databases:

- Metadata per backup set is only a few kB.
- The size of a backed up transaction log depends mostly on number of new transactions since the last transaction log backup; see section 3.
- The size of a streaming-based full backup is roughly the same size as the online database.

More information on how to size is described in section 3.



Click to select the check box for databases that you want to back up.

Microsoft SQL Servers
- [ ] 8450-PMLAB01
  - DB 1 (X:)
  - master (C:)
  - model (C:)
  - msdb (C:)
  - MyDb (C:)
  - [✓] Northwind (LUN Z:)
  - [✓] pubs (LUN Z:)
  - tempdb (C:)
  - [✓] test (LUN Y:)
- [ ] 8450-PMLAB01\SQL_SERVER_2
  - master (C:)

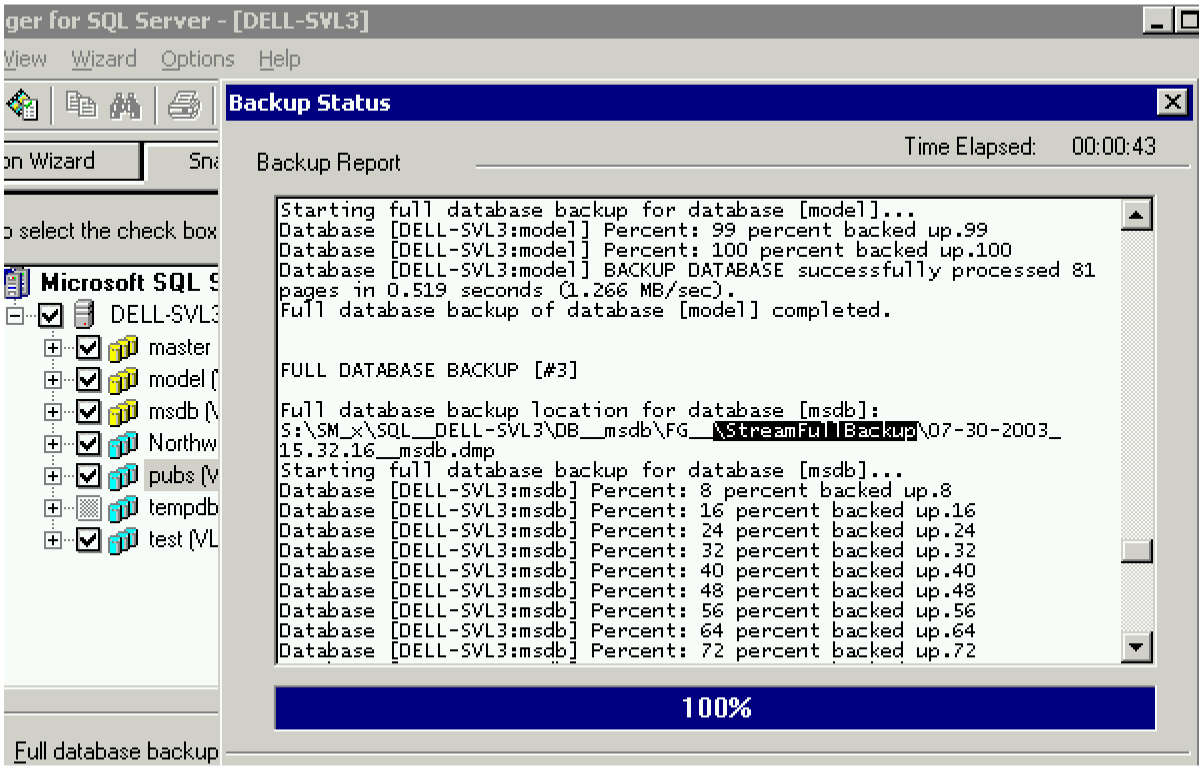
Item	Detail
Name	test (LUN Y:)
Virtual Disk Filer -1	dietcoke
Virtual Disk Volume -1	sql_dbs
Database SnapInfo	N/A

[✓] Full database backup
( ) Transaction log backup
( ) Verify (run DBCC) most recent [1] snapshot backup(s) only

[✓] Run transaction log backup after full database backup
[✓] Delete oldest full backups in excess of [7]
[ ] Verify snapshot after backup (run DBCC) on 8450-PMLAB01

Figure 29. Main Backup Screen

Correct backups are critical for recover a corrupt database, therefore, it is strongly recommended to daily check new backup reports daily for possible errors. An example of a report is shown in figure 30.

