

Microsoft SQL Server on EMC Symmetrix Storage Systems Solutions Guide

Version 1.0

- Generating Restartable Microsoft SQL Server Database Copies
- Microsoft SQL Server Remote Replication and Disaster Restart
- Microsoft SQL Server Database Layout and Performance

Copyright © 2006 EMC® Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

All other trademarks used herein are the property of their respective owners.

Microsoft SQL Server on EMC Symmetrix Storage Systems

Version 1.0

Solutions Guide

P/N 300-003-722

H2203



	Preface	xv
Chapter 1	Microsoft SQL Server	1-1
	1.1 Microsoft SQL Server overview	1-2
	1.2 Microsoft SQL Server instances and databases	1-2
	1.3 Microsoft SQL Server logical components	1-2
	1.3.1 A SQL Server database	1-3
	1.4 Data access	1-4
	1.5 Microsoft SQL Server physical components	1-5
	1.5.1 Data files	1-5
	1.5.2 Transaction log	1-7
	1.6 System databases	1-8
	1.7 Microsoft SQL Server instances	1-8
	1.8 Microsoft Cluster Service installations	1-11
	1.9 Backup and recovery interfaces – Virtual Device Interface	1-12
	1.10 Microsoft SQL Server and EMC integration	1-12
	1.11 Additional Microsoft SQL Server tools	1-13
Chapter 2	EMC Foundation Products	2-1
	2.1 EMC Symmetrix DMX	2-4
	2.2 EMC Solutions Enabler base management	2-4
	2.3 Change Tracker	2-6
	2.4 EMC Symmetrix Remote Data Facility	2-7
	2.4.1 SRDF benefits	2-8
	2.4.2 SRDF modes of operation	2-8
	2.4.3 SRDF device and composite groups	2-9
	2.4.4 SRDF consistency groups	2-9
	2.4.5 SRDF terminology	2-12
	2.4.6 SRDF control operations	2-14
	2.4.7 Failover and failback operations	2-17
	2.4.8 SRDF/A (Asynchronous) operations	2-19
	2.4.9 EMC SRDF/Cluster Enabler solutions	2-20

2.5	EMC TimeFinder	2-21
2.5.1	TimeFinder/Mirror Establish operations	2-22
2.5.2	TimeFinder split operations	2-23
2.5.3	TimeFinder restore operations	2-24
2.5.4	TimeFinder consistent split	2-24
2.5.5	TimeFinder reverse split	2-27
2.5.6	TimeFinder/Clone operations	2-27
2.5.7	TimeFinder/Snap operations	2-29
2.6	EMC Storage Resource Management	2-31
2.7	EMC ControlCenter	2-35
2.8	EMC PowerPath	2-37
2.9	EMC Replication Manager	2-38
Chapter 3	Creating Microsoft SQL Server Database Clones	3-1
3.1	Overview	3-2
3.2	Recoverable versus restartable copies of databases	3-3
3.2.1	Recoverable database copies	3-3
3.2.2	Restartable database copies	3-3
3.3	Copying the database with Microsoft SQL Server shutdown	3-3
3.3.1	Using TimeFinder/Mirror	3-4
3.3.2	Using TimeFinder/Clone	3-6
3.3.3	Using TimeFinder/Snap	3-7
3.4	Copying the database using EMC Consistency technology	3-9
3.4.1	Using TimeFinder/Mirror	3-9
3.4.2	Using TimeFinder/Clone	3-11
3.4.3	Using TimeFinder/Snap	3-12
3.5	Copying the database using SQL Server Virtual Device Interface	3-14
3.5.1	Using TimeFinder/Mirror	3-14
3.5.2	Using TimeFinder/Clone	3-18
3.5.3	Using TimeFinder/Snap	3-22
3.6	Copying the database using Replication Manager	3-26
3.7	Transitioning disk copies to Microsoft SQL Server databases clones	3-27
3.7.1	Instantiating clones from consistent split or shutdown images	3-28
3.7.2	Using SQL Server VDI to process the database image	3-29
3.8	Reinitializing the cloned environment	3-35
3.9	Choosing a database cloning methodology	3-35
Chapter 4	Backing up Microsoft SQL Server Databases	4-1
4.1	EMC Consistency technology and backup	4-2
4.2	SQL Server backup functionality	4-3
4.2.1	Microsoft SQL Server recovery models	4-4
4.2.2	Types of SQL Server backups	4-7
4.3	SQL Server log markers	4-8
4.4	EMC products for SQL Server backup	4-9
4.4.1	Integrating TimeFinder and Microsoft SQL Server	4-9

	4.4.2	EMC Storage Resource Management	4-11
	4.5	TF/SIM VDI backup	4-13
	4.5.1	Using TimeFinder/Mirror	4-14
	4.5.2	Using TimeFinder/Clone	4-16
	4.5.3	Using TimeFinder/Snap	4-18
	4.6	SYMIOCTL VDI backup	4-20
	4.6.1	Using TimeFinder/Mirror	4-20
	4.7	Replication Manager VDI backup	4-22
	4.8	Saving the VDI backup to long-term media	4-23
Chapter 5		Restoring and Recovering Microsoft SQL Server Databases	5-1
	5.1	SQL Server restore functionality	5-2
	5.1.1	SQL Server – RESTORE WITH RECOVERY	5-4
	5.1.2	SQL Server – RESTORE WITH NORECOVERY	5-4
	5.1.3	SQL Server – RESTORE WITH STANDBY	5-5
	5.2	EMC products for SQL Server recovery	5-6
	5.3	EMC Consistency technology and restore	5-7
	5.4	TF/SIM VDI restore	5-9
	5.4.1	Using TimeFinder/Mirror	5-10
	5.4.2	Using TimeFinder/Clone	5-18
	5.4.3	Using TimeFinder/Snap	5-18
	5.5	SYMIOCTL VDI restore	5-28
	5.5.1	Executing TimeFinder restore	5-29
	5.5.2	SYMIOCTL with NORECOVERY	5-31
	5.5.3	SYMIOCTL with STANDBY	5-33
	5.5.4	SYMIOCTL with RECOVERY	5-36
	5.6	Replication Manager VDI restore	5-37
	5.7	Applying logs up to timestamps or marked transactions	5-38
Chapter 6		Understanding Microsoft SQL Server Disaster Restart and Disaster Recovery	6-1
	6.1	Definitions	6-3
	6.1.1	Dependent-write consistency	6-3
	6.1.2	Database restart	6-3
	6.1.3	Database recovery	6-3
	6.1.4	Roll forward recovery	6-4
	6.2	Design considerations for disaster restart and disaster recovery	6-4
	6.2.1	Recovery Point Objective (RPO)	6-4
	6.2.2	Recovery Time Objective (RTO)	6-5
	6.2.3	Operational complexity	6-5
	6.2.4	Source server activity	6-6
	6.2.5	Production impact	6-6
	6.2.6	Target server activity	6-6
	6.2.7	Number of copies	6-6
	6.2.8	Distance for solution	6-6
	6.2.9	Bandwidth requirements	6-7

6.2.10	Federated consistency.....	6-7
6.2.11	Testing the solution	6-7
6.2.12	Cost.....	6-7
6.3	Tape-based solutions.....	6-8
6.3.1	Tape-based disaster recovery.....	6-8
6.3.2	Tape-based disaster restart.....	6-8
6.4	Local high-availability solutions.....	6-9
6.5	Multisite high-availability solutions.....	6-9
6.6	Remote replication challenges.....	6-10
6.6.1	Propagation delay	6-11
6.6.2	Bandwidth requirements.....	6-11
6.6.3	Network infrastructure.....	6-11
6.6.4	Method of instantiation.....	6-12
6.6.5	Method of re-instantiation	6-12
6.6.6	Change rate at the source site	6-12
6.6.7	Locality of reference.....	6-12
6.6.8	Expected data loss	6-13
6.6.9	Failback operations.....	6-13
6.7	Array-based remote replication.....	6-13
6.8	Planning for array-based replication	6-14
6.9	SQL Server specific issues	6-14
6.10	SRDF/S: Single Symmetrix to single Symmetrix	6-15
6.10.1	How to restart in the event of production site loss	6-17
6.11	SRDF/S and consistency groups	6-17
6.11.1	Rolling disaster.....	6-17
6.11.2	Protecting against a rolling disaster.....	6-19
6.11.3	SRDF/S with multiple source Symmetrix arrays	6-20
6.12	SRDF/A.....	6-22
6.12.1	SRDF/A using single source Symmetrix.....	6-23
6.12.2	SRDF/A using multiple source Symmetrix	6-24
6.12.3	Restart processing.....	6-25
6.13	SRDF/AR single hop.....	6-26
6.13.1	Restart processing.....	6-28
6.14	SRDF/AR multi-hop	6-28
6.14.1	Restart processing.....	6-30
6.15	Database log-shipping solutions.....	6-30
6.15.1	Overview of log shipping	6-30
6.15.2	Log shipping considerations.....	6-33
6.15.3	Log shipping and the remote database	6-35
6.15.4	Shipping logs with SRDF.....	6-39
6.15.5	SQL Server Database Mirroring.....	6-39
6.16	Running database solutions.....	6-43
6.16.1	SQL Server transactional replication.....	6-43
6.17	Other transactional systems.....	6-45

Chapter 7	Microsoft SQL Server Database Layouts on EMC Symmetrix.....	7-1
7.1	The performance stack.....	7-2
7.1.1	Optimizing I/O.....	7-3
7.1.2	Storage system layout considerations.....	7-4
7.2	SQL Server layout recommendations.....	7-5
7.2.1	File system partition alignment.....	7-5
7.2.2	General principles for layout.....	7-5
7.2.3	SQL Server layout – replication considerations.....	7-7
7.3	Symmetrix performance guidelines.....	7-7
7.3.1	Front-end connectivity.....	7-7
7.3.2	Symmetrix cache.....	7-9
7.3.3	Back-end considerations.....	7-16
7.3.4	Additional layout considerations.....	7-18
7.3.5	Configuration recommendations.....	7-18
7.4	RAID considerations.....	7-19
7.4.1	Types of RAID.....	7-19
7.4.2	RAID recommendations.....	7-22
7.4.3	Symmetrix metavolumes.....	7-23
7.5	Host versus array-based striping.....	7-24
7.5.1	Host-based striping.....	7-24
7.5.2	Symmetrix-based striping (metavolumes).....	7-25
7.5.3	Striping recommendation.....	7-25
7.6	Data placement considerations.....	7-26
7.6.1	Disk performance considerations.....	7-27
7.6.2	Hypervolume contention.....	7-29
7.6.3	Maximizing data spread across back-end devices.....	7-29
7.6.4	Minimizing disk head movement.....	7-30
7.7	Other layout considerations.....	7-31
7.7.1	Database layout considerations with SRDF/S.....	7-31
7.7.2	Database cloning, TimeFinder, and sharing spindles.....	7-31
7.7.3	Database clones using TimeFinder/Snap.....	7-32
7.8	Database-specific settings for SQL Server.....	7-32
7.8.1	Remote replication performance considerations.....	7-32
7.8.2	TEMPDB storage and replication.....	7-33
Appendix A	Related Documents.....	A-1
A.1	Related documents.....	A-2
Appendix B	References.....	B-1
B.1	Sample SYMCLI group creation commands.....	B-2

Figure 1-1 Microsoft SQL Server architecture overview.....	1-3
Figure 1-2 SQL Server database architecture.....	1-5
Figure 1-3 SQL Server internal data file logical structure	1-6
Figure 1-4 Multiple SQL Server instance installation directories.....	1-9
Figure 1-5 SQL Server Enterprise Manager with multiple instances	1-10
Figure 1-6 Connecting to default and named instances	1-11
Figure 1-7 SRDF/CE for MSCS resource group for a SQL Server instance	1-12
Figure 2-1 Basic synchronous SRDF configuration.....	2-8
Figure 2-2 SRDF consistency group	2-11
Figure 2-3 SRDF establish and restore control operations.....	2-15
Figure 2-4 SRDF failover and failback control operations	2-17
Figure 2-5 Geographically distributed four-node EMC SRDF/CE clusters.....	2-21
Figure 2-6 EMC Symmetrix configured with standard volumes and BCVs.....	2-22
Figure 2-7 ECA consistent split across multiple database associated hosts.....	2-25
Figure 2-8 ECA consistent split on a local Symmetrix	2-26
Figure 2-9 Creating a copy session using the symclone command.....	2-28
Figure 2-10 TimeFinder/Snap copy of a standard device to a VDEV	2-30
Figure 2-11 SRM commands	2-32
Figure 2-12 ControlCenter family overview	2-35
Figure 3-1 Copying a shutdown SQL Server database with TimeFinder/Mirror.....	3-5
Figure 3-2 Copying a shutdown SQL Server database with TimeFinder/Clone.....	3-6
Figure 3-3 Copying a shutdown SQL Server database with TimeFinder/Snap	3-8
Figure 3-4 Copying a running SQL Server database with TimeFinder/Mirror.....	3-10
Figure 3-5 Copying a running SQL Server database with TimeFinder/Clone.....	3-11
Figure 3-6 Copying a running SQL Server database with TF/Snap.....	3-13
Figure 3-7 Creating a TimeFinder/Mirror VDI backup of a SQL Server database.....	3-15
Figure 3-8 Sample TimeFinder/SIM backup execution with TimeFinder/Mirror	3-16
Figure 3-9 Sample SYMIOCTL execution with TimeFinder/Mirror.....	3-18
Figure 3-10 Creating a VDI backup of a SQL Server database with TimeFinder/Clone.....	3-19
Figure 3-11 Sample SYMIOCTL execution with TF/Clone.....	3-22

Figure 3-12 Creating a VDI backup of a SQL Server database with TimeFinder/Snap.....	3-23
Figure 3-13 Sample TF/SIM backup execution with TF/Snap.....	3-24
Figure 3-14 Sample SYMIOCTL execution with TF/SNAP.....	3-26
Figure 3-15 Using Replication Manager to make a TimeFinder Replica of a SQL Server database.....	3-27
Figure 3-16 Attaching a consistent split image to SQL Server	3-29
Figure 3-17 TF/SIM and SQL Server VDI to create a clone.....	3-30
Figure 3-18 SYMIOCTL and SQL Server VDI to create a clone	3-31
Figure 3-19 Using TF/SIM and SQL Server VDI to create a standby database.....	3-32
Figure 3-20 Using SYMIOCTL and SQL Server VDI to create a standby database ..	3-32
Figure 3-21 Attaching a cloned database with relocated data and log locations	3-33
Figure 3-22 SQL Query Analyzer executing sp_helpdb.....	3-34
Figure 3-23 Mapping database logical components to new file locations.....	3-34
Figure 3-24 Using TF/SIM and VDI to restore database to a new location	3-35
Figure 4-1 SQL Enterprise Manager backup interface	4-3
Figure 4-2 DATABASE BACKUP Transact-SQL execution	4-4
Figure 4-3 Recovery model options for a SQL Server database	4-6
Figure 4-4 Setting recovery model via Transact-SQL.....	4-7
Figure 4-5 SYMRDB listing SQL database file locations.....	4-12
Figure 4-6 SYMMIR query of a device group	4-13
Figure 4-7 Creating a TimeFinder/Mirror VDI backup of a SQL Server database	4-15
Figure 4-8 TF/SIM backup execution with TimeFinder/Mirror	4-16
Figure 4-9 Creating a VDI backup of a SQL Server database with TimeFinder/Clone	4-17
Figure 4-10 Creating a VDI backup of a SQL Server database with TimeFinder/Snap.....	4-18
Figure 4-11 TF/SIM backup execution with TimeFinder/Snap.....	4-19
Figure 4-12 Sample SYMIOCTL execution with TF/Mirror	4-22
Figure 4-13 Using RM to make a TimeFinder replica of a SQL Server database.....	4-23
Figure 5-1 SQL Enterprise Manager restore interface.....	5-3
Figure 5-2 Additional Enterprise Manager restore options	5-4
Figure 5-3 DATABASE RESTORE Transact-SQL execution	5-6
Figure 5-4 Attaching a Consistent split image to SQL Server.....	5-9
Figure 5-5 TF/SIM restore process using TimeFinder/Mirror.....	5-10
Figure 5-6 TF/SIM restore production database with TimeFinder/Mirror - NORECOVERY	5-12
Figure 5-7 SQL Enterprise Manager view of a LOADING database.....	5-13
Figure 5-8 Restore of incremental transaction log with NORECOVERY	5-14
Figure 5-9 TF/SIM restore production database with TimeFinder/Mirror - STANDBY	5-15
Figure 5-10 SQL Enterprise Manager view of a STANDBY (read-only) database	5-16
Figure 5-11 Restore of incremental transaction log with STANDBY	5-17
Figure 5-12 TF/SIM restore using TimeFinder/Snap	5-18
Figure 5-13 TimeFinder/Snap listing restore session	5-20

Figure 5-14 TF/SIM restore production database with TimeFinder/SNAP– NORECOVERY	5-21
Figure 5-15 SQL Enterprise Manager view of a LOADING database	5-22
Figure 5-16 Restore of incremental transaction log with NORECOVERY	5-23
Figure 5-17 TF/SIM restore production database with TimeFinder/SNAP - STANDBY	5-25
Figure 5-18 SQL Enterprise Manager view of a STANDBY (read-only) database.....	5-26
Figure 5-19 Restore of incremental transaction log with STANDBY	5-27
Figure 5-20 Using SYMIOCTL for TimeFinder/Mirror restore of production database	5-29
Figure 5-21 SYMIOCTL restore production database – NORECOVERY.....	5-31
Figure 5-22 SQL Enterprise Manager view of a LOADING database	5-32
Figure 5-23 Restore of incremental transaction log with NORECOVERY	5-33
Figure 5-24 SYMIOCTL restore production database – STANDBY	5-34
Figure 5-25 SQL Enterprise Manager view of a STANDBY (read-only) database.....	5-35
Figure 5-26 Restore of incremental transaction log with STANDBY	5-36
Figure 5-27 SYMIOCTL restore production database – recovery mode	5-37
Figure 5-28 Replication Manager/Local restore overview.....	5-38
Figure 6-1 SRDF/S replication process.....	6-15
Figure 6-2 Rolling disaster with multiple production Symmetrix arrays.....	6-18
Figure 6-3 SRDF consistency group protection against rolling disaster	6-19
Figure 6-4 SRDF/S with multiple source Symmetrix and ConGroup protection	6-21
Figure 6-5 SRDF/Asynchronous replication internals	6-22
Figure 6-6 SRDF/AR single-hop replication internals.....	6-27
Figure 6-7 SRDF/AR multi-hop replication internals.....	6-29
Figure 6-8 Log shipping configuration – Add Destination Database dialog.....	6-32
Figure 6-9 Log shipping implementation overview	6-36
Figure 6-10 Restore of incremental transaction log with NORECOVERY	6-37
Figure 6-11 Restore of incremental transaction log with STANDBY	6-38
Figure 6-12 SQL Server Database Mirroring with SRDF/S	6-40
Figure 6-13 SQL Server Database Mirroring flow overview – SYNC mode	6-41
Figure 7-1 The performance stack	7-3
Figure 7-2 Relationship between I/O size, operations per second, and throughput	7-9
Figure 7-3 Performance Manager graph of write pending for single hypervolume.....	7-14
Figure 7-4 Performance Manager graph of write pending for four member striped metavolume	7-15
Figure 7-5 Comparison of write workload single hyper and striped metavolume	7-16
Figure 7-6 RAID 5 (3+1) layout detail.....	7-20
Figure 7-7 Anatomy of a RAID 5 write	7-21
Figure 7-8 Disk performance factors	7-28

Table 1-1	Microsoft SQL Server system databases	1-8
Table 2-1	SYMCLI base commands	2-5
Table 2-2	TimeFinder device type summary	2-29
Table 2-3	Data object SRM commands	2-32
Table 2-4	Data object mapping commands.....	2-33
Table 2-5	File system SRM command	2-34
Table 2-6	File system SRM commands	2-34
Table 2-7	SRM statistics command	2-34
Table 3-1	A comparison of database cloning technologies.....	3-36
Table 3-2	Database cloning requirements and solutions.....	3-36
Table 7-1	SQL Server data and log response guidelines.....	7-5

This document describes how the EMC Symmetrix storage system operates and interfaces with Microsoft SQL Server. The information in this document is based on Microsoft SQL Server 2000 and Microsoft SQL Server 2005 on Symmetrix storage systems running Solutions Enabler V6.0.2, and current releases of Symmetrix Enginuity microcode.

This document provides an overview of Microsoft SQL Server 2000 along with a general description of EMC products and utilities that can be used for SQL Server administration. EMC Symmetrix storage systems and EMC software products can be used to manage Microsoft SQL Server environments and to enhance database and storage management backup/recovery and restart procedures. Using EMC products and utilities to manage Microsoft SQL Server environments can help reduce database and storage management administration, reduce system CPU resource consumption, and reduce the time required to clone, backup, recover, or restart Microsoft SQL Server databases.