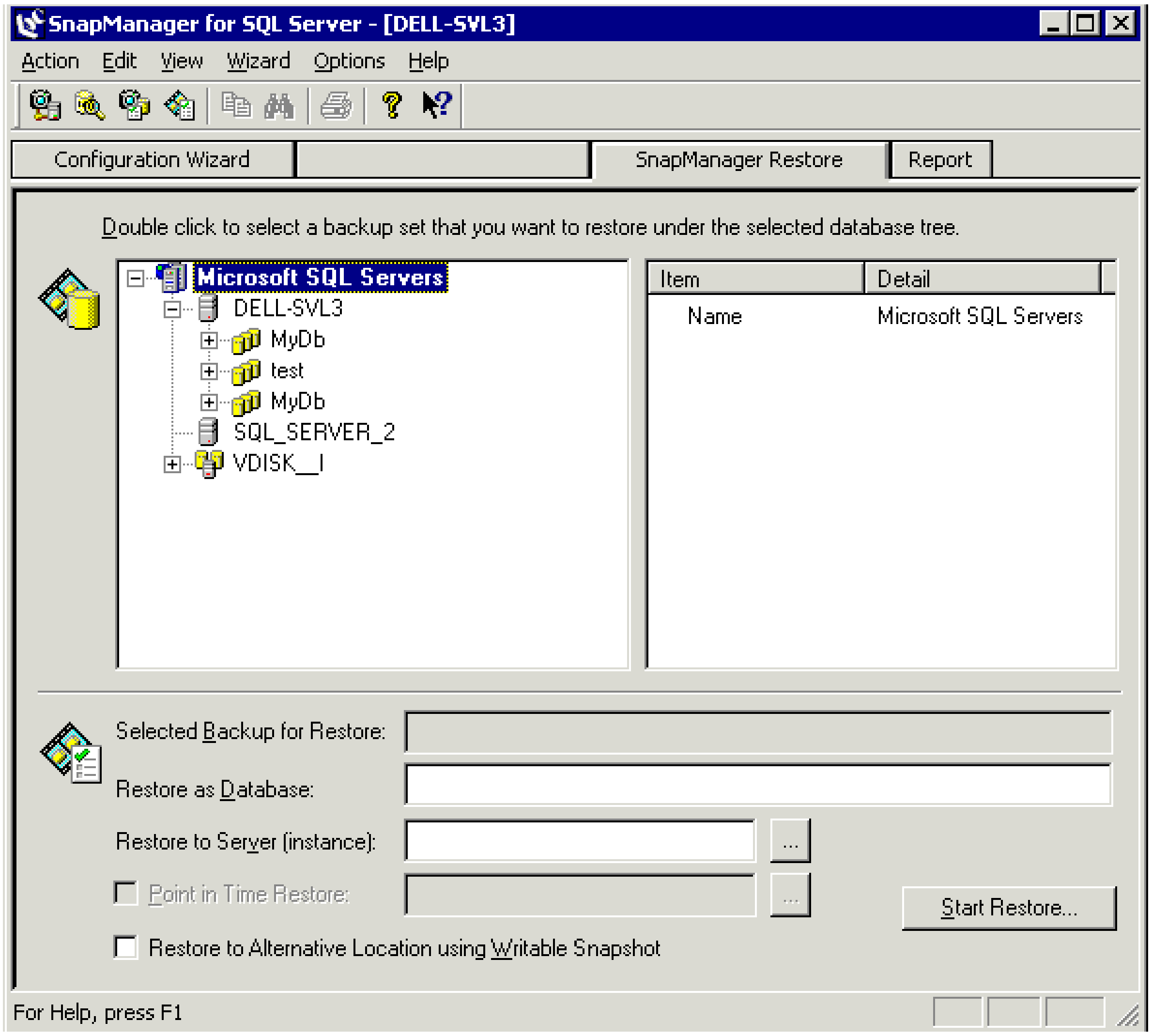


```
Starting full database backup for database [model]...
Database [DELL-SVL3:model] Percent: 99 percent backed up.99
Database [DELL-SVL3:model] Percent: 100 percent backed up.100
Database [DELL-SVL3:model] BACKUP DATABASE successfully processed 81
pages in 0.519 seconds (1.266 MB/sec).
Full database backup of database [model] completed.


FULL DATABASE BACKUP [#3]

Full database backup location for database [msdb]:
S:\SM_x\SQL__DELL-SVL3\DB__msdb\FG__\StreamFullBackup\07-30-2003_
15.32.16__msdb.dmp
Starting full database backup for database [msdb]...
Database [DELL-SVL3:msdb] Percent: 8 percent backed up.8
Database [DELL-SVL3:msdb] Percent: 16 percent backed up.16
Database [DELL-SVL3:msdb] Percent: 24 percent backed up.24
Database [DELL-SVL3:msdb] Percent: 32 percent backed up.32
Database [DELL-SVL3:msdb] Percent: 40 percent backed up.40
Database [DELL-SVL3:msdb] Percent: 48 percent backed up.48
Database [DELL-SVL3:msdb] Percent: 56 percent backed up.56
Database [DELL-SVL3:msdb] Percent: 64 percent backed up.64
Database [DELL-SVL3:msdb] Percent: 72 percent backed up.72
```

Figure 30. Report of System Database Backup

10. Restore Databases

It is critical to be able to restore a corrupt database quickly and efficiently when needed. It is never possible to plan when a disaster will occur, and it is therefore strongly recommended to test the implemented backup and restore processes occasionally. The process has indirectly been decided when the database's physical layout was implemented. Databases that are not sharing any virtual disks with another database can be restored without any effect on any other database. Section 2 discussed strategies for grouping multiple databases in a single virtual disk. This section will discuss strategies for restoring a single database stored in a virtual disk and how to restore all databases stored in a virtual disk. The SnapManager Restore Wizard or SnapManager Restore tab can be used for restoring databases. Both tools use the same central screen, as shown in figure 31.


SnapManager for SQL Server - [DELL-SVL3]

Action Edit View Wizard Options Help

Configuration Wizard | | SnapManager Restore | Report

Double click to select a backup set that you want to restore under the selected database tree.

```
Microsoft SQL Servers
  DELL-SVL3
    MyDb
    test
    MyDb
  SQL_SERVER_2
  VDISK__I
```

Item	Detail
Name	Microsoft SQL Servers

Selected Backup for Restore:

Restore as Database:

Restore to Server (instance):

Point in Time Restore:

Restore to Alternative Location using Writable Snapshot

Start Restore...

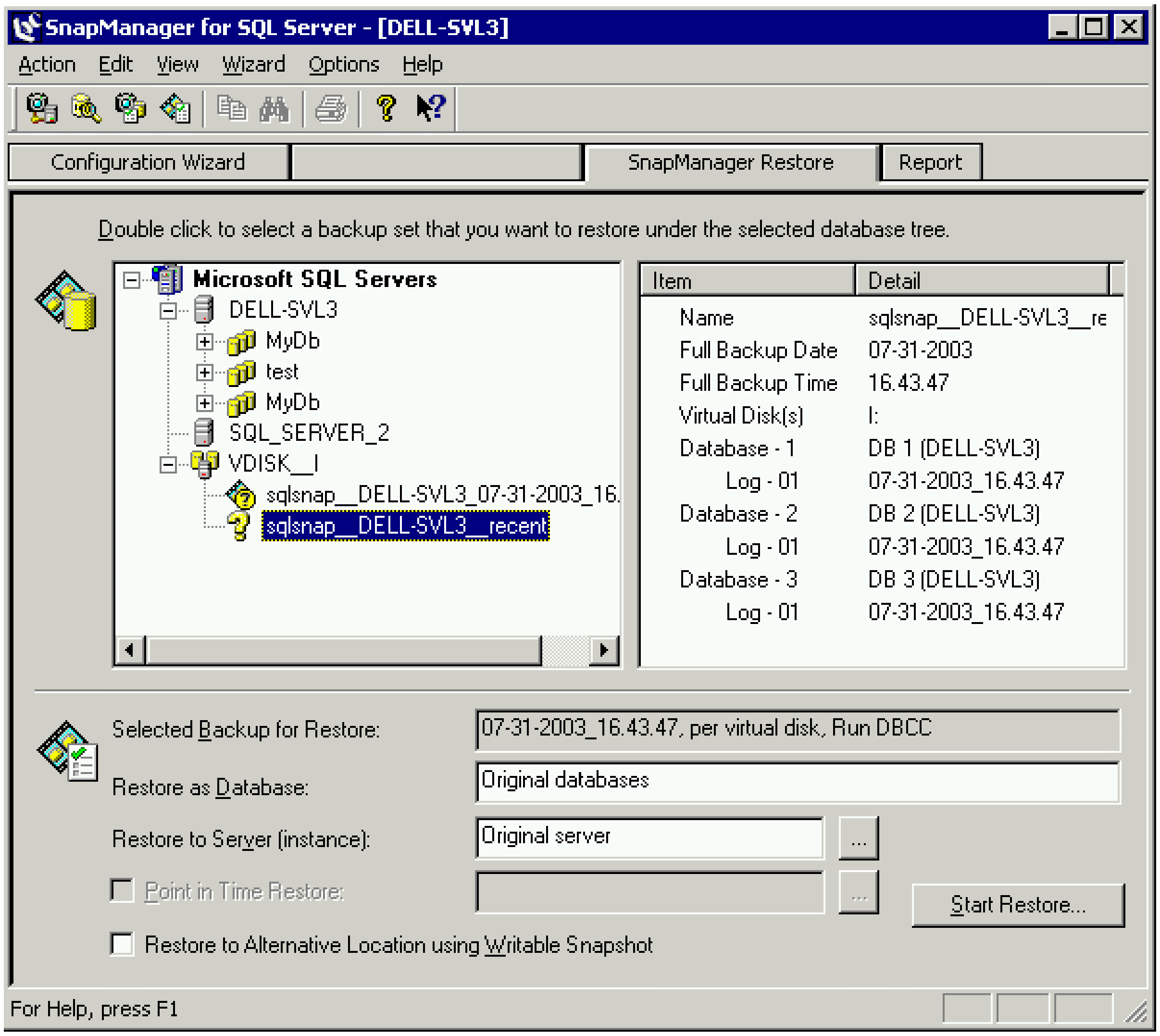
For Help, press F1

Figure 31. Main Restore Screen

The structure of SnapInfo (figure 20) is recognizable in figure 31. The top level is either a SQL Server instance or a virtual disk:

- SQL Server instance: DEL-SVL3 with a number of backed up databases
- SQL Server instance: SQL Server_2 with no backed up databases
- A virtual disk: VDISK_I

Expanding VDISK_I (figure 32) will show an overview of databases that can be restored. Restoring all databases in a virtual disk is straightforward and is probably quicker in many cases than restoring a single database.