This document is divided into seven chapters. Topics range from general Microsoft SQL Server and EMC product descriptions in chapters 1 and 2 to detailed discussions of implementation procedures in chapters 3 thru 7. Detailed examples are included to illustrate features and functions, and are only provided to clarify the topic being discussed. Not all possible combinations of solutions are covered in this document.

Examples provided in this document cover methods for performing various Microsoft SQL Server functions using Symmetrix systems with EMC software. These examples were developed for laboratory testing and may need tailoring to suit other operational environments. Any procedures outlined in this document should be thoroughly tested before production implementation.

In this document the product name “Microsoft SQL Server 2000” and “Microsoft SQL Server 2005” may be referred to as “SQL Server”. The acronym “SQL” refers to the Structured Query Language, and should not be confused with the product “Microsoft SQL Server”. The Structured Query Language (SQL) is used within many relational database management systems (RDBMS) to store, retrieve, and manipulate data. Microsoft provides a specific implementation of the SQL language called Transact SQL (T-SQL). T-SQL is used throughout this document in the various examples.

Microsoft provides extensive documentation on SQL Server via its website, and through

the SQL Server Books On-Line documentation set. This should be the primary source for information on SQL Server-specific commands and T-SQL syntax. The Books On Line documentation may be installed through the SQL Server installation process, and may be installed independently of the SQL Server database engine. Updated versions of the Books On Line documentation are available for free download from the Microsoft SQL Server website <http://www.microsoft.com/sql>.

Audience

The intended audience is SQL Server systems administrators, database administrators, and storage management personnel responsible for managing SQL Server systems.

This chapter presents these topics:

1.1	Microsoft SQL Server overview	1-2
1.2	Microsoft SQL Server instances and databases	1-2
1.3	Microsoft SQL Server logical components.....	1-2
1.4	Data access.....	1-4
1.5	Microsoft SQL Server physical components	1-5
1.6	System databases	1-8
1.7	Microsoft SQL Server instances	1-8
1.8	Microsoft Cluster Service installations.....	1-11
1.9	Backup and recovery interfaces – Virtual Device Interface	1-12
1.10	Microsoft SQL Server and EMC integration.....	1-12
1.11	Additional Microsoft SQL Server tools.....	1-13

1.1 Microsoft SQL Server overview

Microsoft SQL Server is Microsoft Corporation's premier relational database management system (RDBMS). Developed over several years, Microsoft SQL Server has grown to become one of the most scalable and highly performing database systems currently available, as shown by several industry leading Transaction Processing Council (TPC) benchmarks.

The origin of the RDBMS stems back to initial co-development work between Microsoft and Sybase, which focused on developing a database management system for the then evolving OS/2 environment. Later, a variation of this initial release would become available on Microsoft's LAN Manager platform. In 1995, Microsoft developed and released Microsoft SQL Server 6.0 on the Windows platform. In the intervening years a number of subsequent releases providing increased functionality, performance, and scalability have been widely adopted.

As of November 2005, Microsoft SQL Server 2005 is the latest in a series of SQL Server product releases that specifically cater to the Microsoft Windows platform. Currently, Microsoft SQL Server does not support any platform other than Windows Server. With the introduction of the Itanium 64-bit version of the Windows Server platform in 2003, Microsoft released an Itanium 64-bit (IA-64) version of Microsoft SQL Server 2000.

As of Microsoft SQL Server 2005, support is available across multiple architectures including 32-bit (IA-32), Itanium 64-bit (IA-64), and x64 (Intel EM64T). Further information and support may be found at the appropriate Microsoft product and support website locations.

1.2 Microsoft SQL Server instances and databases

A Microsoft SQL Server instance is defined as a discrete set of files and executables, which operate autonomously to provide service. SQL Server supports multiple instances operating independently on a given operating system. Within any given SQL Server instance there are typically a number of independent user databases. It is important to understand this relationship of database-to-SQL Server instance, and the capability to support multiple SQL Server instances on the server. A single database cannot physically exist across multiple database instances, but can be logically represented across multiple instances and/or servers.

1.3 Microsoft SQL Server logical components

Microsoft SQL Server may be divided into two main logical components and a number of subsidiary subsystems:

- ◆ Relational engine – responsible for verifying SQL statements, and selecting the most efficient means of retrieving the data requested
- ◆ Storage engine – responsible for executing physical I/O requests, which return the rows requested by the relational engine

Together, these components create a complete relational database environment providing continuous availability and data integrity. Figure 1-1 provides an overview of the Microsoft SQL Server architecture.

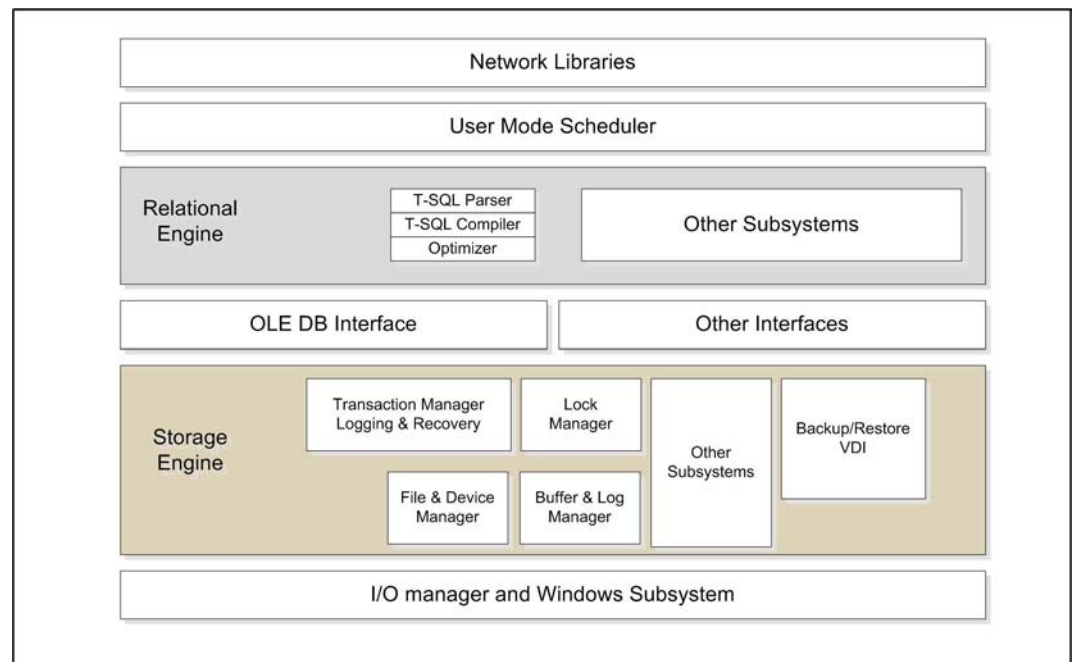


Figure 1-1 Microsoft SQL Server architecture overview

1.3.1 A SQL Server database

A SQL Server database exists as a collection of physical objects (data files and transaction log) that contain the data in the form of tables. As previously mentioned, it is possible to create many databases within a SQL Server instance. Typically, within any SQL Server instance, there are by default four system databases and one or more user databases. Each database has a defined *owner* who may grant or revoke access permissions to other users.

Typical SQL Server logical database objects are:

- ◆ Tables
- ◆ Indexes
- ◆ Views

The database *owner* is associated with a user within a database instance. All permissions and ownership of objects in the database are controlled by the owner's user account. Typically, the owner of a SQL Server database is referred to as user *dbo*. This means that for example, a user "xyz" of database *myDB_1* and a user "abc" of database *myDB_2* may both be referred to as the *dbo* user for their respective databases.

Microsoft SQL Server 2000 and 2005 include the ability to use “Integrated Windows Security” with the Windows Active Directory. This allows access and ownership to be defined to Windows login accounts stored within the Active Directory, rather than the previous SQL Server internal user accounts. It is possible to run either SQL Server authentication; Integrated Windows-based authentication; or a combination of both. Note that it is the responsibility of the various programs and utilities to utilize the appropriate authentication methods. Applications may fail to function correctly if they do not support the installed authentication method. Implemented authentication methods can be changed through the SQL Server properties page.

1.4 Data access

Microsoft SQL Server utilizes a *page* as the basic physical representation of data. A SQL Server data page is 8 KB (8192 bytes). Data pages are then logically aggregated to form *tables*, which are collections of data suitable for quick reference. Each table is a data structure defined with a table name and a set of columns and rows, with data occupying each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.

In SQL Server, an index is an optional structure associated with a table which may increase data retrieval performance. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a range of rows or a specific row. There are two forms of indexes within SQL Server, a *clustered index* or a *non-clustered index*. There can only be one clustered index for a given table, as the clustered index defines the order in which the data is stored in the table. Non-clustered indexes are logically and physically independent of the table data and can therefore be created or dropped at any time. If no clustered index is defined for a table, then the table is referred to as a *heap*.

A *view* may be best considered as a virtualized table, though it should be noted that the view does not persist as an object within the database. The Transact-SQL statement, which defined the view, is stored, and the view may be materialized when referenced. This may be useful to generate a large virtual table which joins data from different tables.

A *filegroup* is a named storage pool that aggregates the physical database data files. As shown in Figure 1-2 on page 1-5, one or more table and index structures make up the database files of a filegroup. The data is stored logically in filegroups and physically in data files that are associated with the corresponding filegroups.

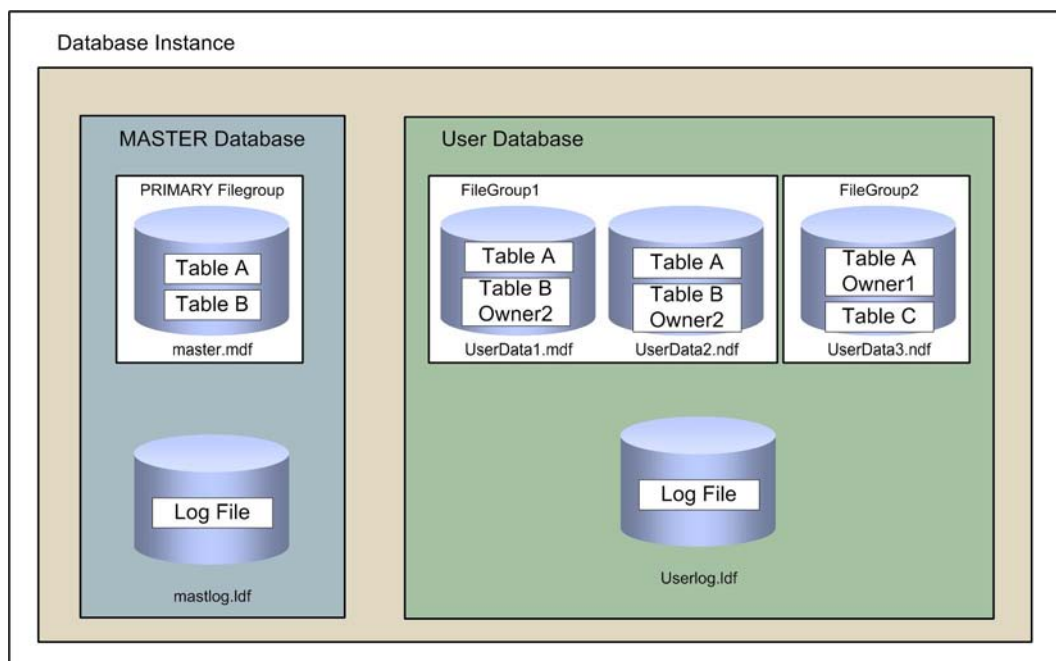


Figure 1-2 SQL Server database architecture

Every SQL Server instance contains four system databases named *master*, *tempdb*, *msdb*, and *model*, which a SQL Server instance creates automatically when it is installed. The master database contains the data dictionary tables and views for the entire SQL Server instance in addition to security information relating to the user databases defined within the instance.

1.5 Microsoft SQL Server physical components

1.5.1 Data files

SQL Server maintains its data and index information in data files. Figure 1-3 on page 1-6 represents the physical layout of a single data file object, and shows the relationship of pages and extents. A page is a logical storage structure that is the smallest unit of storage and I/O used by the SQL Server database. The data page size for both SQL Server 2000 and 2005 is 8 KB (8192 bytes).

When data is added to a given table (or index), space must be allocated for the new data. SQL Server allocates additional space within the relevant data file in a unit size of eight contiguous data pages. SQL Server refers to this eight 8 KB page allocation as an *extent*. Therefore the extent size for SQL Server is 64 KB (65536 bytes).

Extents are either “mixed” or “uniform,” where mixed extents contain both data and index pages which may belong to multiple tables within the database, and uniform extents only contain data or index pages for a single table. Typically, as data is added to a table within a database, only uniform extents are used to store the relevant index or data pages.

As shown in Figure 1-2, several data files may be aggregated into filegroups. Utilizing this functionality provides a facility to constrain given data tables and/or indexes to a

particular filegroup. There is always at least one filegroup, the PRIMARY filegroup. As data or index information is added to a table or index, extents are allocated from the data files which represent the filegroup wherein the table or index exists.

The first file created in the first (PRIMARY) filegroup for the database is unique in that it contains additional information (metadata) regarding the structure of the database itself. This file is referred to as the primary file, and typically is assigned the .mdf extension to signify this functionality. Subsequent data files are assigned the .ndf extension and log files are assigned the .ldf extension. In actuality, file extensions are somewhat irrelevant to SQL Server itself, and are provided simply as a means to make the function of the file more obvious to the user/administrator.

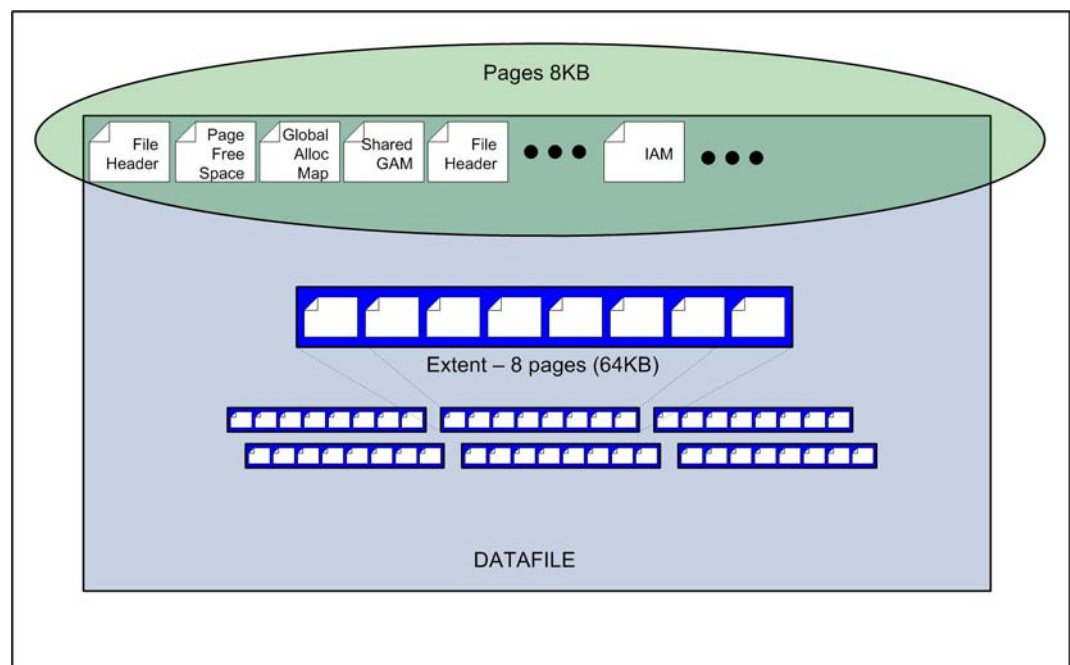


Figure 1-3 SQL Server internal data file logical structure

It is possible to allocate specific data or index information to a named (other than the default) filegroup, such that the named filegroup will only contain the given data or index information. This can be used to isolate particular data or index information, which is unique in some manner, or has a different I/O profile and therefore requires a specific storage location. However, this is not generally required in the majority of SQL Server deployments, and Microsoft fully supports mixing both data and index information within the filegroups—this is the default behavior.

When allocating new data or index storage space, SQL Server will use extents allocated from the filegroup in a “proportional fill” fashion, such that data and index information are evenly distributed amongst the files within the filegroup as data is added. This functionality ensures that there is a more even distribution of data and index information and therefore I/O load. Proportional fill ensures that over time, all the data files have storage allocated based on the available free space in the data file. As such, if one data file within a filegroup has twice as much free space as other data files within the same filegroup, twice as many extent allocations will be made from the larger data file. It is

therefore optimal to equally size all data files within a filegroup, as this results in a more even round-robin allocation of space across all the data files.

1.5.2 Transaction log

In addition to the data and index storage areas, which are the data files, Microsoft SQL Server also creates and maintains a transaction log for each database. The transaction log maintains records relating to changes within the data files. It is possible to define multiple transaction logs for a given SQL Server database. However, only one transaction log is actively receiving writes at any given time. SQL Server takes a serial approach to logging within the transaction log, such that the first log file is fully written to before switching to another transaction log file.

The transaction log does not use allocations such as extents used by the data files. Once created, any single physical transaction log is logically segmented into virtual logs based on internal SQL Server algorithms and the initial size of the transaction log. Once transactional activity begins, transactional information is recorded into a virtual log within the physical log file. A logical sequence number (LSN) is assigned to each transaction log record. Additional information is also recorded in the transaction log record, including the transaction ID of the transaction to which this record belongs, as well as a pointer to any preceding log record for the transaction.

The transaction log is also considered to have an active log component. This active log area is the sequence of log records (which may span multiple virtual logs) relating to transactions in process. This is required to facilitate any recovery operations on pending transactions at the database level. Those virtual logs, which contain data relating to the active log portion, cannot be marked for reuse until their state changes.

The information recorded within the transaction log records is used by Microsoft SQL Server to resolve inconsistencies within the database either in an operational state, or subsequent to an unexpected server failure. In general these are:

- ◆ *rollback* operations (when the data need to be returned to the state before a transaction executing).
- ◆ *roll forward* changes into the data files (when a transaction has completed successfully and the data files have not yet been updated).

Microsoft SQL Server maintains relational integrity by utilizing in-memory structures and the data recorded within the transaction log combined with the data in the data files. Transaction log records always contain any updates to data pages which have been modified by committed transactions. They may also contain updated pages that belong to a transaction that may not have been committed.

In the event of a server crash, SQL Server utilizes the information recorded in the transaction log to return to a transactionally consistent point in time. To ensure that the log always maintains the current state of data, the transaction log is written to synchronously, bypassing all file system buffering. Once updates are recorded in the transaction log, the subsequent updates to the data files may occur asynchronously. It is obvious then, that the state of the data files and the transaction log are not synchronized, but the log always maintains information ahead of the data files.

A point of consistency is created by SQL Server when a checkpoint occurs. A checkpoint forces updated (or dirty) pages out of the memory data buffers and onto disk. At the point when a checkpoint completes, the state of the transaction log and the data files is consistent. The log is required to identify the state of data pages belonging to those transactions that have not been committed and therefore could potentially be rolled back if they abort.

1.6 System databases

Microsoft SQL Server maintains a number of system databases which are used internally by various systems and functions. A list of these databases is provided in Table 1-1 including a description on the function of the database.

Table 1-1 Microsoft SQL Server system databases

Database Name	Function
MASTER	Records system information on all user databases, login accounts, security information, etc.
MODEL	Default “blank” template for new databases.
TEMPDB	Used to maintain temporary sort areas, stored procedures, etc. This database is rebuilt each time the SQL Server instance is started.
MSDB	Used for recording backup/restore events; scheduling and alerts.

System databases themselves comprise a data file and a transaction log. In most instances, there is minimal activity to the system databases, so placement is not performance critical. There may be operational issues that dictate that these databases need to be placed on SAN devices.

Of all the system databases, TEMPDB has specific functionality which differentiates it from the other system databases. The TEMPDB data file(s) and transaction log(s) may need to be appropriately located as they can be the destination of significant I/O load. In general, the default location of these databases is in the %SYSTEMDRIVE% drive. Discussion on placement and requirements for these databases are covered in Chapter 7.

1.7 Microsoft SQL Server instances

It is possible to install multiple separate instances of the SQL Server software on a given Windows Server environment. In general, SQL Server can be installed as a *default* instance, or as a *named* instance. Each instance can be considered a completely autonomous SQL Server environment. The instances can be started or shut down independently, except for actions such as the Windows Server shutdown, which will obviously affect all instances.

Each instance is installed with its own system databases, as previously described, and a separate set of executables for the server components. Thus, it is possible to implement completely separate security mechanisms for each environment, and even to have each instance at a different version or even different SQL Server Service Pack level.