SnapManager for SQL Server - [DELL-SVL3]
Action Edit View Wizard Options Help
Configuration Wizard
SnapManager Restore
Report

Double click to select a backup set that you want to restore under the selected database tree.

- Microsoft SQL Servers
  - DELL-SVL3
    - MyDb
    - test
    - MyDb
  - SQL_SERVER_2
  - VDISK__I
    - sqlsnap__DELL-SVL3_07-31-2003_16.
    - sqlsnap__DELL-SVL3__recent

Item	Detail
Name	sqlsnap__DELL-SVL3__re
Full Backup Date	07-31-2003
Full Backup Time	16.43.47
Virtual Disk(s)	I:
Database - 1	DB 1 (DELL-SVL3)
Log - 01	07-31-2003_16.43.47
Database - 2	DB 2 (DELL-SVL3)
Log - 01	07-31-2003_16.43.47
Database - 3	DB 3 (DELL-SVL3)
Log - 01	07-31-2003_16.43.47

Selected Backup for Restore: 07-31-2003_16.43.47, per virtual disk, Run DBCC

Restore as Database: Original databases

Restore to Server (instance): Original server

☐ Point in Time Restore:

Start Restore...

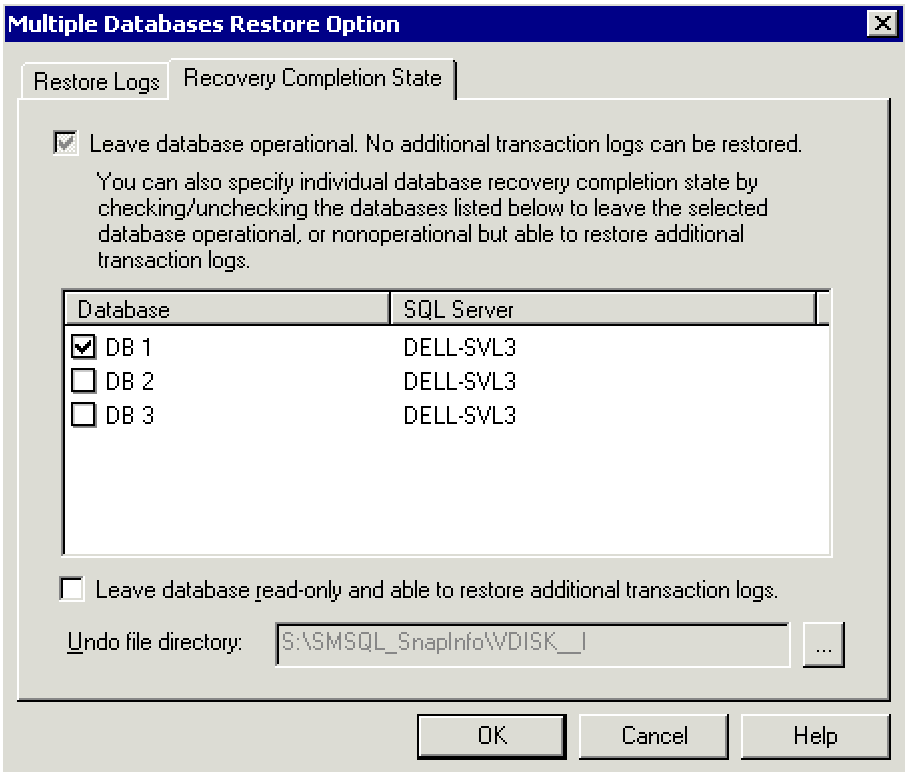
☐ Restore to Alternative Location using Writable Snapshot

For Help, press F1

Figure 32. Main Restore Screen

Restoring a single database in a virtual disk requires some manual steps. The process begins with the same screens as shown in figure 32:

1. Select the virtual disk with the database to be restored.
2. Select the option Restore to Alternative Location using Writable Snapshot.
3. Click StartRestore.
4. The next screen makes it possible to select only the database to restore (figure 33).


Multiple Databases Restore Option

☑ Leave database operational. No additional transaction logs can be restored.

You can also specify individual database recovery completion state by checking/unchecking the databases listed below to leave the selected database operational, or nonoperational but able to restore additional transaction logs.

Database	SQL Server
☑ DB 1	DELL-SVL3
☐ DB 2	DELL-SVL3
☐ DB 3	DELL-SVL3

☐ Leave database read-only and able to restore additional transaction logs.