In Figure 1-4, two instances have been installed on a single Windows Server. This results in two sets of executables being installed. The default instance is referenced by the MSSQL directory, the secondary named instance was called DEV, and is therefore referenced as MSSQL\$DEV

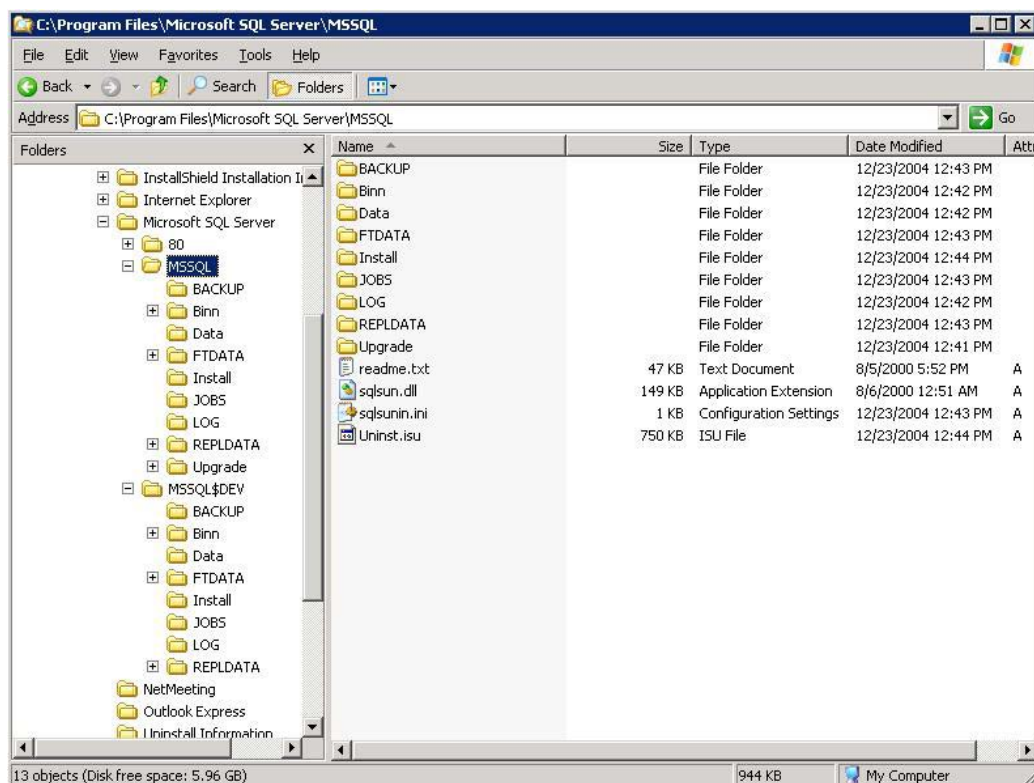


Figure 1-4 Multiple SQL Server instance installation directories

As a result of the ability to have multiple instances executing on a given Windows Server, Microsoft provided extensions to its client connectivity to allow for defining the specific SQL Server instance required. It is possible to reference the default SQL Server instance by simply defining the connection to be the Windows Server hostname itself. For a named instance, the server name needs to include the instance name.

All Microsoft management tools support and display multiple instances installed on a server. In Figure 1-5, Enterprise Manager is used to display the two SQL Server instances created on this SANBOOT1 server.

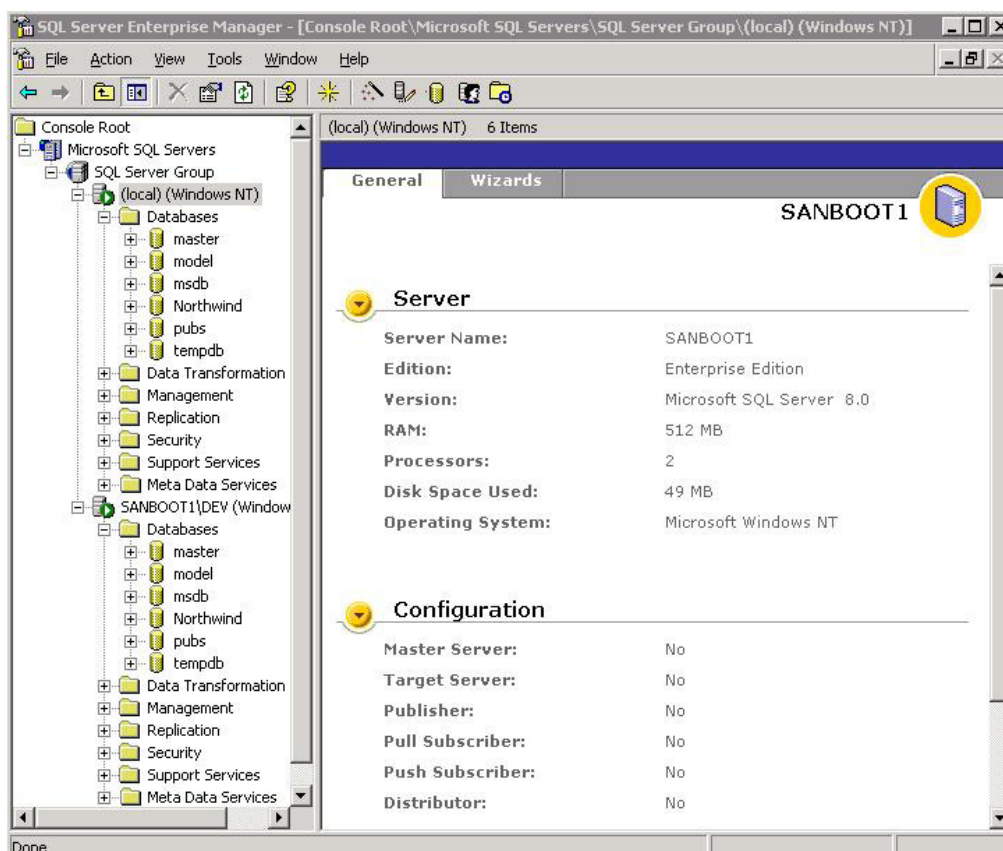


Figure 1-5 SQL Server Enterprise Manager with multiple instances

For example, on the sample Windows Server documented in Figure 1-5, it is possible to connect to the two instances from the `osql` command line interface in the following manner. Each connection specifies the relevant instance. In the first instance, a connection is made to the default instance by only providing the server name, and in the second, the named instance `DEV` is appended to the server name. The server is named `SANBOOT1`. In both cases, a statement is executed to return the name of the server instance.

The figure consists of two screenshots of a Windows Command Prompt window. The top screenshot shows a query to connect to the default instance 'SANBOOT1'. The bottom screenshot shows a query to connect to a named instance 'SANBOOT1\DEV'.

```

C:\>isql -S SANBOOT1 -E
1> select CAST(@SERVERNAME AS VARCHAR(15)) AS INSTANCE
2> go
INSTANCE
-----
SANBOOT1
<1 row affected>
1> exit

C:\>isql -S SANBOOT1\DEV -E
1> select CAST(@SERVERNAME AS VARCHAR(15)) AS INSTANCE
2> go
INSTANCE
-----
SANBOOT1\DEV
<1 row affected>
1> exit

C:\>_

```

Figure 1-6 Connecting to default and named instances

All EMC products that interact with Microsoft SQL Server support multiple SQL Server instances on a given Windows server.

1.8 Microsoft Cluster Service installations

SQL Server supports installations in a Microsoft Cluster Server (MSCS) environment. When installed into a cluster configuration, the installation locations differ from those of a standard local installation.

All data and transaction log files, including those for the system databases must be located on disk devices viewed by the cluster as shared disks. This is a requirement to ensure that all nodes within a given cluster are able to access all required databases within the instance. By default, the SQL Server instance itself will implement a resource dependency on the shared disk resources installed within the resource group. This dependency must be maintained, and any modifications to the database structure that may introduce an additional disk resource, such as adding a data file on a new shared LUN, must be replicated within the resource group, by adding the new disk as a resource within the group, and adding it as a dependency for SQL Server.

Solutions built using EMC's geographically dispersed cluster product SRDF[®]/CE for MSCS implement identical functionality and restrictions. In Figure 1-7 on page 1-12, an SRDF/CE for MSCS implementation is displayed. The dependencies as described are required to be maintained, but now additionally include the disks depending on the SRDF/CE for MSCS resource. In this manner, states for RDF devices are managed appropriately before upper-level services attempting start. In many ways, the implementation of SRDF/CE for MSCS is somewhat transparent to a standard MSCS installation. Figure 1-7 on page 1-12 details an MSCS resource group. In this instance, the view is actually of a SRDF/CE for MSCS resource group, and the SRDF/CE for MSCS resource may be seen as a group member.

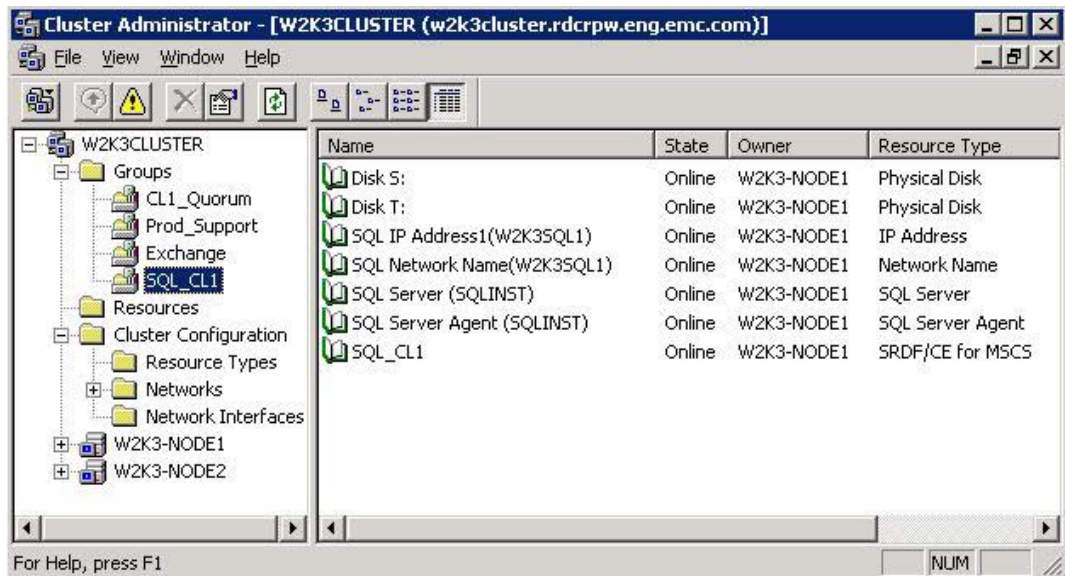


Figure 1-7 SRDF/CE for MSCS resource group for a SQL Server instance

Unlike the shared data files and logs for all the databases within the SQL Server instance, the SQL Server executables are installed locally to each node within the cluster in a similar layout as described in the preceding section. This cluster requirement is also imposed for EMC's geographically dispersed cluster implementation, SRDF/CE for MSCS.

1.9 Backup and recovery interfaces – Virtual Device Interface

Microsoft SQL Server provides an application programming interface (API) to control the interaction of SQL Server backup/restore operations with a disk mirror split. This programmatic interface is utilized by vendors such as EMC to enable specific storage array functionality. For EMC, this facilitates integration of backup/restore operations with disk mirroring technology for local (TimeFinder®) or remote (SRDF) execution.

The Virtual Device Interface (VDI) implementation manages the required operations to perform a controlled quiesce of I/O operations to the data files and transaction logs before suspending write activity, such that disk mirrors may be split.

After execution of a VDI-based backup, the resultant state of the files on the BCV/R2/Clones or SNAP devices represent a valid SQL Server backup of the specified databases, and can be used in the same manner as any other valid SQL Server backup image.

1.10 Microsoft SQL Server and EMC integration

Operationally, the primary level of integration is provided around backup/recovery and restart operations. EMC provides a number of utilities that can be used to facilitate the administration of backup and recovery procedures. These utilities are designed to reduce operating system overhead and mitigate the amount of time required to perform backup and recovery operations.

Products such as the EMC® TimeFinder Integration Module for Microsoft SQL Server (TF/SIM) and EMC's Replication Manager (RM/Local) facilitate low impact backup and recovery procedures, which utilize disk mirror technologies to optimize execution time and enhance availability. The backup images created by these products represent a valid backup image for a SQL Server database by using SQL Server's Virtual Device Interface (VDI).

When executed, the VDI causes the SQL Server instance to issue a checkpoint for the respective databases (which ensures a consistent on-disk image of the database) and suspends write activity to the transaction log and data files for the duration of a disk mirror split. Once complete, the I/O activity is resumed, and a backup indicator is entered into the system databases. This backup marker is a critical requirement for being able to initiate incremental transaction log backups for those databases where a full recovery model is used. It is necessary to process log backups in this environment to ensure that the transaction log does not run out of space.

These disk mirror-based backup images may also be used to facilitate other additional business requirements such as reporting database instances or log-shipping instances. Utilizing EMC Symmetrix® Remote Data Facility (SRDF), customers can create or migrate the backup images to remote arrays, providing the capability of creating disaster recovery or disaster restart solutions.

EMC provides a series of solutions based on consistency technology, which can be used to provide a valid restart point for individual databases or SQL Server instances. More importantly, this functionality allows customers to create larger federated restart solutions. These are generally represented by a number of disparate and/or heterogeneous database environments tied together to create a business application or process.

Restart solutions provide functionality beyond standard backup/recovery processes. It is often impossible to utilize standard backup and recovery processes to create a business point of consistency across multiple environments.

1.11 Additional Microsoft SQL Server tools

Throughout this document, reference will be made to tools and services provided by Microsoft SQL Server and the various tools deployed by default.

- ◆ **SQL Server Enterprise Manager for SQL Server 2000 and SQL Server Management Studio for SQL Server 2005.** This is the graphical user interface into the management and maintenance functions for SQL Server.
- ◆ **Query Analyzer for SQL Server 2000.** Another graphical user interface, Query Analyzer provides a user with various interfaces into the SQL Server environment. In SQL Server 2005, the Query Analyzer functionality has been included within the SQL Server Management Studio. For SQL Server 2000, Query Analyzer lets a user generate and execute Transact SQL (T-SQL) statements against databases. A command line interface is also available in the form of the isql and osql tools. As of SQL Server 2005, the isql utility is no longer available. However, the sqlcmd utility has been added.

- ◆ **Books On Line.** This documentation set covers all the various aspects of SQL Server deployment and management. It provides exhaustive coverage on the correct syntax and options to all T-SQL statements. It also provides discussion on options and features which may be deployed for SQL Server database environments.

All of these tools (as appropriate) are available in the Microsoft SQL Server menu once Microsoft SQL Server has been installed.

Chapter 2 EMC Foundation Products

This chapter presents these topics:

2.1	EMC Symmetrix DMX.....	2-4
2.2	EMC Solutions Enabler base management.....	2-4
2.3	Change Tracker.....	2-6
2.4	EMC Symmetrix Remote Data Facility.....	2-7
2.5	EMC TimeFinder.....	2-21
2.6	EMC Storage Resource Management.....	2-31
2.7	EMC ControlCenter.....	2-35
2.8	EMC PowerPath	2-37
2.9	EMC Replication Manager	2-38

EMC provides many hardware and software products that support application environments on Symmetrix arrays. The following products, which are highlighted and discussed, were used and/or tested with Microsoft SQL Server. This chapter provides a technical overview of the EMC products used in this solution guide.

- ◆ **EMC Symmetrix** — EMC offers an extensive product line of high-end storage solutions targeted to meet the requirements of customers' mission-critical databases and applications. The Symmetrix product line includes the DMX Direct Matrix Architecture series, and the 8000, 5000, and 3000 series family. EMC Symmetrix is a fully redundant, high-availability storage processor, providing non-disruptive component replacements and code upgrades. The Symmetrix system features high levels of performance, data integrity, reliability, and availability.
- ◆ **EMC Solutions Enabler** — Solutions Enabler is a package that contains the SYMAPI runtime libraries and the SYMCLI command line interface. SYMAPI provides the interface to the Symmetrix operating system. SYMCLI are set of commands that can be invoked via the command line or within scripts. These commands can be used to monitor device configuration and status, and to perform control operations on devices and data objects within a storage complex. The target storage environments are typically Symmetrix based, however, CLARiiON® arrays can also be managed via the SYMCLI SRM component.
- ◆ **EMC Symmetrix Remote Data Facility (SRDF)** — SRDF is a business continuity software solution that replicates and maintains a mirror image of data at the storage block level in a remote Symmetrix system. The SRDF component extends the basic SYMCLI command set of Solutions Enabler to include commands that specifically manage SRDF.
 - **EMC SRDF consistency groups** — An SRDF consistency group is a collection of related Symmetrix devices that are configured to act in unison to maintain data integrity. The devices in consistency groups can be spread across multiple Symmetrix systems.
- ◆ **EMC TimeFinder** — TimeFinder is a family of products that enable LUN-based replication within a single Symmetrix array. Data is copied from Symmetrix devices using array-based resources without using host CPU or I/O. The source Symmetrix devices remain online for regular I/O operations while the copies are created. The TimeFinder family has three separate and distinct software products, TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap:
 - **TimeFinder/Mirror** allows users to configure special devices, called business continuance volumes (BCVs), to create a mirror image of Symmetrix standard devices. Using BCVs, TimeFinder creates a point-in-time copy of data that can be repurposed. The TimeFinder/Mirror component extends the basic SYMCLI command set of Solutions Enabler to include commands that specifically manage Symmetrix BCVs and standard devices.
 - **TimeFinder/Clone** allows users to make copies of data simultaneously on multiple target devices from a single source device. The data is available to a target's host immediately upon activation, even if the copy process has not completed. Data may be copied from a single source device to as many as 16 target devices. A source device can be either a Symmetrix standard device or a TimeFinder BCV device. A target device can be any volume in the array of equal or greater size than the source device with which it is paired.

- TimeFinder/Snap allows users to configure special devices in the Symmetrix DMX array called virtual devices (VDEVs) and save area devices (SAVDEVs). These devices can be used to make pointer-based, space-saving copies of data simultaneously on multiple target devices from a single source device. The data is available to a target's host immediately upon creation. Data may be copied from a single source device to as many as 15 VDEVs. A source device can be either a Symmetrix standard device or a TimeFinder BCV device. A target device is a VDEV. A SAVDEV is a special device without a host address that is used to hold the changing contents of the source or target device.
- ◆ EMC Open Replicator — Open Replicator provides block-level data replication and data migration between a Symmetrix DMX™ and secondary heterogeneous storage environments over a SAN. Open Replicator is array-based replication software that runs exclusively on a Symmetrix DMX that enables copy operations independent of the host, operating system, and data type.
- ◆ EMC Change Tracker — EMC Symmetrix Change Tracker software measures changes to data on a Symmetrix volume or group of volumes. Change Tracker software is often used as a planning tool in the analysis and design of configurations that use the EMC TimeFinder or SRDF components to store data at remote sites.
- ◆ Solutions Enabler Storage Resource Management (SRM) Component — The SRM component extends the basic SYMCLI command set of Solutions Enabler to include commands that allow users to systematically find and examine attributes of various objects on the host, within a specified relational database, or in the EMC enterprise storage. The SRM commands provide mapping support for relational databases, file systems, logical volumes and volume groups, as well as performance statistics.
- ◆ EMC ControlCenter™ — EMC ControlCenter is an integrated family of software products that enables users to discover, monitor, automate, provision, and report on storage area networks, host resources, and storage resources across the entire information environment.
- ◆ EMC PowerPath® — PowerPath is host-based software that provides I/O path management. PowerPath operates with several storage systems, on several enterprise operating systems and provides failover and load balancing transparent to the host application and database.
- ◆ EMC SRDF/Cluster Enabler — SRDF/Cluster Enabler (SRDF/CE) for MSCS is one in a family of automated business restart solutions that integrates EMC SRDF with cluster technology. SRDF/CE for MSCS provides disaster recovery protection in geographically distributed clusters.
- ◆ Connectrix® — Connectrix is a Fibre Channel director or switch that moves information throughout the SAN environment, enabling the networked storage solution.
- ◆ EMC Replication Manager — EMC Replication Manager is software that creates replicas of mission-critical databases on disk arrays with traditional tape media. Replication Manager can create a disk replica of data simply, quickly, and automatically. It automates all tasks and/or procedures related to data replication, and reduces the amount of time, resources, and expertise involved with integrating and managing disk-based replication technologies

2.1 EMC Symmetrix DMX

All Symmetrix systems provide advanced data replication capabilities, full mainframe and open systems support, and flexible connectivity options, including Fibre Channel, FICON, ESCON, Gigabit Ethernet, and iSCSI.

Interoperability between Symmetrix storage systems enables current customers to migrate their storage solutions from one generation to the next, protecting their investment even as their storage demands expand.

Symmetrix DMX enhanced cache director technology allows configuration of up to 256 GB of cache. The cache can be logically divided into 32 independent regions providing up to 32 concurrent 500 MB/s transaction throughput.

The Symmetrix on-board data integrity features include:

- ◆ Continuous cache and on-disk data integrity checking and error detection/correction.
- ◆ Fault isolation.
- ◆ Non-disruptive hardware and software upgrades.
- ◆ Automatic diagnostics and phone-home capabilities.

In addition to the models listed previously, for environments that require ultra-high performance, EMC provides DMX1000-P and DMX2000-P systems. These two storage systems are built for extra speed to operate in extreme performance-intensive environments such as decision support, data warehousing, and other high-volume, back-end sequential processing applications.