Undo file directory: S:\SMSQL_SnapInfo\VDISK__I

Figure 33. Restoring Multiple Databases

5. Click OK.
6. Start restore.
7. Wait for Restore Complete message.

The result of the restore is:

- The Snapshot has been restored and mounted on an unused drive letter.
- The selected database is online but renamed DB 1__Mount
- The deselected databases are in loading state.

Another four steps have to be executed before the restored database is ready for use:

1. Start Enterprise Manager and browse to the database(s). Delete databases that are not to be restored; DB 2__Mount and DB 3 __Mount (figure 34).
    - o Close Enterprise Manager

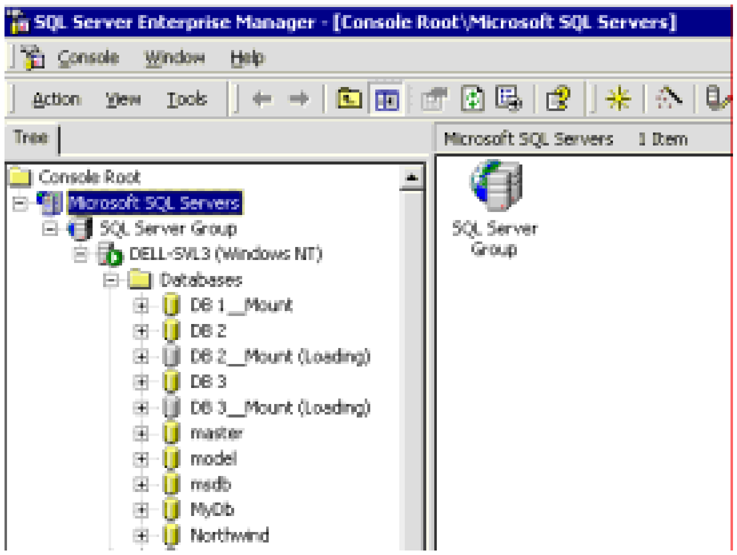
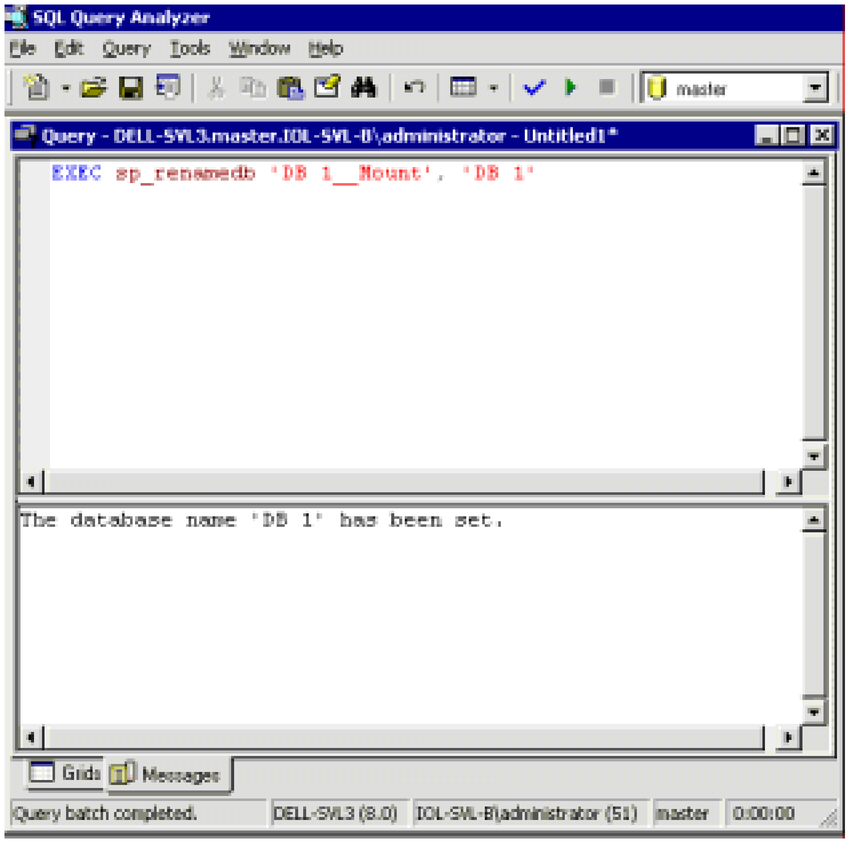


Figure 34. Enterprise Manager

2. Start Query Analyzer and rename the database; see figure 35.
   - o Close Query Analyzer


SQL Query Analyzer
File Edit Query Tools Window Help
master
Query - DELL-SVL3.master.IOL-SVL-B\administrator - Untitled1*
Grids Messages
Query batch completed.
DELL-SVL3 (8.0)
IOL-SVL-B\administrator (51)
master
0:00:00

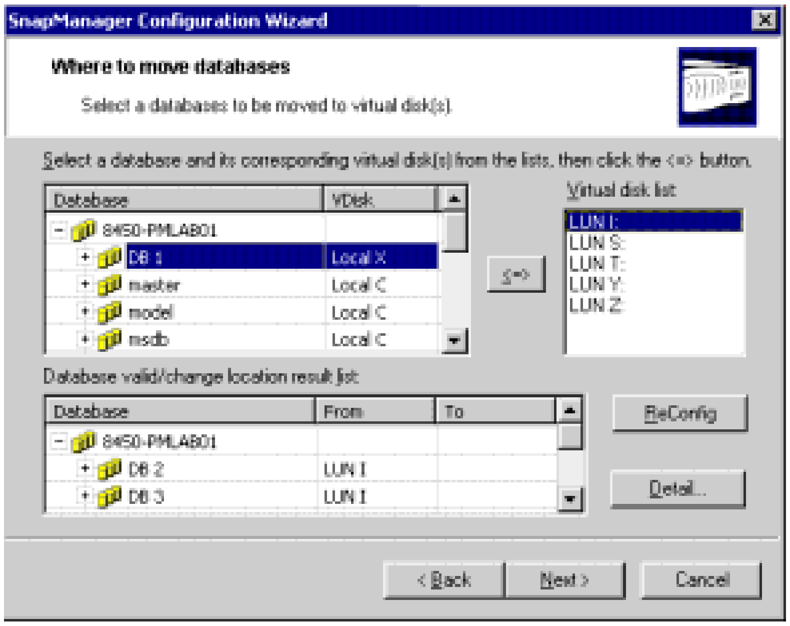
```
EXEC sp_renamedb 'DB 1__Mount', 'DB 1'
```

```
The database name 'DB 1' has been set.
```

Figure 35. Rename Database

3. Migrate the restored database back to the original location with SMSQL; see figure 36.

1.



Where to move databases

Select a databases to be moved to virtual disk(s).

Select a database and its corresponding virtual disk(s) from the lists, then click the <=> button.

Database	VDisk
− 8450-PMLAB01	
+ DB 1	Local X
+ master	Local C
+ model	Local C
+ msdb	Local C

Virtual disk list:

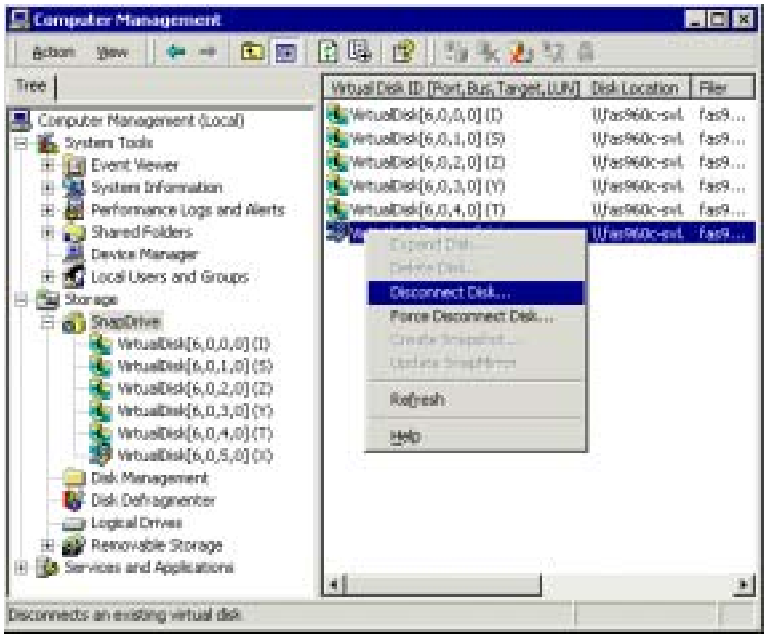
- LUN I:
- LUN S:
- LUN T:
- LUN Y:
- LUN Z:

Database valid/change location result list:

Database	From	To
− 8450-PMLAB01		
+ DB 2	LUN I	
+ DB 3	LUN I	

Figure 36. Migrate Restored Database

4. Open Computer Manager and disconnect the writable Snapshot copy; see figure 37. The writable Snapshot copy was mounted on X in this case.



Computer Management (Local)
- System Tools
  - Event Viewer
  - System Information
  - Performance Logs and Alerts
  - Shared Folders
  - Device Manager
  - Local Users and Groups
- Storage
  - SnapDrive
    - VirtualDisk[6,0,0,0] (I)
    - VirtualDisk[6,0,1,0] (S)
    - VirtualDisk[6,0,2,0] (Z)
    - VirtualDisk[6,0,3,0] (Y)
    - VirtualDisk[6,0,4,0] (T)
    - VirtualDisk[6,0,5,0] (X)
  - Disk Management
  - Disk Defragmenter
  - Logical Drives
  - Removable Storage
- Services and Applications

Virtual Disk ID [Port,Bus,Target,LUN]	Disk Location	Filer
VirtualDisk[6,0,0,0] (I)	\\fas960c-svl	fas9...
VirtualDisk[6,0,1,0] (S)	\\fas960c-svl	fas9...
VirtualDisk[6,0,2,0] (Z)	\\fas960c-svl	fas9...
VirtualDisk[6,0,3,0] (Y)	\\fas960c-svl	fas9...
VirtualDisk[6,0,4,0] (T)	\\fas960c-svl	fas9...
[illegible]	\\fas960c-svl	fas9...