At the software level, advanced integrity features ensure information is always protected and available. By choosing a mix of RAID 1 (mirroring), RAID 1/0, and high performance RAID 5 (3+1 and 7+1) protection, users have the flexibility to choose the protection level most appropriate to the value and performance requirements of their information. The Symmetrix DMX is EMC's latest generation of high-end storage solutions.

From the perspective of the host operating system, a Symmetrix system appears to be multiple physical devices connected through one or more I/O controllers. The host operating system addresses each of these devices using a physical device name. Each physical device includes attributes, vendor ID, product ID, revision level, and serial ID. The host physical device maps to a Symmetrix device. In turn, the Symmetrix device is a virtual representation of a section of the physical disk called a hypervolume.

2.2 EMC Solutions Enabler base management

The EMC Solutions Enabler kit contains all the base management software that provides a host with SYMAPI-shared libraries and the basic Symmetrix command line interface (SYMCLI). Other optional subcomponents in the Solutions Enabler (SYMCLI) series enable users to extend functionality of the Symmetrix systems. Three principle sub-components are:

- ◆ Solutions Enabler SYMCLI SRDF, SRDF/CG, and SRDF/A
- ◆ Solutions Enabler SYMCLI TimeFinder/Mirror, TimeFinder/CG, TimeFinder/Snap, TimeFinder/Clone
- ◆ Solutions Enabler SYMCLI Storage Resource Management (SRM)

These components will be discussed later in this chapter.

SYMCLI resides on a host system to monitor and perform control operations on Symmetrix storage units. SYMCLI commands are invoked from the host operating system command line or via scripts. SYMCLI commands invoke low-level channel commands to specialized devices on the Symmetrix called gatekeepers. Gatekeepers are very small devices carved from disks in the Symmetrix that act as SCSI targets for the SYMCLI commands.

SYMCLI is used in single command line entries or in scripts to monitor and perform control operations on devices and data objects toward the management of the storage complex. It also monitors device configuration and status of devices that make up the storage environment. To reduce the number of inquiries from the host to the Symmetrix units, configuration and status information is maintained in a host database file.

SYMCLI base commands discussed in this document are listed in Table 2-1.

Table 2-1 SYMCLI base commands

Command	Argument	Description
symdgm		Performs operations on a device group (dg)
	create	Creates an empty device group
	delete	Deletes a device group
	rename	Renames a device group
	release	Releases a device external lock associated with all devices in a device group
	list	Displays a list of all device groups known to this host
symcmg	show	Shows detailed information about a device group and any gatekeeper or BCV devices associated with the device group
		Performs operations on a composite group (cg)
	create	Creates an empty composite group
	add	Adds a device to a composite group
	remove	Removes a device from a composite group
	delete	Deletes a composite group
	rename	Renames a composite group

	release hold unhold list show	Releases a device external lock associated with all devices in a composite group Hold devices in a composite group Unhold devices in a composite group Displays a list of all composite groups known to this host Shows detailed information about a composite group, and any gatekeeper or BCV devices associated with the composite group
symld	add list remove rename show	Performs operations on a device in a device group (dg) Adds devices to a device group and assigns the device a logical name Lists all devices in a device group and any associated BCV devices Removes a device from a device group Renames a device in the device group Shows detailed information about a device in a device group
symbcv	list associate disassociate associate -rdf disassociate -rdf	Performs support operations on BCV pairs Lists BCV devices Associates BCV devices to a device group – required to perform operations on the BCV device Disassociates BCV devices from a device group Associates remotely attached BCV devices to a RDF device group Disassociates remotely attached BCV devices from an RDF device group

2.3 Change Tracker

The EMC Symmetrix Change Tracker software is also part of the base Solutions Enabler SYMCLI management offering. Change Tracker commands are used to measure changes to data on a Symmetrix volume or group of volumes. Change Tracker functionality is often used as a planning tool in the analysis and design of configurations that use the EMC SRDF and TimeFinder components to create copies of production data.

The Change Tracker command (symchg) is used to monitor the amount of changes to a group of hypervolumes. The command timestamps and marks specific volumes for

tracking and maintains a bitmap to record which tracks have changed on those volumes. The bitmap can be interrogated to gain an understanding of how the data on the volume changes over time and to assess the locality of reference of applications.

2.4 EMC Symmetrix Remote Data Facility

The Symmetrix Remote Data Facility (SRDF) component of EMC Solutions Enabler extends the basic SYMCLI command set to enable users to manage SRDF. SRDF is a business continuity solution that provides a host-independent, mirrored data storage solution for duplicating production site data to one or more physically separated target Symmetrix systems. In basic terms, SRDF is a configuration of multiple Symmetrix units whose purpose is to maintain multiple copies of logical volume data in more than one location.

SRDF replicates production or primary (source) site data to a secondary (target) site transparently to users, applications, databases, and host processors. The local SRDF device, known as the source (R1) device, is configured in a partner relationship with a remote target (R2) device, forming an SRDF pair. While the R2 device is mirrored with the R1 device, the R2 device is write-disabled to the remote host. After the R2 device synchronizes with its R1 device, the R2 device can be split from the R1 device at any time, making the R2 device fully accessible again to its host. After the split, the target (R2) device contains valid data and is available for performing business continuity tasks through its original device address.

SRDF requires configuration of specific source Symmetrix volumes (R1) to be mirrored to target Symmetrix volumes (R2). If the primary site is no longer able to continue processing when SRDF is operating in synchronous mode, data at the secondary site is current up to the last I/O transaction. When primary systems are down, SRDF enables fast failover to the secondary copy of the data so that critical information becomes available in minutes. Business operations and related applications may resume full functionality with minimal interruption.

Figure 2-1 illustrates a basic SRDF configuration. As shown in the figure, connectivity between the two Symmetrix is provided using ESCON, Fibre Channel, or Gigabit Ethernet. The connection between the R1 and R2 volumes is through a logical grouping of devices called an RA group. The RA group is independent of the device and composite groups defined and discussed in Section 2.4.3.

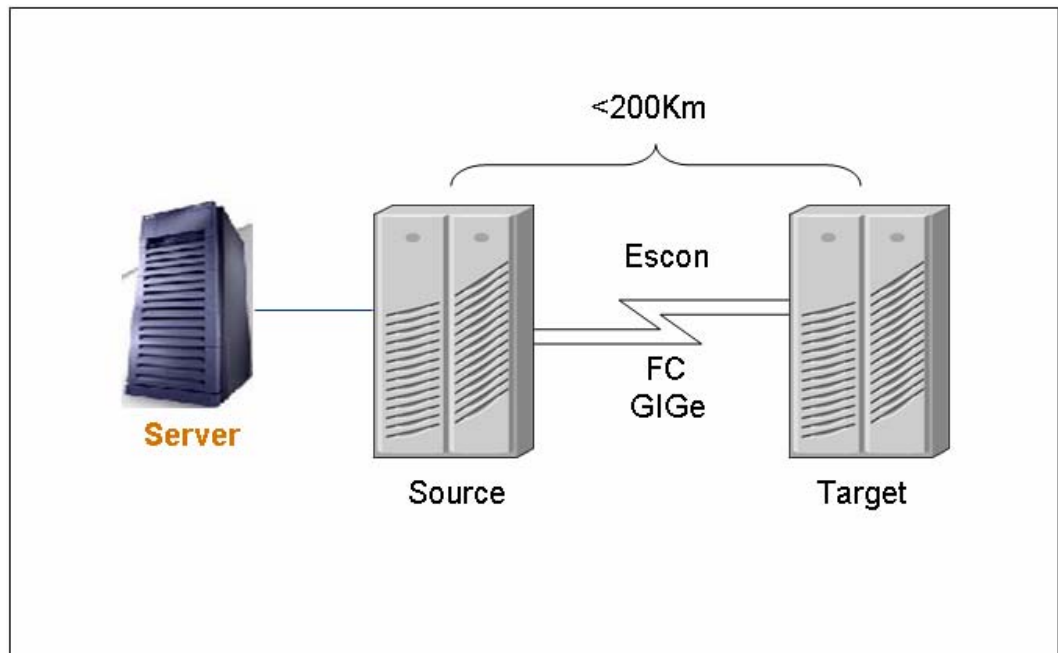


Figure 2-1 Basic synchronous SRDF configuration

2.4.1 SRDF benefits

SRDF offers the following features and benefits:

- ◆ High data availability
- ◆ High performance
- ◆ Flexible configurations
- ◆ Host and application software transparency
- ◆ Automatic recovery from a component or link failure
- ◆ Significantly reduced recovery time after a disaster
- ◆ Increased integrity of recovery procedures
- ◆ Reduced backup and recovery costs
- ◆ Reduced disaster recovery complexity, planning, testing, etc.
- ◆ Support Business Continuity across and between multiple databases on multiple servers and Symmetrix systems.

2.4.2 SRDF modes of operation

SRDF currently supports the following modes of operation:

- ◆ *Synchronous mode (SRDF/S)* provides real-time mirroring of data between the source Symmetrix system(s) and the target Symmetrix system(s). Data is written

simultaneously to the cache of both systems in real time before the application I/O is completed, thus ensuring the highest possible data availability. Data must be successfully stored in both the local and remote Symmetrix units before an acknowledgment is sent to the local host. This mode is used mainly for metropolitan area network distances less than 200 km.

- ◆ *Asynchronous mode (SRDF/A)* maintains a dependent-write consistent copy of data at all times across any distance with no host application impact. Customers wanting to replicate data across long distances historically have had limited options. SRDF/A delivers high-performance, extended-distance replication and reduced telecommunication costs while leveraging existing management capabilities with no host performance impact.
- ◆ *Adaptive copy mode* transfers data from source devices to target devices regardless of order or consistency, and without host performance impact. This is especially useful when transferring large amounts of data during data center migrations, consolidations, and in data mobility environments. Adaptive copy mode is the data movement mechanism of the Symmetrix Automated Replication solution.

2.4.3 SRDF device and composite groups

Applications running on Symmetrix arrays normally involve a number of Symmetrix devices. Therefore, any Symmetrix operation must ensure all related devices are operated upon as a logical group. Defining device or composite groups achieves this.

A device group is a user-defined group of devices that SYMCLI commands can execute upon atomically. Device groups are limited to a single Symmetrix array and RA group. A composite group, on the other hand, can span multiple Symmetrix arrays and RA groups. The device or composite group type may contain R1 or R2 devices and may contain various device lists for standard, BCV, virtual, and remote BCV devices. The **symdg** and **symcg** commands are used to create and manage device and composite groups, respectively.

2.4.4 SRDF consistency groups

An SRDF consistency group is collection of devices defined by a composite group enabled for consistency. Its purpose is to protect data integrity for applications that span multiple RA groups and/or multiple Symmetrix arrays. The protected applications may comprise multiple heterogeneous data resource managers across multiple host operating systems.

An SRDF consistency group uses PowerPath support to provide synchronous disaster restart with zero data loss. Disaster restart solutions that use Consistency groups provide remote restart with short recovery time objectives. Zero data loss implies that all completed transactions at the beginning of a disaster will be available at the target.

When the amount of data for an application becomes very large, the time and resources required for host-based software to protect, back up, or run decision-support queries on these databases become critical factors. The time required to quiesce or shut down the application for offline backup is no longer acceptable. SRDF consistency groups allow users to remotely mirror the largest data environments and automatically split off

dependent-write consistent, restartable copies of applications in seconds without interruption to online service.

A consistency group is a composite group of SRDF devices (R1 or R2) that act in unison to maintain the integrity of applications distributed across multiple Symmetrix units or multiple RA groups within a single Symmetrix. If a source (R1) device in the consistency group cannot propagate data to its corresponding target (R2) device, EMC software suspends data propagation from all R1 devices in the consistency group, halting all data flow to the R2 targets. This suspension, called tripping the consistency group, ensures that a dependent-write consistent R2 copy of the database up to the point in time that the consistency group tripped.

Tripping a consistency group can occur either automatically or manually. Scenarios in which an automatic trip would occur include:

- ◆ One or more R1 devices cannot propagate changes to their corresponding R2 devices
- ◆ The R2 device fails
- ◆ The SRDF directors on the R1 side or R2 side fail

In an automatic trip, the Symmetrix unit completes the write to the R1 device, but indicates that the write did not propagate to the R2 device. EMC software intercepts the I/O and instructs the Symmetrix to suspend all R1 source devices in the consistency group from propagating any further writes to the R2 side. Once the suspension is complete, writes to all of the R1 devices in the consistency group continue normally, but they are not propagated to the target side until normal SRDF mirroring resumes.

An explicit trip occurs when the command `symrdf -cg suspend` or `split` is invoked. Suspending or splitting the consistency group creates an on-demand, restartable copy of the database at the R2 target site. BCV devices that are synchronized with the R2 devices are then split after the consistency group is tripped, creating a second dependent-write consistent copy of the data. During the explicit trip, SYMCLI issues the command to create the dependent-write consistent copy, but may require assistance from PowerPath if I/O is received on one or more R1 devices, or if the SYMCLI commands issued are abnormally terminated before the explicit trip.

An EMC consistency group maintains consistency within applications spread across multiple Symmetrix units in an SRDF configuration, by monitoring data propagation from the source (R1) devices in a consistency group to their corresponding target (R2) devices as depicted in Figure 2-2. Consistency groups provide data integrity protection during a rolling disaster.

Figure 2-2 depicts a dependent-write I/O sequence where a predecessor log write happens before a page flush from a database buffer pool. The log device and data device are on different Symmetrix arrays with different replication paths. Figure 2-2 demonstrates how rolling disasters can be prevented using EMC consistency group technology.

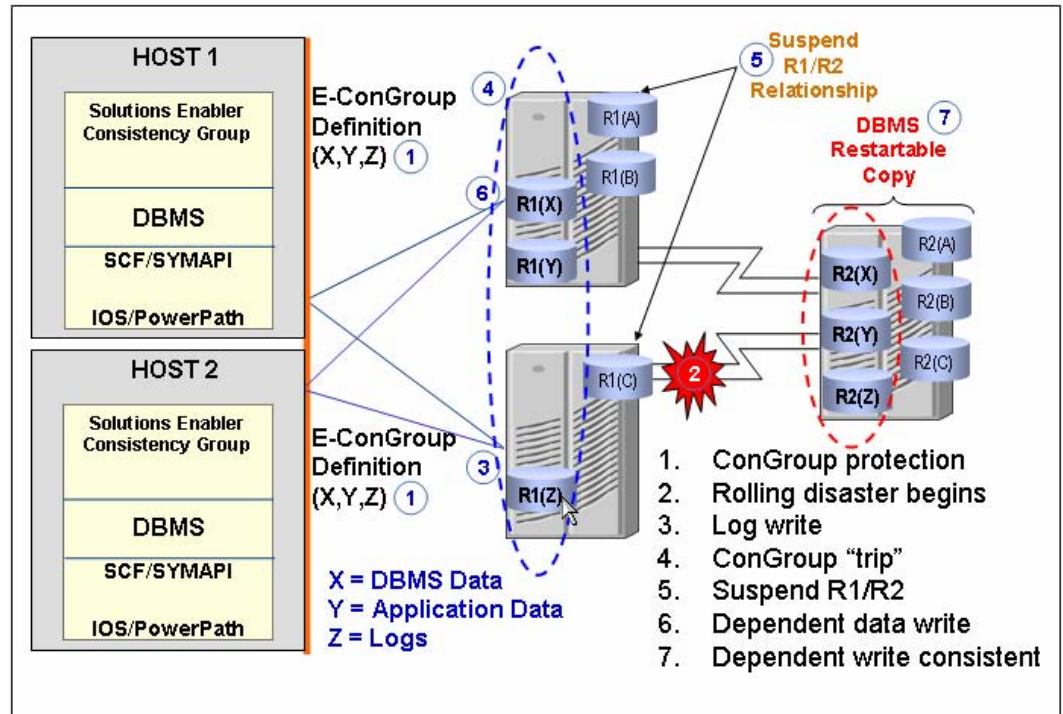


Figure 2-2 SRDF consistency group

A consistency group protection is defined containing volumes X, Y, and Z on the source Symmetrix. This consistency group definition must contain all of the devices that need to maintain dependent-write consistency and reside on all participating hosts involved in issuing I/O to these devices. A mix of CKD (mainframe) and FBA (UNIX/Windows) devices can be logically grouped together. In some cases, the entire processing environment may be defined in a consistency group to ensure dependent-write consistency.

The rolling disaster described previously begins, preventing the replication of changes from volume Z to the remote site.

Since the predecessor log write to volume Z cannot be propagated to the remote Symmetrix system, a consistency group trip occurs.

A ConGroup trip holds the write that could not be replicated along with all of the writes to the logically grouped devices. The writes are held by PowerPath on UNIX/Windows hosts, and IOS on the mainframe hosts long enough to issue two I/Os to all of the Symmetrix arrays involved in the consistency group. The first I/O changes the state of the devices to a suspend-pending state.

The second I/O performs the suspend actions on the R1/R2 relationships for the logically grouped devices which immediately disables all replication to the remote site. This allows other devices outside of the group to continue replicating, provided the communication links are available. After the relationship is suspended, the completion of the predecessor write is acknowledged back to the issuing host. Furthermore, all I/Os that were held during the consistency group trip operation are released.

After the second I/O per Symmetrix completes, the I/O is released, allowing the predecessor log write to complete to the host. The dependent data write is issued by the DBMS and arrives at X but is not replicated to the R2(X).

When a complete failure occurs from this rolling disaster, the dependent-write consistency at the remote site is preserved. If a complete disaster does not occur and the failed links are activated again, the consistency group replication can be resumed. It is recommended to create a copy of the dependent-write consistent image while the resume takes place. Once the SRDF process reaches synchronization the dependent-write consistent copy is achieved at the remote site.

2.4.5 SRDF terminology

This section describes various terms related to SRDF operations.

2.4.5.1 Suspend and resume operations

Practical uses of suspend and resume operations usually involve unplanned situations in which an immediate suspension of I/O between the R1 and R2 devices over the SRDF links is desired. In this way, data propagation problems can be stopped. When suspend is used with consistency groups, immediate backups can be performed off the R2s without affecting I/O from the local host application. I/O can then be resumed between the R1 and R2 and return to normal operation.

2.4.5.2 Establish and split operations

The **establish** and **split** operations are normally used in planned situations in which use of the R2 copy of the data is desired without interfering with normal write operations to the R1 device. Splitting a point-in-time copy of data allows access to the data on the R2 device for various business continuity tasks. The ease of splitting SRDF pairs to provide exact database copies makes it convenient to perform scheduled backup operations, reporting operations, or new application testing from the target Symmetrix data while normal processing continues on the source Symmetrix system.

The R2 copy can also be used to test disaster recovery plans without manually intensive recovery drills, complex procedures, and application service interruptions. Upgrades to new versions can be tested or changes to actual code can be made without affecting the online production server. For example, modified server code can be run on the R2 copy of the database until the upgraded code runs with no errors before upgrading the production server.

In cases where an absolute real-time copy of the production data is not essential, users may choose to split the SRDF pair periodically and use the R2 copy for queries and report generation. The SRDF pair can be reestablished periodically to provide incremental updating of data on the R2 device. The ability to refresh the R2 device periodically provides the latest information for data processing and reporting.

2.4.5.3 Failover and failback operations

Practical uses of failover and failback operations usually involve the need to switch business operations from the production site to a remote site (failover) or the opposite

(failback). Once failover occurs, normal operations continue using the remote (R2) copy of synchronized application data. Scheduled maintenance at the production site is one example of where failover to the R2 site might be needed.

Testing of disaster recovery plans is the primary reason to temporarily fail over to a remote site. Traditional disaster recovery routines involve customized software and complex procedures. Offsite media must be either electronically transmitted or physically shipped to the recovery site. Time-consuming restores and the application of logs usually follow. SRDF failover/failback operations significantly reduce the recovery time by incrementally updating only the specific tracks that have changed; this accomplishes in minutes what might take hours for a complete load from dumped database volumes.

2.4.5.4 Update operation

The update operation allows users to resynchronize the R1s after a failover while continuing to run application and database services on the R2s. This function helps reduce the amount of time that a failback to the R1 side takes. The update operation is a subset of the failover/failback functionality. Practical uses of the R1 update operation usually involve situations in which the R1 becomes almost synchronized with the R2 data before a failback, while the R2 side is still online to its host. The -until option, when used with update, specifies the target number of invalid tracks that are allowed to be out of sync before resynchronization to the R1 completes.

2.4.5.5 Concurrent SRDF

Concurrent SRDF means having two target R2 devices configured as concurrent mirrors of one source R1 device. Using a concurrent SRDF pair allows the creation of two copies of the same data at two remote locations. When the two R2 devices are split from their source R1 device, each target site copy of the application can be accessed independently.

2.4.5.6 R1/R2 swap

Swapping R1/R2 devices of an SRDF pair causes the source R1 device to become a target R2 device and vice versa. Swapping SRDF devices allows the R2 site to take over operations while retaining a remote mirror on the original source site. Swapping is especially useful after failing over an application from the R1 site to the R2 site. SRDF swapping is available with Enginuity version 5567 or higher.

2.4.5.7 Data mobility

Data mobility is an SRDF configuration that restricts SRDF devices to operating only in adaptive copy mode. This is a lower-cost licensing option that is typically used for data migrations. It allows data to be transferred asynchronously from source to target, and is not designed as a solution for DR requirements unless used in combination with TimeFinder.

2.4.5.8 Dynamic SRDF

Dynamic SRDF allows the creation of SRDF pairs from non-SRDF devices while the Symmetrix unit is in operation. Historically, source and target SRDF device pairing has been static and changes required assistance from EMC personnel. However, beginning with the SRDF component of EMC Solutions Enabler version 5.0 running on Symmetrix units using Enginuity™ version 5568, users can use SRDF-capable, non-SRDF devices in creating and synchronizing SRDF pairs. This feature provides greater flexibility in deciding where to copy protected data.

Beginning with Solutions Enabler version 5.2 running on Symmetrix units using Enginuity version 5669, dynamic RA groups can be created in a SRDF switched fabric environment. An RA group represents a logical connection between two Symmetrix units. Historically, RA groups were limited to those static RA groups defined at configuration time. However, RA groups can now be created, modified, and deleted while the Symmetrix unit is in operation. This feature provides greater flexibility in forming SRDF-pair-associated links.

2.4.6 SRDF control operations

This section describes typical control operations that can be performed by the Solutions Enabler `symrdf` command.

Solutions Enabler SYMCLI SRDF commands perform the following basic control operations on SRDF devices:

- ◆ *Establish* synchronizes an SRDF pair by initiating a data copy from the source (R1) side to the target (R2) side. This operation can be a full or incremental establish. Changes on the R2 volumes are discarded by this process.
- ◆ *Restore* resynchronizes a data copy from the target (R2) side to the source (R1) side. This operation can be a full or incremental restore. Changes on the R1 volumes are discarded by this process.
- ◆ *Split* stops mirroring for the SRDF pair(s) in a device group and write-enables the R2 devices.
- ◆ *Swap* exchanges the source (R1) and target (R2) designations on the source and target volumes.
- ◆ *Failover* switches data processing from the source (R1) side to the target (R2) side. The source side volumes (R1), if still available, are write-disabled.
- ◆ *Failback* switches data processing from the target (R2) side to the source (R1) side. The target side volumes (R2), if still available, are write-disabled.

2.4.6.1 Establishing an SRDF pair

Establishing an SRDF pair initiates remote mirroring—the copying of data from the source (R1) device to the target (R2) device. SRDF pairs come into existence in two different ways:

- ◆ At configuration time through the pairing of SRDF devices. This is a static pairing configuration discussed earlier.
- ◆ Anytime during a dynamic pairing configuration in which SRDF pairs are created on demand.

A full establish (`symrdf establish -full`) is typically performed after an SRDF pair is initially configured and connected via the SRDF links. After the first full establish, users can perform an incremental establish, where the R1 device copies to the R2 device only the new data that was updated while the relationship was split or suspended.

To initiate an establish operation on all SRDF pairs in a device or composite group, all pairs must be in the split or suspended state. The `symrdf query` command is used to check the state of SRDF pairs in a device or composite group.

When the establish operation is initiated, the system write-disables the R2 device to its host and merges the track tables. The merge creates a bitmap of the tracks that need to be copied to the target volumes discarding the changes on the target volumes. When the establish operation is complete, the SRDF pair is in the synchronized state. The R1 device and R2 device contain identical data, and continue to do so until interrupted by administrative command or unplanned disruption. Figure 2-3 depicts SRDF establish and restore operations:

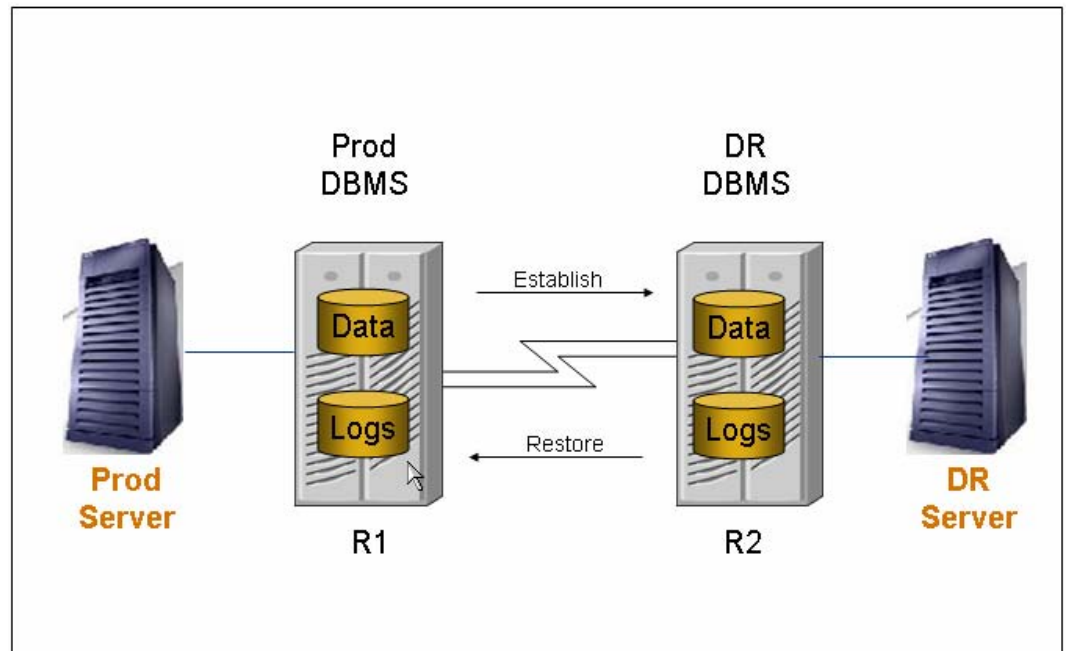


Figure 2-3 SRDF establish and restore control operations

The establish operation may be initiated by any host connected to either Symmetrix array, provided that an appropriate device group has been built on that host. The following command initiates an incremental establish operation for all SRDF pairs in the device group named *device_group*:

```
symrdf -g <device_group> establish -noprompt
```

2.4.6.2 Splitting an SRDF pair

When read/write access to a target (R2) device is necessary, the SRDF pair can be split. When the split completes, the target host can access the R2 device for write operations. The R2 device contains valid data and is available for business continuity tasks or restoring data to the R1 device if there is a loss of data on that device.

While an SRDF pair is in the split state, local I/O to the R1 device can still occur. These updates are not propagated to the R2 device immediately. Changes on each Symmetrix system are tracked through bitmaps and are reconciled when normal SRDF mirroring operations are resumed. To initiate a split, an SRDF pair must already be in one of the following states:

- ◆ Synchronized
- ◆ Suspended
- ◆ R1 updated
- ◆ SyncInProgress (if the `-symforce` option specified for the split)

The split operation may be initiated from either host. The following command initiates a split operation on all SRDF pairs in the device group named *device_group*:

```
symrdf -g <device_group> split -noprompt
```

The `symrdf split` command provides exactly the same functionality as the `symrdf suspend` and `symrdf rw_enable R2` commands together. Furthermore, the split and suspend operations have exactly the same consistency characteristics as SRDF consistency groups. Therefore, when SRDF pairs are in a single device group, users can split the SRDF pairs in the device group as shown previously and have restartable copies on the R2 devices. If the application data spans multiple Symmetrix units or multiple RA groups, include SRDF pairs in a consistency group to achieve the same results.

2.4.6.3 Restoring an SRDF pair

When the target (R2) data must be copied back to the source (R1) device, the SRDF restore command is used (see Figure 2-3). After an SRDF pair is split, the R2 device contains valid data and is available for business continuance tasks (such as running a new application) or restoring data to the R1 device. Moreover, if the results of running a new application on the R2 device need to be preserved, moving the changed data and new application to the R1 device is another option.

Users may perform a full or incremental restore. A full restore operation copies the entire contents of the R2 device to the R1 device. An incremental restore operation is much quicker because it copies only new data that was updated on the R2 device while the SRDF pair was split. Any tracks on the R1 device that changed while the SRDF pair was split are replaced with the corresponding tracks on the R2 device.

To initiate a restore, an SRDF pair must already be in the split state. The restore operation can be initiated from either host. The following command initiates an incremental restore operation on all SRDF pairs in the device group named *device_group* (add the `-full` option for a full restore):

```
symrdf -g <device_group> restore -noprompt
```

The restore operation is complete when the R1 and R2 devices contain identical data. The SRDF pair is then in a synchronized state and may be reestablished by initiating the `symrdf establish` command.

2.4.7 Failover and failback operations

Having a synchronized SRDF pair allows users to switch data processing operations from the source site to the target site if operations at the source site are disrupted or if downtime must be scheduled for maintenance. This switchover from source to target is called failover. When operations at the source site are back to normal, a failback operation is used to reestablish I/O communications links between source and target, resynchronize the data between the sites, and resume normal operations on the R1 devices as shown in Figure 2-4.

Figure 2-4 illustrates the failover and failback operations. The failover and failback operations relocate the processing from the source site to the target site or vice versa. This may or may not imply movement of data.

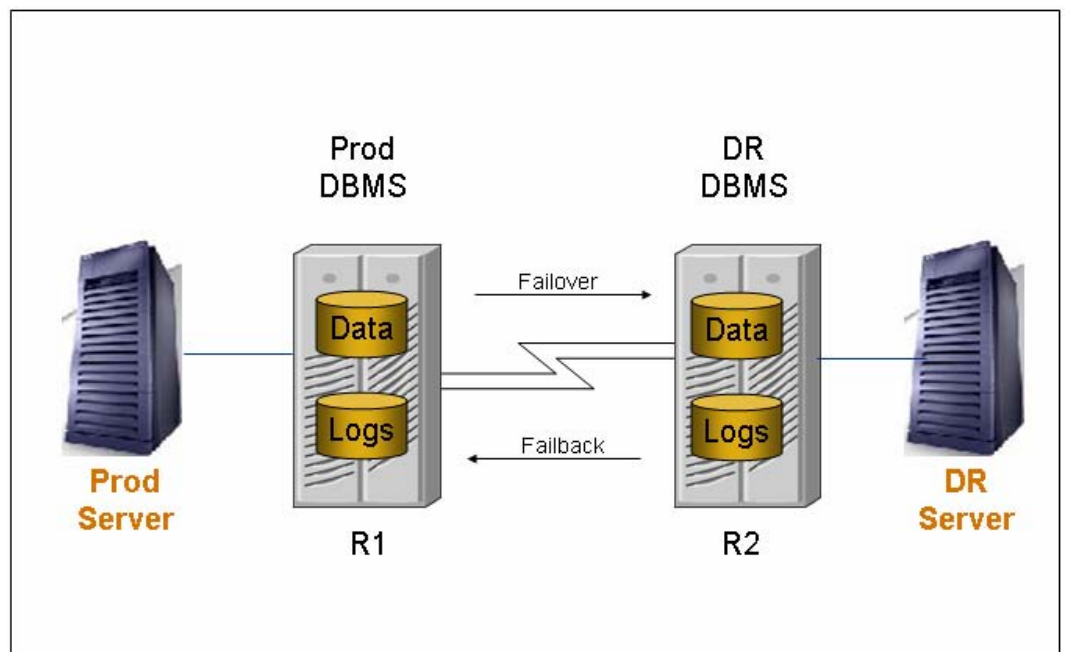


Figure 2-4 SRDF failover and failback control operations

2.4.7.1 Failover

Scheduled maintenance or storage system problems can disrupt access to production data at the source site. In this case, a failover operation can be initiated from either host to make the R2 device read/write-enabled to its host. Before issuing the failover, all applications services on the R1 volumes must be stopped. This is because the failover operation will make the R1 volumes read-only. The following command initiates a failover on all SRDF pairs in the device group *device_group*:

```
symrdf -g <device_group> failover -noprompt
```

To initiate a failover, the SRDF pair must already be in one of the following states:

- ◆ Synchronized
- ◆ Suspended
- ◆ R1 updated
- ◆ Partitioned (when invoking this operation at the target site)

The failover operation:

- ◆ Suspends data traffic on the SRDF links
- ◆ Write-disables the R1 devices
- ◆ Write-enables the R2 volumes.

2.4.7.2 Failback

To resume normal operations on the R1 side, a failback (R1 device takeover) operation is initiated. This means read/write operations on the R2 device must be stopped, and read/write operations on the R1 device must be started. When the failback command is initiated, the R2 becomes read-only to its host, while the R1 becomes read/write-enabled to its host. The following command performs a failback operation on all SRDF pairs in the device group *device_group*:

```
symrdf -g <device_group> failback -noprompt
```

The SRDF pair must already be in one of the following states for the failback operation to succeed:

- ◆ Failed over
- ◆ Suspended and write-disabled at the source
- ◆ Suspended and not ready at the source
- ◆ R1 Updated
- ◆ R1 UpdInProg

The failback operation:

- ◆ Write-enables the R1 devices.
- ◆ Performs a track table merge to discard changes on the R1s.
- ◆ Transfers the changes on the R2s.
- ◆ Resumes traffic on the SRDF links.
- ◆ Write-disables the R2 volumes.

2.4.8 SRDF/A (Asynchronous) operations

SRDF Asynchronous (SRDF/A) operations are supported with EMC Solutions Enabler SYMCLI 5.3 and higher. Symmetrix DMX arrays running Enginuity 5670 code and higher support SRDF/A mode for SRDF devices. Asynchronous mode provides a dependent-write consistent, point-in-time image on the target (R2) devices, which are only slightly behind the source (R1) device. SRDF/A session data is transferred to the remote Symmetrix array in predefined timed cycles or delta sets. This functionality requires an SRDF/A license.

The default configuration for SRDF/A is for the target volumes to be at most two cycles behind the data state of the source volumes. In the default configuration for SRDF/A, this would equate to 60 seconds.

SRDF/A provides a long-distance replication solution with minimal impact on performance that preserves data consistency within application data in the specified business process. This level of protection is intended for DR environments that always need a DBMS restartable copy of data at the R2 site. Dependent-write consistency is guaranteed on a delta set boundary. In the event of a disaster at the R1 site or if SRDF links are lost during data transfer, a partial delta set of data will be discarded. This preserves consistency on the R2 with a maximum data loss of two SRDF/A cycles.

2.4.8.1 Enabling SRDF/A

To set remotely mirrored pairs in the prod group to the asynchronous mode (SRDF/A), enter:

```
symrdf -g <device_group> set mode async -noprompt
```

2.4.8.2 SRDF/A benefits

SRDF/A mode provides the following features and benefits:

- ◆ Lowers operational costs for long-distance data replication with application consistency.
- ◆ Promotes efficient link utilization, resulting in lower link bandwidth.
- ◆ Maintains a dependent-write consistent, point-in-time image on the R2 devices at all times.
- ◆ Supports all current SRDF topologies, including point-to-point and switched fabric.
- ◆ Operates at any given distance without adding response time to the R1 host.

- ◆ Includes all hosts and data emulation types supported by the Symmetrix array (such as FBA, CKD, and AS/400).
- ◆ Minimizes the impact imposed on the back-end directors.
- ◆ Provides an application response time equivalent to writing to local non-SRDF devices.
- ◆ Allows restore, failover, and failback capability between the R1 and R2 sites.

2.4.8.3 SRDF/A capable devices

A listing of SRDF/A capable devices is displayed with the following command:

```
symrdf list -rdfa
```

2.4.8.4 SRDF/A session status

When asynchronous mode is set for a group of devices, the SRDF/A capable devices in the group are considered part of the SRDF/A session. To check SRDF/A session status, use either of the following commands:

```
symdg show <device_group>
symcg show <composite_group>
```

The session status is displayed as active or inactive:

- ◆ Inactive — The SRDF/A devices are either ready or not ready on the link
- ◆ Active — The SRDF/A mode is activated and the SRDF/A session data is currently being transmitted in operational cycles to the R2.

2.4.9 EMC SRDF/Cluster Enabler solutions

EMC SRDF/Cluster Enabler (SRDF/CE) for MSCS is an integrated solution that combines SRDF and clustering protection over distance. EMC SRDF/CE provides disaster-tolerant capabilities that enable a cluster to span geographically separated Symmetrix systems. It operates as a software extension (MMC snap-in) to the Microsoft Cluster Service (MSCS).

SRDF/CE achieves this capability by exploiting SRDF disaster restart capabilities. SRDF allows the MSCS cluster to have two identical sets of application data in two different locations. When cluster services are failed over or failed back, SRDF/CE is invoked automatically to perform the SRDF functions necessary to enable the requested operation.

Figure 2-5 illustrates the hardware configuration of two, four-node, geographically distributed EMC SRDF/CE clusters using bidirectional SRDF.

For more detailed information on the EMC SRDF/CE offering, refer to the *Integrating [EMC SRDF and Cluster Technology for High Availability and Disaster Restart Solutions Guide](#)*.

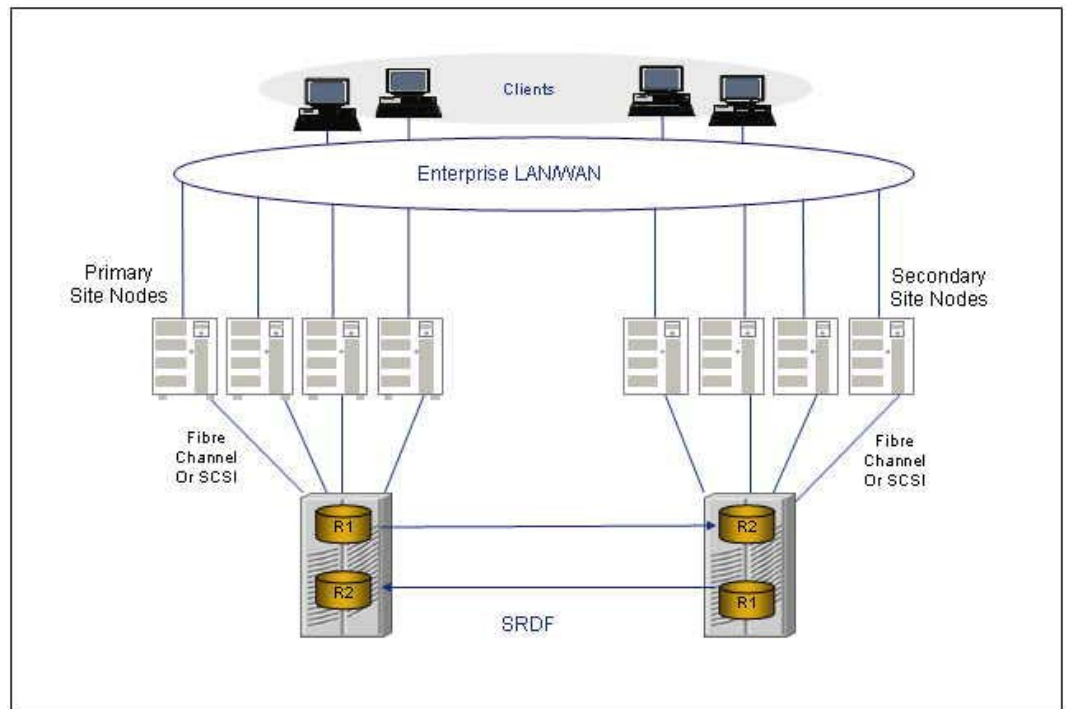


Figure 2-5 Geographically distributed four-node EMC SRDF/CE clusters

2.5 EMC TimeFinder

The SYMCLI TimeFinder component extends the basic SYMCLI command set to include TimeFinder or business continuity commands that allow control operations on device pairs within a local replication environment. This section specifically describes the functionality of:

- ◆ TimeFinder/Mirror — General monitor and control operations for business continuance volumes (BCV).
- ◆ TimeFinder/CG — Consistency groups
- ◆ TimeFinder/Clone — Clone copy sessions
- ◆ TimeFinder/Snap — Snap copy sessions

Commands such as `symmir` and `symbcv` perform a wide spectrum of monitor and control operations on standard/BCV device pairs within a TimeFinder/Mirror environment. The TimeFinder/Clone command, `symclone`, creates a point-in-time copy of a source device on nonstandard device pairs (such as standard/standard, BCV/BCV). The TimeFinder/Snap command, `symsnap`, creates virtual device copy sessions between a source device and multiple virtual target devices. These virtual devices only store pointers to changed data blocks from the source device, rather than a full copy of the data. Each product requires a specific license for monitoring and control operations.

Configuring and controlling remote BCV pairs requires EMC SRDF business continuity software discussed in section 2.4. The combination of TimeFinder with SRDF provides for multiple local and remote copies of production data.

Figure 2-6 illustrates Application Usage for TimeFinder/Mirror configuration in a Symmetrix system.