Disconnects an existing virtual disk

Figure 37. Disconnect Writable Snapshot Backup

11. Checklist

This technical report has discussed how to lay out database files on NetApp virtual disks, and it has illustrated how to use SMSQL. Many recommendations have been discussed and illustrated throughout this report. This section will summarize common questions and recommendations from throughout this technical report. The list consists of two sections; the first section is a list of items to evaluate with recommended data to collect before migrating or implementing a SQL Server environment. The second section is a template of different database environments with generic recommended physical database layouts.

1. Number of SQL Server Instances per server? See section 6.2.
2. Number of databases per SQL Server instance? See section 6.2.
    a. If there are more databases than available drive letters, see sections 2.5, 4, and 6.2.
    b. How to group databases when storing in a single virtual disk; see section 2.5.
    c. One large database; see sections 2.3, 2.4, 4, and 6.2.
    d. Many small databases; see sections 2.5.
3. Space requirements (section 3):
    a. Initial size per database?
    b. Growth rate per database?
    c. Change rate per database?
    d. Physical I/O rate per database?
    e. Space for transaction log file?
    f. Performance requirements?
    g. Space for system databases? See section 3.1
    h. SnapInfo space requirements? See section 3.2
4. Number of Windows servers connected to one NetApp storage device? A relationship of one volume per server is recommended; see section 4.
5. MSCS; see section 2.5 and 8.1.
    a. Number of drive letters per MSCS node; see section 2.5.
    b. Local SQL Server instance or a remote server has to be used for verifying a database in an MSCS environment; see section 8.1.
6. SnapMirror; see sections 5 and 6.1.
    a. How many SnapInfo directories?
    b. We recommend that you locate all database files in one single volume!
    c. Use rolling Snapshot backups to update the mirror more frequently than when the database is backed up?
7. Back up (see section 9).
    a. Is full backup done with Snapshot or by using streaming backup?

    b. Drive letter has to be available for verifying a backed up database; see section 2.5.

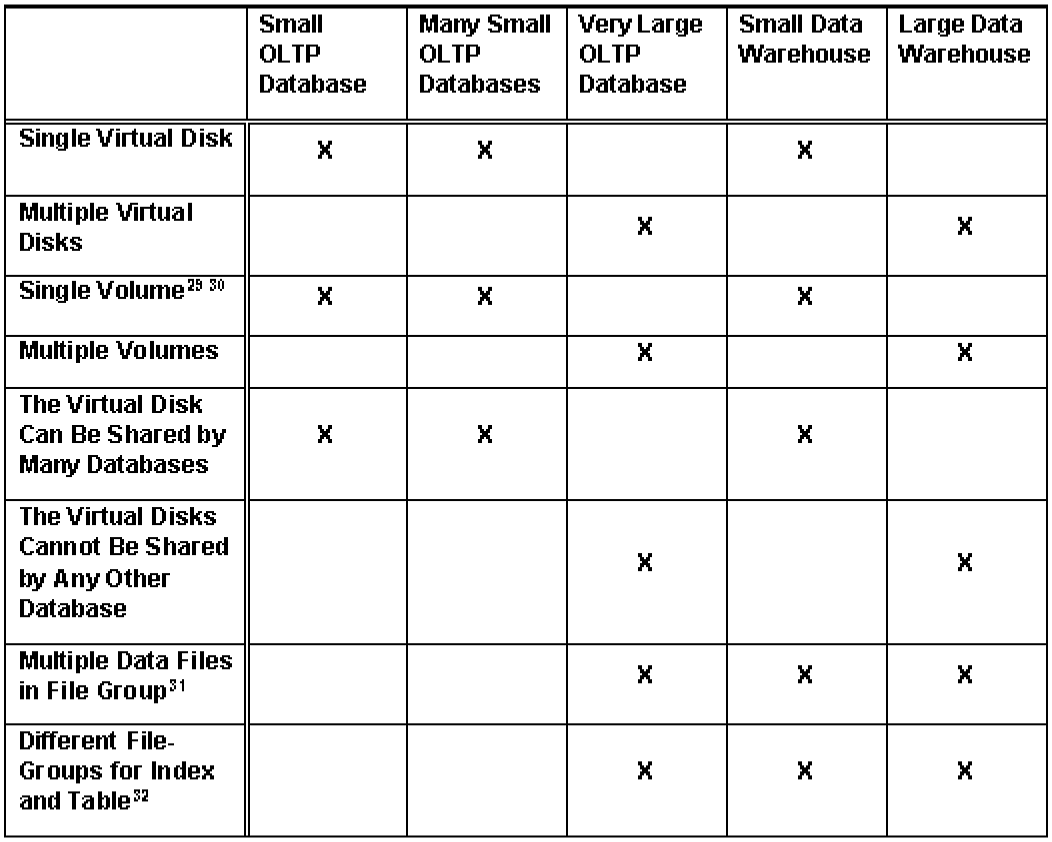
    c. Truncating transaction log; section 9.