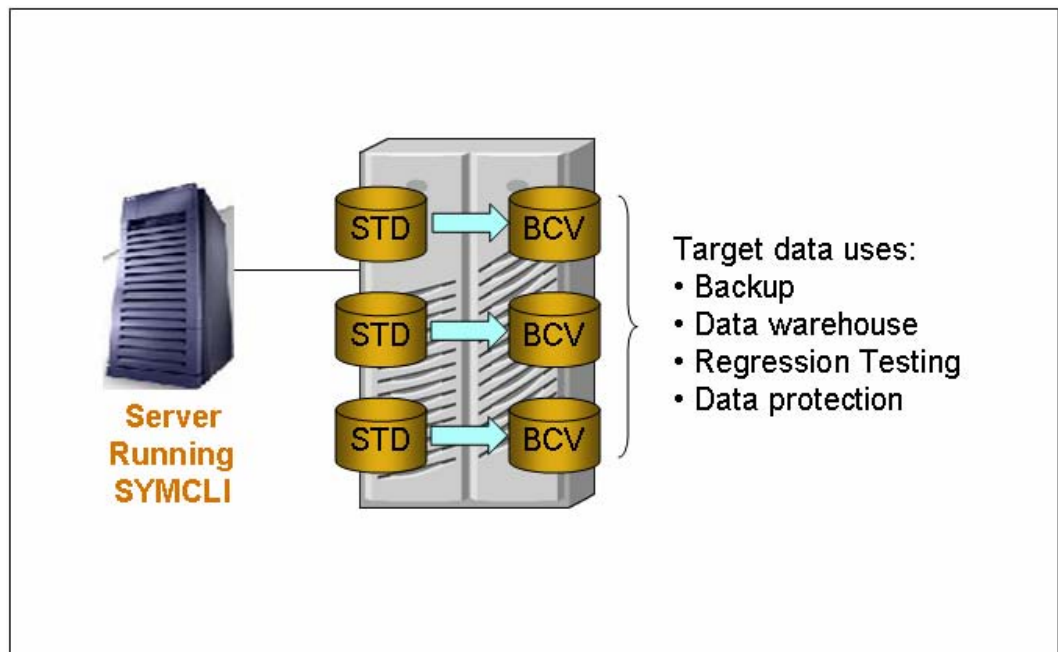


Figure 2-6 EMC Symmetrix configured with standard volumes and BCVs

2.5.1 TimeFinder/Mirror Establish operations

A BCV device can be fully or incrementally established. After configuration and initialization of a Symmetrix unit, BCV devices contain no data. BCV devices, like standard devices, can have unique host addresses and can be online and ready to the host(s) to which they are connected. A full establish must be used the first time the standard devices are paired with the BCV devices. An incremental establish of a BCV device can be performed to resynchronize any data that has changed on the standard since the last establish.

When BCVs are established, they are inaccessible to any host.

Symmetrix arrays allow up to four mirrors for each logical volume. The mirror positions are commonly designated M1, M2, M3, and M4. An unprotected BCV can be the second, third, or fourth mirror position of the standard device. A host, however, logically views the Symmetrix M1/M2 mirrored devices as a single device.

To assign a BCV as a mirror of a standard Symmetrix device, the `symmir establish` command is used. One method of establishing a BCV pair is to allow the standard/BCV device-pairing algorithm to arbitrarily create BCV pairs from multiple devices within a device group:

```
symmir -g <device_group> establish -full -noprompt
```

With this method, TimeFinder/Mirror first checks for any attach assignments (specifying a preferred BCV match from among multiple BCVs in a device group).

TimeFinder/Mirror then checks if there are any pairing relationships among the devices. If either of these previous conditions exists, TimeFinder/Mirror uses these assignments.

2.5.2 TimeFinder split operations

Splitting a BCV pair is a TimeFinder/Mirror action that detaches the BCV from its standard device and makes the BCV ready for host access. When splitting a BCV, the system must perform housekeeping tasks that may require a few milliseconds on a busy Symmetrix array. These tasks involve a series of steps that result in separation of the BCV from its paired standard:

- ◆ I/O is suspended briefly to the standard device.
- ◆ Write pending tracks for the standard device that have not yet been written out to the BCV are duplicated in cache to be written to the BCV.
- ◆ The BCV is split from the standard device.
- ◆ The BCV device status is changed to ready.

2.5.2.1 Regular split

A regular split is the type of split that has existed for TimeFinder/Mirror since its inception. With a regular split (before Enginuity version 5568), I/O activity from the production hosts to a standard volume was not accepted until it was split from its BCV pair. Therefore, applications attempting to access the standard or the BCV would experience a short wait during a regular split. Once the split was complete, no further overhead was incurred.

Beginning with Enginuity version 5568, any **split** operation is an instant split. A regular split is still valid for earlier versions and for current applications that perform regular split operations. However, current applications that perform regular splits with Enginuity version 5568 actually perform an instant split.

By specifying the **-instant** flag on the command line, an instant split with Enginuity 5x66 and 5x67 can be performed. Since 5568, this option is no longer required because instant split mode has become the default behavior.

2.5.2.2 Instant split

An instant split shortens the wait period during a split by dividing the process into a foreground split and a background split. During an instant split, the system executes the foreground split almost instantaneously and returns a successful status to the host. This instantaneous execution allows minimal I/O disruptions to the production volumes. Furthermore, the BCVs are accessible to the hosts as soon as the foreground process is complete. The background split continues to split the BCV pair until it is complete. When the **-instant** option is included or defaulted, SYMCLI returns immediately after

the foreground split, allowing other operations while the BCV pair is splitting in the background.

The following operation performs an instant split on all BCV pairs in *device_group*, and allows SYMCLI to return to the server process while the background split is in progress.

```
symmir -g <device_group> split -instant -noprompt
```

The following symmir query command example checks the progress of a split on *composite_group*. The `-bg` option is provided to query the status of the background split:

```
symmir -cg <composite_group> query -bg
```

2.5.3 TimeFinder restore operations

A BCV device can be used to fully or incrementally restore data on the standard volume. Like the full establish operation, a full restore operation copies the entire contents of the BCV devices to the standard devices. Optionally, users may specify devices in a device group, composite group, or device file as in the following examples:

```
symmir -g <device_group> -full restore -noprompt
symmir -cg <composite_group> -full restore -noprompt
symmir -f[file] <FileName> -full -sid <###> restore -noprompt
```

The incremental restore process accomplishes the same thing as the full restore process with a major time-saving exception. The BCV copies to the standard device only new data that was updated on the BCV device while the BCV pair was split. The data on the corresponding tracks of the BCV device also overwrites any changed tracks on the standard device. This maximizes the efficiency of the resynchronization process. This process is useful, for example, if, after testing or validating an updated version of a database or a new application on the BCV device is completed, a user wants to migrate and utilize a copy of the tested data or application on the production standard device.

An incremental restore of a BCV volume to a standard volume is only possible when the two volumes have an existing TimeFinder relationship

2.5.4 TimeFinder consistent split

TimeFinder consistent split allows users to split off a dependent-write consistent, restartable image of an application without interrupting online services. Consistent split helps to avoid inconsistencies and restart problems that can occur when splitting an application-related BCV without first quiescing or halting the application. Consistent split is implemented using the Engenuity Consistency Assist (ECA) feature. This functionality requires a TimeFinder/CG license.

2.5.4.1 Engenuity Consistency Assist

The Engenuity Consistency Assist feature of the Symmetrix operating environment can be used to perform consistent split operations across multiple heterogeneous environments. This functionality requires a TimeFinder/CG license and uses the `-consistent` option of the `symmir` command.

Using ECA to consistently split BCV devices from the standards, a control host with no database or a database host with a dedicated channel to gatekeeper devices must be available. The dedicated channel cannot be used for servicing other devices or to freeze I/O. For example, to split a device group, execute:

```
symmir -g <device_group> split -consistent -noprompt
```

Figure 2-7 illustrates an Enginuity Consistency Assist split across three database hosts that access devices on a Symmetrix array:

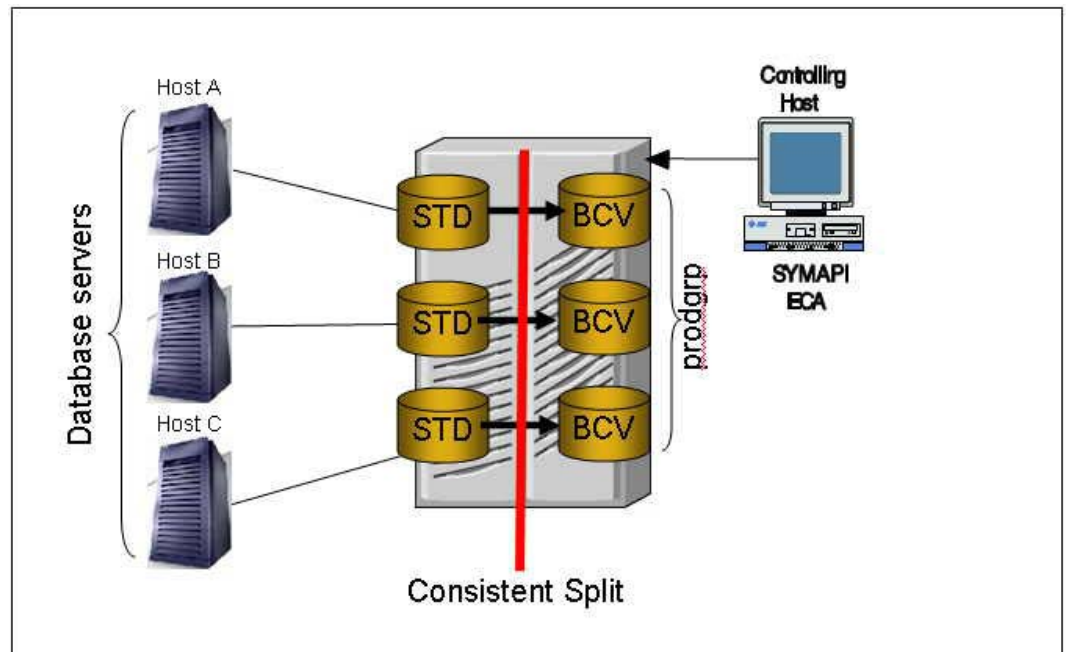


Figure 2-7 ECA consistent split across multiple database associated hosts

Symmetrix device or composite groups must be created on the controlling host for the target application to be consistently split. Device groups can be created to include all of the required devices for maintaining business continuity. For example, if a device group is defined that includes all of the devices being accessed by Hosts A, B, and C (see Figure 2-7), then all of the BCV pairs related to those hosts can be consistently split with a single command.

However, if a device group is defined that includes only the devices accessed by Host A, then the BCV pairs related to Host A can be split without affecting the other hosts. The solid vertical line in Figure 2-7 represents the ECA holding of I/Os during an instant split process, creating a dependent-write consistent image in the BCVs.

Figure 2-8 illustrates the use of local consistent split with a database management system (DBMS).

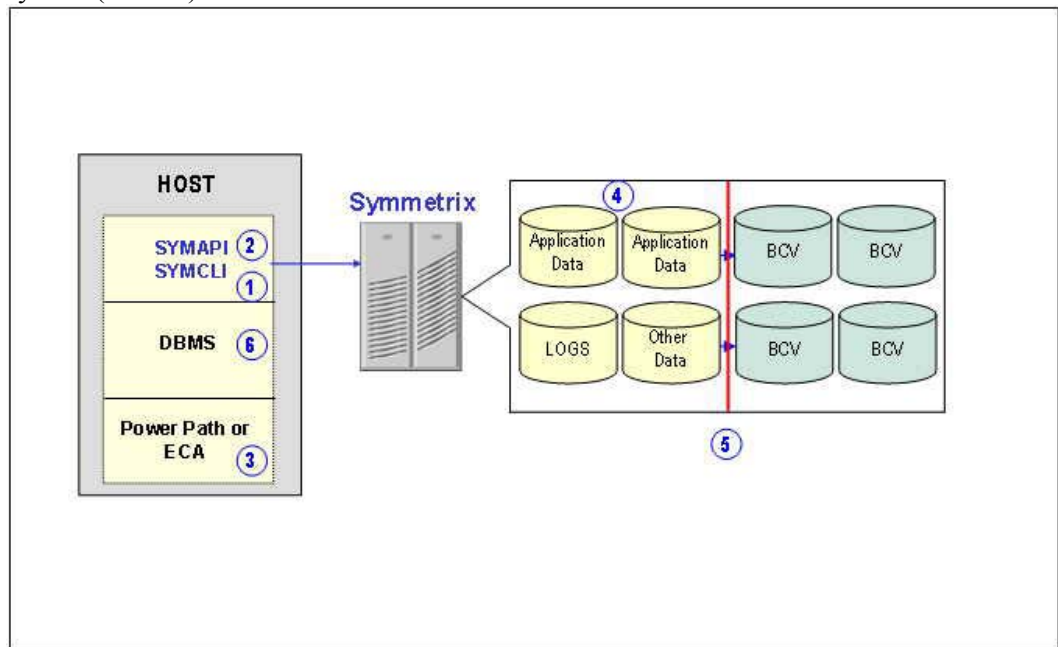


Figure 2-8 ECA consistent split on a local Symmetrix

When a split command is issued with ECA from the production host, a consistent database image is created using the following sequence of events shown in Figure 2-8:

1. The Symmetrix API (SYMAPI) identifies the standard devices that hold the database.
2. SYMAPI validates that all identified BCV pairs can be split.
3. SYMAPI sends a suspend write message to ECA.
4. ECA suspends writes to the standard devices that hold the database. The DBMS cannot write to the devices and subsequently waits for these devices to become available before resuming any further write activity. Read activity to the device is not affected unless attempting to read from a device which has a write queued against it.
5. SYMAPI sends an instant split request to all BCV pairs in the specified device group and waits for the Symmetrix to acknowledge that the foreground split has occurred. SYMAPI then sends a resume I/O message to ECA.
6. The application resumes writing to the production devices.

The BCV devices now contain a restartable copy of the production data that is consistent up until the time of the instant split. The production application is unaware that the split or suspend/resume operation occurred. When the application on the secondary host is started using the BCVs, there is no record of a successful shutdown. Therefore, the

secondary application instance views the BCV copy as a crashed instance and proceeds to perform the normal crash recovery sequence to restart.

When performing a consistent split, it is a good practice to issue host-based commands that will commit any data that has not been written to disk before the split. For example on UNIX systems, the sync command can be run. From a database perspective, a checkpoint or equivalent should be executed.

2.5.5 TimeFinder reverse split

BCVs can be mirrored to guard against data loss through physical drive failures. A reverse split is applicable for a BCV that is configured to have two local mirrors. It is generally used to recover from an unsuccessful restore operation. When data is restored from the BCV to the standard device, any writes that occur while the standard is being restored alter the original copy of data on the BCV's primary mirror. If the original copy of BCV data is needed again at a later time, it can be restored to the BCV's primary mirror from the BCV's secondary mirror using a reverse split. For example, whenever logical corruption is reintroduced to a database during a recovery process (following a BCV restore), both the standard device and the primary BCV mirror are left with corrupted data. In this case, a reverse split can restore the original BCV data from a BCV's secondary mirror to its primary mirror.

This is particularly useful when performing a restore and immediately restarting processing on the standard devices when the process may have to be restarted many times.

Reverse split is not available when protected restore is used to return the data from the BCVs to the standards.

2.5.6 TimeFinder/Clone operations

Symmetrix TimeFinder/Clone operations using SYMCLI have been available since Engenuity version 5568. TimeFinder/Clone can create up to 16 copies from a source device onto target devices. Unlike TimeFinder/Mirror, TimeFinder/Clone does not require the traditional standard to BCV device pairing. Instead, TimeFinder/Clone allows any combination of source and target devices. For example, a BCV can be used as the source device, while another BCV can be used as the target device. Any combination of source and target devices can be used. Additionally, TimeFinder/Clone does not use the traditional mirror positions the way that TimeFinder/Mirror does. Because of this, TimeFinder/Clone is a useful option when more than three copies of a source device are desired.

Normally, one of the three copies is used to protect the data against hardware failure.

The source and target devices must be the same emulation type (FBA or CKD). The target device must be equal or greater in size than the source. Clone copies of striped or concatenated metavolumes can also be created providing the source, and target metavolumes must be identical in configuration. Once activated, the target device can be instantly accessed by a target's host, even before the data is fully copied to the target device.

TimeFinder/Clone copies are appropriate in situations where multiple copies of production data is needed for testing, backups, or report generation. Clone copies can also be used to reduce disk contention and improve data access speed by assigning users to copies of data rather than accessing the one production copy. A single source device may maintain as many as 16 relationships that can be a combination of BCV's, clones and snaps. When using the -copy option of TimeFinder/Clones, one can copy up to four full data copies simultaneously, without disruption to database production activity.

2.5.6.1 Clone copy sessions

TimeFinder/Clone functionality is controlled via copy sessions, which pair the source and target devices. Sessions are maintained on the Symmetrix array and can be queried to verify the current state of the device pairs. A copy session must first be created to define and set up the TimeFinder/Clone devices. The session is then activated, enabling the target device to be accessed by its host. When the information is no longer needed, the session can be terminated. TimeFinder/Clone operations are controlled from the host by using the symclone command to create, activate, and terminate the copy sessions.

Figure 2-9 illustrates a copy session where the controlling host creates a TimeFinder/Clone copy of standard device DEV001 on target device DEV005, using the symclone command.

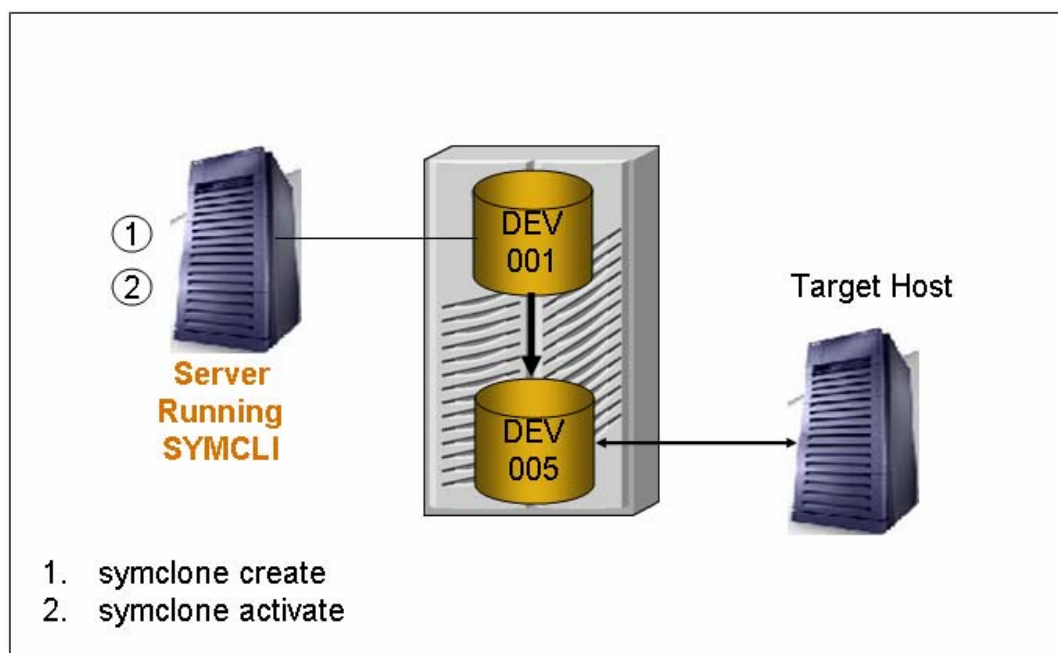


Figure 2-9 Creating a copy session using the symclone command

The symclone command is used to enable cloning operations. The cloning operation happens in two phases: creation and activation. The creation phase builds bitmaps of the source and target that are later used during the activation or copy phase. The creation of a symclone pairing does not start copying of the source volume to the target volume. To create clone sessions on all the standards and BCVs in the device group *device_group*, use the following command:

```
symclone -g <device_group> create -noprompt
```

The activation of a clone enables the copying of the data. The data may start copying immediately if the **—copy** keyword is used. If the **—copy** keyword is not used, tracks are only copied when they are accessed from the target volume or when they are changed on the source volume.

Activation of the clone session established in the previous create command can be accomplished using the following command:

```
symclone -g <device_group> activate -noprompt
```

2.5.7 TimeFinder/Snap operations

Symmetrix DMX arrays running 5669 or later version of the Enginuity operating environment provide another technique to create copies of application data. The functionality, called TimeFinder/Snap, allows users to make pointer-based, space-saving copies of data simultaneously on multiple target devices from a single source device. The data is available for access instantly. TimeFinder/Snap allows data to be copied from a single source device to as many as 15 target devices (a sixteenth Snap session is reserved for restoring data from one of the targets). A source device can be either a Symmetrix standard device or a BCV device controlled by TimeFinder/Mirror. The target device is a Symmetrix virtual device (VDEV) that consumes negligible physical storage through the use of pointers to track changed data.

A VDEV is a host-addressable Symmetrix device with special attributes created when the Symmetrix DMX system is configured. However, unlike a BCV which contains a full volume of data, a VDEV is a logical-image device that offers a space-saving way to create instant, point-in-time copies of volumes. Any updates to a source device after its activation with a virtual device, causes the pre-update image of the changed tracks to be copied to a save device. The virtual device's indirect pointer is then updated to point to the original track data on the save device, preserving a point-in-time image of the volume. TimeFinder/Snap uses this copy-on-first-write technique to conserve disk space, since only changes to tracks on the source cause any incremental storage to be consumed.