8. Restore; see section 10.

    a. Many databases in one virtual disk?

    b. One single database out of several in one virtual disk.

    c. Bypass database verification? See section 8 and figure 28.



	Small OLTP Database	Many Small OLTP Databases	Very Large OLTP Database	Small Data Warehouse	Large Data Warehouse
Single Virtual Disk	X	X		X	
Multiple Virtual Disks			X		X
Single Volume[29, 30]	X	X		X	
Multiple Volumes			X		X
The Virtual Disk Can Be Shared by Many Databases	X	X		X	
The Virtual Disks Cannot Be Shared by Any Other Database			X		X
Multiple Data Files in File Group[31]			X	X	X
Different File-Groups for Index and Table[32]			X	X	X

Figure 38. Database Environments and Recommendations

12. References

1. Network Appliance - Guide to Integrating SQL Server with SnapDrive V 2.0.1 and NetApp Filers

2. SnapManager 1.0 for Microsoft SQL Server: Installation and System Administration Guide. Downloadable from the NOW™ site.

3. SMSQL's Online Help

4. Microsoft SQL Server 2000 Performance Tuning; Technical Reference - ISBN 0-7356-1270-6

5. Microsoft SQL Server 2000 Optimization Guide - ISBN 0-13-088358-1

13. Footnotes

1 There are several databases installed during the installation of SQL Server. 1) The master database is the main database. 2) The model database is a template for creating new databases. 3) The tempdb is for temporary storage and is recreated every time SQL Server is started; there is no need to back up. 4) msdb contains information about backup, alerts, and jobs scheduled by the SQL Server Agent. 5) The distribution database is only visible if replication is active.

2 Northwind and pubs demonstration databases are, by default, installed with SQL Server.

3 Virtual disks are created by SnapDrive and mapped to drive letters. Windows® has a fixed number of drive letters; about 20 are available for SnapDrive to create virtual disks on.

4 When a database's files utilize several virtual disks, then those Vdisks cannot be shared by any other database's files.

5 The potential I/O is higher but the actual I/O rate depends on the application's access rate and pattern.

6 Round-robin allocation is relative to file sizes.

7 A SQL Server page is 8kb, and a table extend is eight pages.

8 A database can have multiple transaction log files but all log files are viewed by SQL Server as one continuos logical file, and only one physical file will be used at the time. When all free space in one transaction log has been used, then the next file will be used. This will continue until the transaction log has been truncated or all available space in all files has been used. At this point, SQL Server will try to enlarge a transaction log file. If this is not possible, SQL Server will stop processing transactions to the database.

9 When the database engine realizes that it is sequentially scanning a table that is located in a file group with several data files, it will use that knowledge to request concurrent reads from each file in the file group.

10 Physical I/O rate depends greatly on the number of available disk drives that the I/O can be spread over.

11 How well an application can drive I/O depends on access pattern and locality. Random access creates longer latency and lower throughput, relative to sequential access.

12 A table is allocated in a file group.

13 When MSCS is used each drive letter has to be available for both nodes, which means that one stand-alone server and two MSCS nodes have the same number of available drive letters.

14 Single File SnapRestore®.

15 Snapshot copies are controlled by SnapDrive and are volume based.

16 Remember that when SMSQL verifies a backed up database it connects the Snapshot copy and a drive letter has to be available.

17 The network infrastructure is usually a shared resource.

18 SnapInfo is described in reference 2 and section 7.4.

19 SMSQL restores databases by restoring virtual disks, so if a virtual disk contains several databases, then all databases have to be restored or a single database has to be restored as described in reference 3 and section 10.

20 The setting checked by accessing: tool->Backup Setting->Transaction Log Backup where the check button is located.

21 A backup set is a full backup or a transaction log backup including metadata describing the backup set.

22 SnapInfo is described in reference 2 and section 7.4.

23 SnapInfo is a generic name for a path to a directory structure containing metadata for backup set (backup set is defined in reference 2).

24 See volume sizing in section 3.

25 The procedure shown will also work when migrating a database to other virtual disks.

26 The virtual disk in Snapshot will be mounted by SMSQL as a local drive and will not be available for MSCS.

27 All recovery models (simple, full, or bulk-logged) support full backup, but Simple doesn't support transaction log backup.

28 When SQL Server is stopped, SQL Server Agent will also stop, but the agent will not be started when SQL Server is started. When SQL Server Agent is started, SQL Server will also be started.

29 SnapMirror is simplest when a database utilizes only one volume.

30 Sequential access and random access are mixed in the same volume, but that is only a potential issue with environments that require high level of I/O access.

31 Potentially higher concurrent I/O access when a table is sequentially scanned.

32 Potentially higher concurrent I/O to both data and index files.

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com