The **symsnap create** and **symsnap activate** are used to create source/target Snap pair.

Table 2-2 summarizes some of the differences between devices used in TimeFinder/Snap operations.

Table 2-2 TimeFinder device type summary

Device	Description
Virtual device	A logical-image device that saves disk space through the use of pointers to track data that is immediately accessible after activation. Snapping data to a virtual device uses a copy-on-first-write technique.
Save device	A device that is not host-accessible but accessed only through the virtual devices that point to it. Save devices provide a pool of physical space to store snap copy data to which virtual devices point.
BCV	A full volume mirror that has valid data after fully synchronizing with its source device. It is accessible only when split from the source device

that it is mirroring.

2.5.7.1 Snap copy sessions

TimeFinder/Snap functionality is managed via copy sessions, which pair the source and target devices. Sessions are maintained on the Symmetrix array and can be queried to verify the current state of the devices. A copy session must first be created—a process which defines the Snap devices in the operation. On subsequent activation, the target virtual devices become accessible to its host. Unless the data is changed by the host accessing the virtual device, the virtual device always presents a frozen point-in-time copy of the source device at the point of activation. When the information is no longer needed, the session can be terminated.

TimeFinder/Snap operations are controlled from the host by using the `symsnap` command to create, activate, terminate, and restore the TimeFinder/Snap copy sessions. The TimeFinder/Snap operations described in this section explain how to manage the devices participating in a copy session through SYMCLI.

Figure 2-10 illustrates a virtual copy session where the controlling host creates a copy of standard device DEV001 on target device VDEV005.

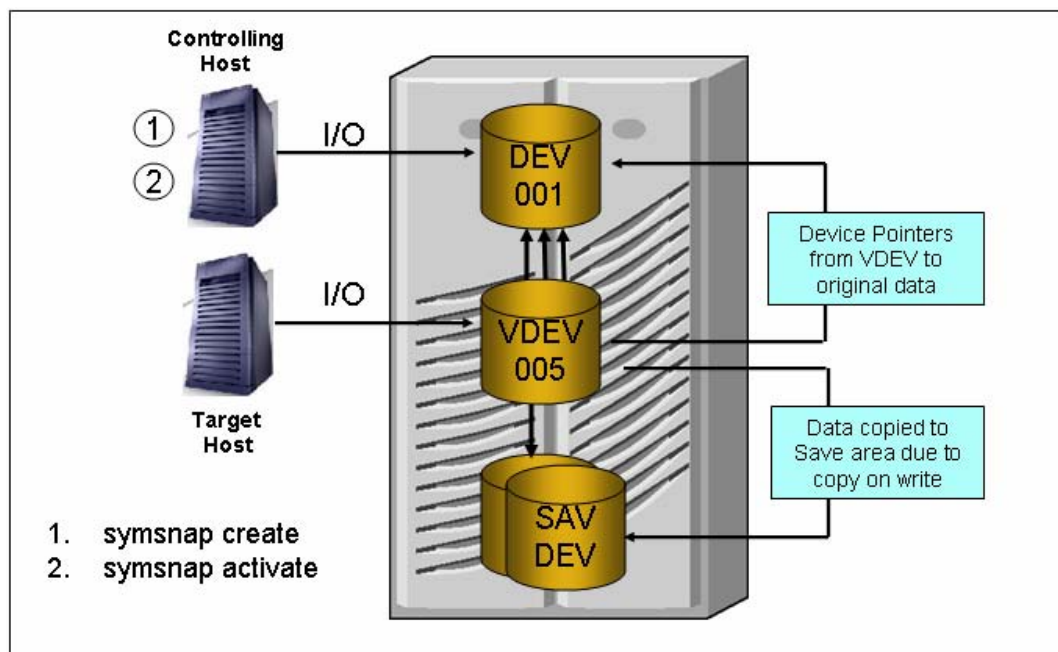


Figure 2-10 TimeFinder/Snap copy of a standard device to a VDEV

The `symsnap` command is used to enable TimeFinder/Snap operations. The Snap operation happens in two phases: creation and activation. The creation phase builds bitmaps of the source and target that are later used to manage the changes on the source and target. The creation of a snap pairing does not copy the data from the source volume to the target volume. To create snap sessions on all the standards and BCVs in the device group *device_group*, use the following command:

```
symsnap -g <device_group> create -noprompt
```

The activation of a snap enables the protection of the source data tracks. When protected tracks are changed on the source volume, they are first copied into the save pool and the VDEV pointers are updated to point to the changed tracks in the save pool. When tracks are changed on the VDEV, the data is written directly to the save pool and the VDEV pointers are updated in the same way.

Activation of the snap session created in the previous **create** command can be accomplished using the following command:

```
symsnap -g <device_group> activate -noprompt
```

2.6 EMC Storage Resource Management

The Storage Resource Management (SRM) component of EMC Solutions Enabler extends the basic SYMCLI command set to include SRM commands that allow users to discover and examine attributes of various objects on a host or in the EMC storage enterprise.

SYMCLI commands support SRM in the following areas:

- ◆ Data objects and files
- ◆ Relational databases
- ◆ File systems
- ◆ Logical volumes and volume groups
- ◆ Performance statistics

SRM allows users to examine the mapping of storage devices and the characteristics of data files and objects. These commands allow the examination of relationships between extents and data files or data objects, and how they are mapped on storage devices. Frequently, SRM commands are used with TimeFinder and SRDF to create point-in-time copies for backup and restart.

Figure 2-11 outlines the process of how SRM commands are used with TimeFinder in a database environment.

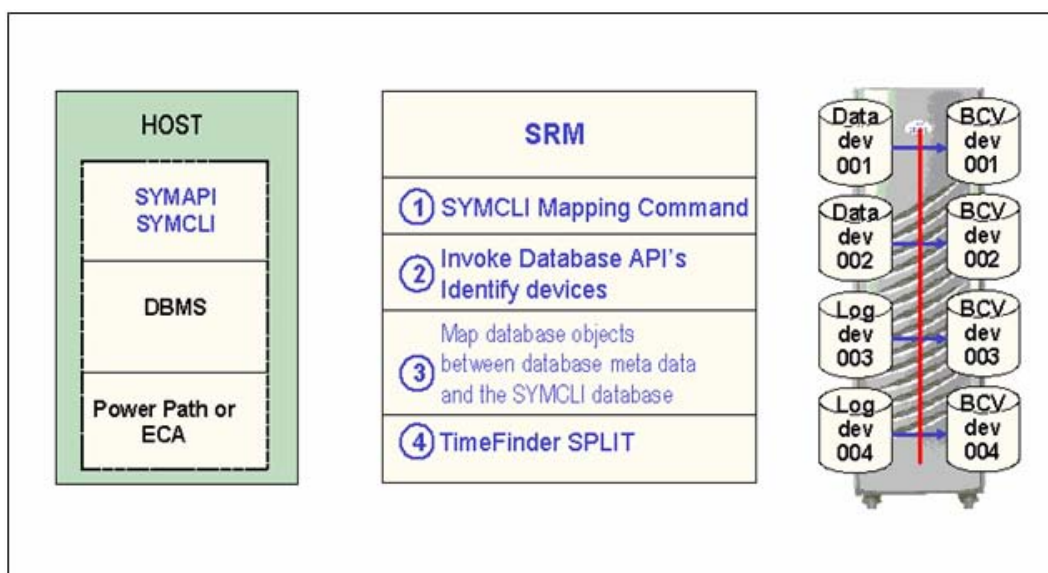


Figure 2-11 SRM commands

EMC Solutions Enabler with a valid license for TimeFinder and SRM is installed on the host. In addition, the host must also have PowerPath or use ECA, and must be utilized with a supported DBMS system. As discussed in section 2.5.2, when splitting a BCV, the system must perform housekeeping tasks that may require a few seconds on a busy Symmetrix array. These tasks involve a series of steps (shown in Figure 2-11) that result in the separation of the BCV from its paired standard:

1. Using the SRM base mapping commands, first query the Symmetrix unit to display the logical-to-physical mapping information about any physical device, logical volume, file, directory, and/or file system.
2. Using the database mapping command, query the Symmetrix to display physical and logical database information.
3. Next, use the database mapping command to translate:
 - The devices of a specified database into a device group or a consistency group, or
 - The devices of a specified tablespace into a device group or a consistency group.
4. The BCV is split from the standard device.

Table 2-3 lists the SYMCLI commands that can be used to examine the mapping of data objects.

Table 2-3 Data object SRM commands

Command	Argument	Action
symrslv	pd	Displays logical to physical mapping information about any physical device.
	lv	Displays logical to physical mapping information

	file	about a logical volume. Displays logical to physical mapping information about a file.
	dir	Displays logical to physical mapping information about a directory.
	fs	Displays logical to physical mapping information about a file system.

SRM commands allow users to examine the host database mapping and the characteristics of a database. The commands provide listings and attributes that describe various databases, their structures, files, tablespaces, and user schemas. Typically, the database commands work with Oracle, Informix, SQL Server, Sybase, Microsoft Exchange, SharePoint Portal Server, and DB2 LUW database applications.

Table 2-4 lists the SYMCLI commands that can be used to examine the mapping of database objects.

Table 2-4 Data object mapping commands

Command	Argument	Action
symrdb	list	Lists various physical and logical database objects: Current relational database instances available Tablespaces, tables, files, or schemas of a database Files, segments, or tables of a database tablespace or schema
	show	Shows information about a database object: Tablespace, tables, file, or schema of a database File, segment, or a table of a specified tablespace or schema Translates the devices of a specified database into a device group.
	rdb2dg	Translates the devices of a specified tablespace into a composite group or a consistency group.
	rdb2cg	Translates the devices of a specified tablespace into a composite group. Only data database files are translated.
	tbs2cg	Translates the devices of a specified tablespace into a device group. Only data database files are translated.
	tbs2dg	

The SYMCLI file system SRM command allows users to investigate the file systems that are in use on the operating system. The command provides listings and attributes that describe file systems, directories, and files, and their mapping to physical devices and extents.

Table 2-5 lists the SYMCLI command that can be used to examine the file system mapping.

Table 2-5 File system SRM command

Command	Argument	Action
symhostfs	list	Displays a list of file systems, files, or directories
	show	Displays more detail information about a file system or file system object.

SYMCLI logical volume SRM commands allow users to map logical volumes to display a detailed view of the underlying storage devices. Logical volume architecture defined by a Logical Volume Manager (LVM) is a means for advanced applications to improve performance by the strategic placement of data.

Table 2-6 lists the SYMCLI commands that can be used to examine the logical volume mapping.

Table 2-6 File system SRM commands

Command	Argument	Action
symvg	deport	Deports a specified volume group so it can be imported later.
	import	Imports a specified volume group.
	list	Displays a list of volume groups defined on the host system by the logical volume manager.
	rescan	Rescans all the volume groups.
	show	Displays more detail information about a volume group.
	vg2cg	Translates volume groups to composite groups.
	vg2dg	Translates volume groups to device groups.
symlv	List	Displays a list of logical volumes on a specified volume group.
	show	Displays detail information (including extent data) about a logical volume.

SRM performance statistics commands allow users to retrieve statistics about a host's CPU, disk, and memory.

Table 2-7 lists the statistics commands.

Table 2-7 SRM statistics command

Command	Argument	Action
symhost	Show	Displays host configuration information.
	Stats	Displays performance statistics.

2.7 EMC ControlCenter

EMC ControlCenter is an integrated family of products that enables users to discover, monitor, automate, provision, and report on networks, host resources, and storage resources across the entire information environment. Figure 2-12 presents the ControlCenter product family and the broad scope of manageability that ControlCenter provides in managing the storage enterprise.

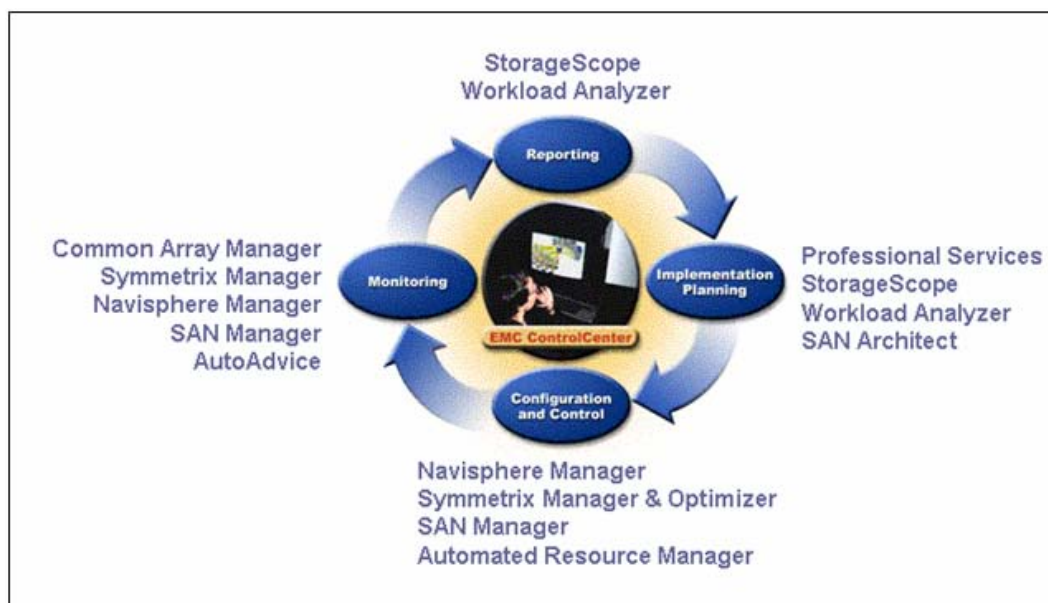


Figure 2-12 ControlCenter family overview

The ControlCenter suite of products provides end-to-end management of storage networks, storage devices, and other storage resources. From the Web Console, ControlCenter can manage, monitor, configure and report on the following components:

- ◆ Storage components for Symmetrix, CLARiiON, Celerra®, HDS, HP StorageWorks, IBM ESS, and NetApp filers
- ◆ Host components, such as logical volume managers, file systems, databases, backup applications, and host processes
- ◆ Connectivity components, such as Fibre Channel switches and hubs
- ◆ Databases, such as Oracle, Sybase, and Microsoft SQL Server

Every physical and logical element that ControlCenter manages is known as a managed object. From a console anywhere on the network, ControlCenter shows an integrated

and consolidated view of the storage environment, enabling the storage administrator to monitor, report, manage, and configure each managed object.

ControlCenter is designed for use in a heterogeneous environment where information is distributed in multivendor storage networks with disparate computing, connectivity, and storage arrays. The ability to manage host applications and their associated storage needs across heterogeneous environments from a single interface simplifies storage management and makes it possible to implement cross-platform storage-wide strategies.

At the heart of ControlCenter is a common foundation and infrastructure, which provides scalability, usability, and information sharing across all ControlCenter applications. This design enables applications to span storage array, storage network, and host management functions. The core components of ControlCenter are known as the Open Integration Components. These provide the infrastructure (ECC Server, Repository, and Store), common services, and a common user interface (both a web-based console for monitoring and a java-based application console for full management functionality). The common services provide the following features:

- ◆ Discover storage arrays, connectivity devices, and hosts.
- ◆ Map the storage topology.
- ◆ Map the relationship of storage structures from databases and file systems to their logical and physical location within the storage array.
- ◆ Display the properties of any object.
- ◆ Monitor the real-time or historical performance of objects.
- ◆ Monitor the status of objects and issue alerts.

The following ControlCenter applications are discussed briefly below:

- ◆ SAN Manager — provides integrated network discovery, topology, and alert capabilities. It actively controls SAN management functions such as zoning and LUN masking.
- ◆ Automated Resource Manager (ARM) — simplifies and automates storage resource management (SRM) across the enterprise. It provides policy-based storage provisioning for file systems and volume groups from a host perspective. This includes determining the location of free storage pool and automating the allocation of that storage to the host. ARM manages backup and restore operations from within ControlCenter, and increases the availability of storage environments.
- ◆ StorageScope™ — provides a variety of SRM reports to help users assess their current storage environment and determine future storage requirements. StorageScope users can create custom report layouts, export reports into CSV or XML format, and integrate StorageScope with third-party applications such as billing or customer service.
- ◆ Performance Manager — collects and presents statistics for performance analysis. It collects metrics for Windows, UNIX, and MVS hosts, file systems, Oracle databases, Fibre Channel switches, and Symmetrix and CLARiON storage arrays.

Performance Manager also allows users to generate web-based reports on scheduled intervals; hourly, daily, weekly.

- ◆ Symmetrix Manager — monitors, automates operations, and provisions storage for Symmetrix arrays.
- ◆ Symmetrix Optimizer — automatically tunes Symmetrix arrays for optimal performance based on business requirements.
- ◆ SRDF-TimeFinder Manager — monitoring, provisioning, and automation of SRDF and TimeFinder in the storage environment.
- ◆ Common Array Manager — simplifies the management of multivendor storage environments by providing monitoring and reporting capabilities for HP StorageWorks, Hitachi/HP/Sun, and IBM ESS storage arrays.
- ◆ SAN Architect — an online, subscription-based product that provides users with an easy-to-use SAN design, validation, and modeling solution. Users can model changes to existing SANs or design new topologies with immediate validation against EMC internal knowledge bases. SAN Architect guides users through the entire design process and automatically validates array, switch, and topology choices.
- ◆ CLARiiON, Celerra, and Network Appliance management — ControlCenter can also be used with other storage element managers like Navisphere® Manager, Connectrix Manager, and non-EMC array and switch managers. Many of these can be launched from the console to deliver extended functionality or capabilities.

2.8 EMC PowerPath

EMC PowerPath is host-based software that works with networked storage systems to intelligently manage I/O paths. PowerPath manages multiple paths to a storage array. Supporting multiple paths enables recovery from path failure because PowerPath automatically detects path failures and redirects I/O to other available paths. PowerPath also uses sophisticated algorithms to provide dynamic load balancing for several kinds of path management policies that the user can set. With the help of PowerPath, systems administrators are able to ensure that applications on the host have highly available access to storage and perform optimally at all times.

A key feature of path management in PowerPath is dynamic, multipath load balancing. Without PowerPath, an administrator must statically load balance paths to logical devices to improve performance. For example, based on current usage, the administrator might configure three heavily used logical devices on one path, seven moderately used logical devices on a second path, and 20 lightly used logical devices on a third path. As I/O patterns change, these statically configured paths may become unbalanced, causing performance to suffer. The administrator must then reconfigure the paths, and continue to reconfigure them as I/O traffic between the host and the storage system shifts in response to use changes.

Designed to use all paths concurrently, PowerPath distributes I/O requests to a logical device across all available paths, rather than requiring a single path to bear the entire I/O burden. PowerPath can distribute the I/O for all logical devices over all paths shared by those logical devices, so that all paths are equally burdened. PowerPath load balances

I/O on a host-by-host basis, and maintains statistics on all I/O for all paths. For each I/O request, PowerPath intelligently chooses the least-burdened available path, depending on the load-balancing and failover policy in effect. In addition to improving I/O performance, dynamic load balancing reduces management time and downtime because administrators no longer need to manage paths across logical devices. With PowerPath, configurations of paths and policies for an individual device can be changed dynamically, taking effect immediately, without any disruption to the applications.

PowerPath provides the following features and benefits:

- ◆ Multiple paths, for higher availability and performance — PowerPath supports multiple paths between a logical device and a host bus adapter (HBA, a device through which a host can issue I/O requests). Having multiple paths enables the host to access a logical device even if a specific path is unavailable. Also, multiple paths can share the I/O workload to a given logical device.
- ◆ Dynamic multipath load balancing — Through continuous I/O balancing, PowerPath improves a host's ability to manage heavy I/O loads. PowerPath dynamically tunes paths for performance as workloads change, eliminating the need for repeated static reconfigurations.
- ◆ Proactive I/O path testing and automatic path recovery — PowerPath periodically tests failed paths to determine if they are available. A path is restored automatically when available, and PowerPath resumes sending I/O to it. PowerPath also periodically tests available but unused paths, to ensure they are operational.
- ◆ Automatic path failover — PowerPath automatically redirects data from a failed I/O path to an alternate path. This eliminates application downtime; failovers are transparent and non-disruptive to applications.
- ◆ Enhanced high availability cluster support — PowerPath is particularly beneficial in cluster environments, as it can prevent interruptions to operations and costly downtime. PowerPath's path failover capability avoids node failover, maintaining uninterrupted application support on the active node in the event of a path disconnect (as long as another path is available).
- ◆ Consistent split — PowerPath allows users to perform TimeFinder consistent splits by suspending device writes at the host level for a fraction of a second while the foreground split occurs. PowerPath software provides suspend-and-resume capability that avoids inconsistencies and restart problems that can occur if a database-related BCV is split without first quiescing the database.
- ◆ Consistency Groups — Consistency Groups are a composite group of Symmetrix devices specially configured to act in unison to maintain the integrity of a database distributed across multiple SRDF units controlled by an open systems host computer.

2.9 EMC Replication Manager

EMC Replication Manager is an EMC software application that dramatically simplifies the management and use of disk-based replications to improve the availability of user's mission-critical data and rapid recovery of that data in case of corruption.

Replication Manager helps users manage replicas as if they were tape cartridges in a tape library unit. Replicas may be scheduled or created on demand, with predefined expiration periods and automatic mounting to alternate hosts for backups or scripted processing. Individual users with different levels of access ensure system and replica integrity. In addition to these features, Replication Manager is fully integrated with many critical applications such as DB2 LUW, Oracle, and Microsoft Exchange.

Replication Manager makes it easy to create point-in-time, disk-based replicas of applications, file systems, or logical volumes residing on existing storage arrays. It can create replicas of information stored in the following environments:

- ◆ Oracle databases
- ◆ DB2 LUW databases
- ◆ Microsoft SQL Server databases
- ◆ Microsoft Exchange databases
- ◆ UNIX file systems
- ◆ Windows file systems

The software utilizes a Java-based client-server architecture. Replication Manager can:

- ◆ Create point-in-time replicas of production data in seconds.
- ◆ Facilitate quick, frequent, and non-destructive backups from replicas.
- ◆ Mount replicas to alternate hosts to facilitate offline processing (for example, decision-support services, integrity checking, and offline reporting).
- ◆ Restore deleted or damaged information quickly and easily from a disk replica.
- ◆ Set the retention period for replicas so that storage is made available automatically.

Replication Manager has a generic storage technology interface that allows it to connect and invoke replication methodologies available on:

- ◆ EMC Symmetrix arrays
- ◆ EMC CLARiiON arrays
- ◆ HP StorageWorks arrays

Replication Manager uses Symmetrix API (SYMAPI) Solutions Enabler software and interfaces to the storage array's native software to manipulate the supported disk arrays. Replication Manager automatically controls the complexities associated with creating, mounting, restoring, and expiring replicas of data. Replication Manager performs all of these tasks and offers a logical view of the production data and corresponding replicas. Replicas are managed and controlled with the easy-to-use Replication Manager console.

Chapter 3 Creating Microsoft SQL Server Database Clones

This chapter presents these topics:

3.1	Overview.....	3-2
3.2	Recoverable versus restartable copies of databases.....	3-3
3.3	Copying the database with Microsoft SQL Server shutdown.....	3-3
3.4	Copying the database using EMC Consistency technology	3-9
3.5	Copying the database using SQL Server Virtual Device Interface.....	3-14
3.6	Copying the database using Replication Manager.....	3-26
3.7	Transitioning disk copies to Microsoft SQL Server databases clones.....	3-27
3.8	Reinitializing the cloned environment.....	3-35
3.9	Choosing a database cloning methodology	3-35

This chapter describes the Microsoft SQL Server database cloning process using various EMC products. Determining which replication products to use depends on the customer's requirements and database environment. Products such as the TimeFinder Integration Module for SQL Server, the TimeFinder CLI tools themselves, and Replication Manager provide an easy method of copying Microsoft SQL Server databases in a single Symmetrix array.

All storage-based technologies used to create cloned databases depend on some form of restart or recovery processing. This methodology is used by many of the existing SQL Server solutions, including implementations such as Microsoft Cluster Service which behave as restart environments. The mechanisms used by EMC clone processing are not unique and utilize tried and tested methodologies.

A database cloning process typically includes some or all of the following steps, depending on the copying mechanism selected and the desired usage of the database clone:

- ◆ Preparing the array for replication
- ◆ Conditioning the source database
- ◆ Making a copy of the database volumes
- ◆ Resetting the source database
- ◆ Presenting the target database copy to a server
- ◆ Conditioning the target database copy

3.1 Overview

There are many choices when cloning databases with EMC array-based replication software. Each software product has differing characteristics that affect the final deployment. A thorough understanding of the options available leads to an optimal replication choice.

A Microsoft SQL Server database can be in one of three data states when it is being copied:

- ◆ Shutdown
- ◆ Processing normally
- ◆ Conditioned using SQL Server Virtual Device Interface (VDI)

Depending on the data state of the database at the time it is copied, the database copy may be restartable or recoverable. This chapter begins with a discussion of recoverable and restartable database clones. It then describes various approaches to data replication using EMC software products and how the replication techniques can be used in combination with the different database data states to facilitate the database cloning process. Following that, database clone usage considerations are discussed along with descriptions of the procedures used to deploy database clones.

3.2 Recoverable versus restartable copies of databases

The Symmetrix-based replication technologies described in this section can create two types of database copies, recoverable or restartable. A significant amount of confusion exists between these two types of database copies; a clear understanding of their differences is critical to ensure the appropriate application of each method when a cloned SQL Server environment is required.

3.2.1 Recoverable database copies

A recoverable database copy is one in which logs can be applied to the database data state and the database is rolled forward to a point in time after the database copy was created. A recoverable SQL Server database copy is intuitively easy for DBAs to understand since maintaining recoverable copies in the form of backups is an important DBA function. In the event of a failure of the production database, the ability to recover the database not only to the point in time when the last backup was taken, but also to roll forward subsequent transactions up to the point of failure, is a key feature of the SQL Server database.

In general, recoverable copies of SQL Server databases must utilize the **BACKUP DATABASE** Transact-SQL statement, utilize the Virtual Device Interface or integrate with the Volume Shadow Copy Service (VSS) subsystem.

3.2.2 Restartable database copies

If a disk mirror copy of a running SQL Server database is created using EMC consistency technology without utilizing a utility which implements VDI backup functionality, the copy is to be considered a DBMS restartable copy. This means that when the files representing the database are attached on the restartable copy, SQL Server will perform crash recovery. First, all transactions that were recorded as committed and written to the transaction log, but which may not have had corresponding data pages written to the data files, are rolled forward. This is the redo phase. Second, when the redo phase is complete, SQL Server enters the undo phase where it looks for database changes that were recorded (dirty page flushed by a lazy write for instance) but which were never actually committed by a transaction. These changes are rolled back or undone. The state attained is often referred to as a transactionally consistent point in time. It is essentially the same process that the RDBMS would undergo should the server have suffered an unanticipated interruption such as a power failure.

Roll-forward recovery using incremental transaction log backups to a point in time after the database copy was created is not supported on a Microsoft SQL Server restartable database copy.

3.3 Copying the database with Microsoft SQL Server shutdown

It is possible to take a copy of a Microsoft SQL Server database while the database is shutdown. Taking a copy after the database has been shut down normally will ensure a clean copy for streaming to tape or for fast startup of the cloned database. In addition, a cold copy of a database is in a known transactional data state which, for some

application requirements, is exceedingly important. Copies of running databases are in unknown transactional data states.

Note that a cold copy of a SQL Server database does not constitute a valid SQL Server backup, as no record of a backup event is ever made by the SQL Server instance. Subsequently, it is also not possible to process the files with functions such as the Transact SQL **RESTORE DATABASE** command set.

The primary method of creating cold copies of a Microsoft SQL Server database is through the use of EMC's local replication product, TimeFinder. TimeFinder is also used by Replication Manager to make database copies. Replication Manager facilitates the automation and management of database clones. Additionally, the EMC TimeFinder SQL Server Integration Module (TF/SIM) also facilitates the creation of copies based on TimeFinder functionality.

TimeFinder operations are implemented in three different forms, TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap. These were discussed in general terms in Chapter 2. Here, they are utilized in a database context.

3.3.1 Using TimeFinder/Mirror

TimeFinder/Mirror is an EMC Symmetrix array implementation that allows an additional hardware mirror to be attached to a source volume. The additional mirror is a specially designated volume in the Symmetrix configuration called a business continuance volume, or BCV. The BCV is synchronized to the source volume through a process referred to as an establish. While the BCV is established, it is "not ready" to all hosts to which it may be presented. At an appropriate time, the BCV can be split from the source volume to create a complete point-in-time copy of the source data that can be used for different purposes, including backup, decision support, and regression testing.

Groups of BCVs are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Mirror operations. If the database spans more than one Symmetrix, a composite group is used. Figure 3-1 on page 3-5 depicts the necessary steps to make a database copy of a cold Microsoft SQL Server database using TimeFinder/Mirror.

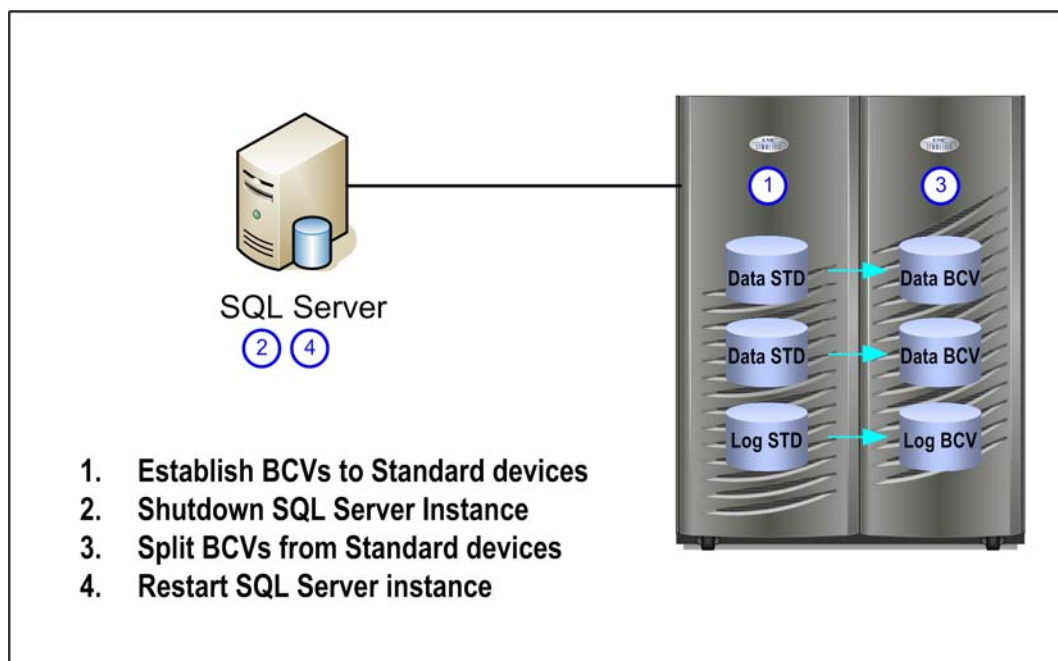


Figure 3-1 Copying a shutdown SQL Server database with TimeFinder/Mirror

1. Establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed.

```
symmir -g <device_group> establish -full -noprompt
```

Note that the first iteration of the establish needs to be a full synchronization. Subsequent iterations by default are incremental if the `-full` keyword is omitted. Once the command is issued, the array begins the synchronization process using only Symmetrix resources. Since this operation occurs independently from the host, the process must be interrogated to see when it completes. The command to interrogate the synchronization process is:

```
symmir -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete. Alternatively, synchronization can be verified using the following:

```
symmir -g <device_group> query
```

After the volumes are synchronized, the split command can be issued at any time.

2. Once BCV synchronization is complete the database needs to be brought down in order to make a copy of a cold database. SQL Server management tools may be used to take the database offline or shut down the instance.
3. When the database is deactivated, split the BCV mirrors using the following command:

```
symmir -g <device_group> split -noprompt
```

The split command takes a few seconds to process. The database copy on the BCVs is now ready for further processing.

4. The source database can now be activated and made available to users once again. Once again, use the SQL Server management tools to reinstate the database, or restart the SQL Server instance.

3.3.2 Using TimeFinder/Clone

TimeFinder/Clone is an EMC software product that copies data internally in the Symmetrix array. A TimeFinder/Clone session is created between a source data volume and a target volume. The target volume needs to be equal to or greater in size than the source volume. The source and target for TimeFinder/Clone sessions can be any hypervolumes in the Symmetrix configuration.

TimeFinder/Clone devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Clone operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix A.

Figure 3-2 depicts the necessary steps to make a copy of a cold SQL Server database onto BCV devices using TimeFinder/Clone.

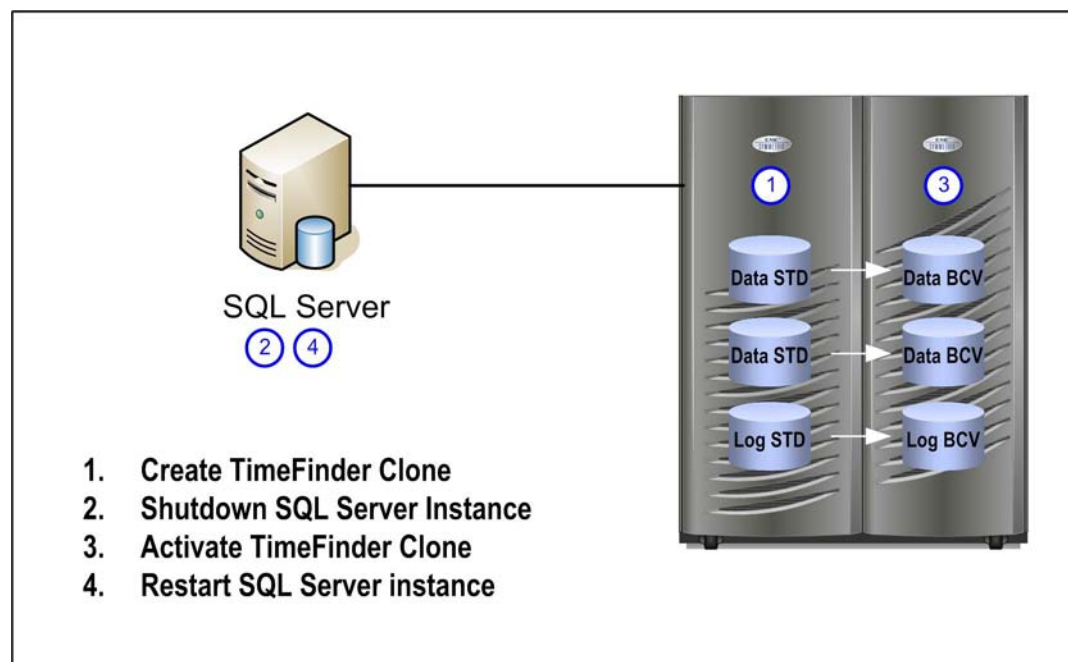


Figure 3-2 Copying a shutdown SQL Server database with TimeFinder/Clone

1. The first action is to create the TimeFinder/Clone pairs. The following command creates the TimeFinder/Clone pairings and protection bitmaps. No data is copied or moved at this time:

```
symclone -g <device_group> create -noprompt
```

Unlike TimeFinder/Mirror, the TimeFinder/Clone relationship is created and activated when it is needed. No prior synchronization of data is necessary. After the TimeFinder/Clone session is created, it can be activated consistently.

2. Once the create command has completed, the database needs to be shut down to make a cold disk copy of the database. SQL Server management tools may be used to take the database offline or shut down the instance.
3. With the database down, the TimeFinder/Clone can now be activated:

```
symclone -g <device_group> activate -noprompt
```

After the activate, the database copy provided by TimeFinder/Clone is immediately available for further processing even though the copying of data may not have completed.

4. The source database can now be activated and made available to users once again.

Databases copied using TimeFinder/Clone are subject to Copy on First Write (COFW) and Copy on Access (COA) penalties. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the target volume, it must first be copied to the target volume before the write from the host is acknowledged. Copy on Access means that if a track on a TimeFinder/Clone volume is accessed before it has been copied, it must first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems.

3.3.3 Using TimeFinder/Snap

TimeFinder/Snap enables users to create complete copies of their data while consuming only a fraction of the disk space required by the original copy.

TimeFinder/Snap is an EMC software product that maintains space-saving pointer-based copies of disk volumes using virtual devices (VDEVs) and save devices (SAVDEVs). The VDEVs contain pointers either to the source data (when it is unchanged) or to the SAVDEVs (when the data has been changed).

TimeFinder/Snap devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Snap operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

Figure 3-3 depicts the necessary steps to make a copy of a cold SQL Server database using TimeFinder/Snap.

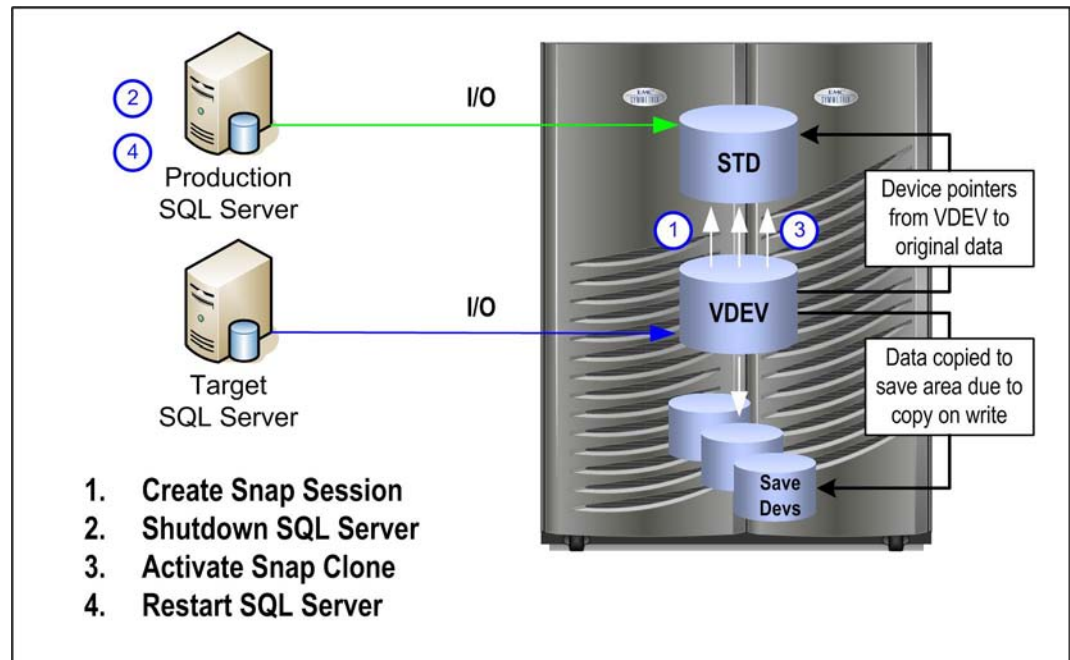


Figure 3-3 Copying a shutdown SQL Server database with TimeFinder/Snap

1. The first action is to create the TimeFinder/Snap pairs. The following command creates the TimeFinder/Snap pairings and protection bitmaps. No data is copied or moved at this time:

```
symsnap -g <device_group> create -noprompt
```

2. Once the create operation has completed, the database needs to be shut down to make a cold TimeFinder/Snap of the DBMS. SQL Server management tools may be used to take the database offline or shut down the instance, whichever is most applicable.

3. With the database down, the TimeFinder/Snap copy can now be activated:

```
symsnap -g <device_group> activate -noprompt
```

After the activate, the pointer-based database copy on the VDEVs is available for further processing.

4. The source database can now be restarted.

Databases copied using TimeFinder/Snap are subject to a Copy on First Write (COFW) penalty while the snap is activated. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the snap save area, it must first be copied to the save area before the write from the host is acknowledged.

3.4 Copying the database using EMC Consistency technology

The replication of a running database system involves a database copying technique that is employed while the database is servicing applications and users. The database copying technique uses EMC Consistency technology in the form of a consistency group (CG) combined with an appropriate data copy process like TimeFinder/Mirror and TimeFinder/Clone. TimeFinder/CG allows for the running database copy to be created in an instant through use of the `–consistent` keyword on the `split` or `activate` commands. The image created in this way is in a dependent-write consistent data state and can be utilized as a restartable copy of the database.

Databases management systems enforce a principle of dependent-write I/O. That is, no dependent-write will be issued until the predecessor write that it is dependent on has completed. This type of programming discipline is used to coordinate database and log updates within a database management system and allows those systems to be restartable in event of a power failure. Dependent-write consistent data states are created when database management systems are exposed to power failures. Using EMC Consistency technology options during the database cloning process also creates a database copy that has a dependent-write consistent data state. See Chapter 2 for more discussion on EMC Consistency technology.

Microsoft SQL Server databases can be copied while they are running and processing transactions. The following sections describe how to copy a running Microsoft SQL Server database using TimeFinder technology.

3.4.1 Using TimeFinder/Mirror

TimeFinder/Mirror is an EMC software product that allows an additional hardware mirror to be attached to a source volume. The additional mirror is a specially designated volume in the Symmetrix configuration called a business continuance volume, or BCV. The BCV is synchronized to the source volume through a process called an establish. While the BCV is established, it is not ready to all hosts. At an appropriate time, the BCV can be split from the source volume to create a complete point-in-time copy of the source data that can be used for different purposes, including backup, decision support, and regression testing.

Groups of BCVs are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Mirror operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

Figure 3-4 depicts the necessary steps to make a database copy of a running SQL Server database using TimeFinder/Mirror:

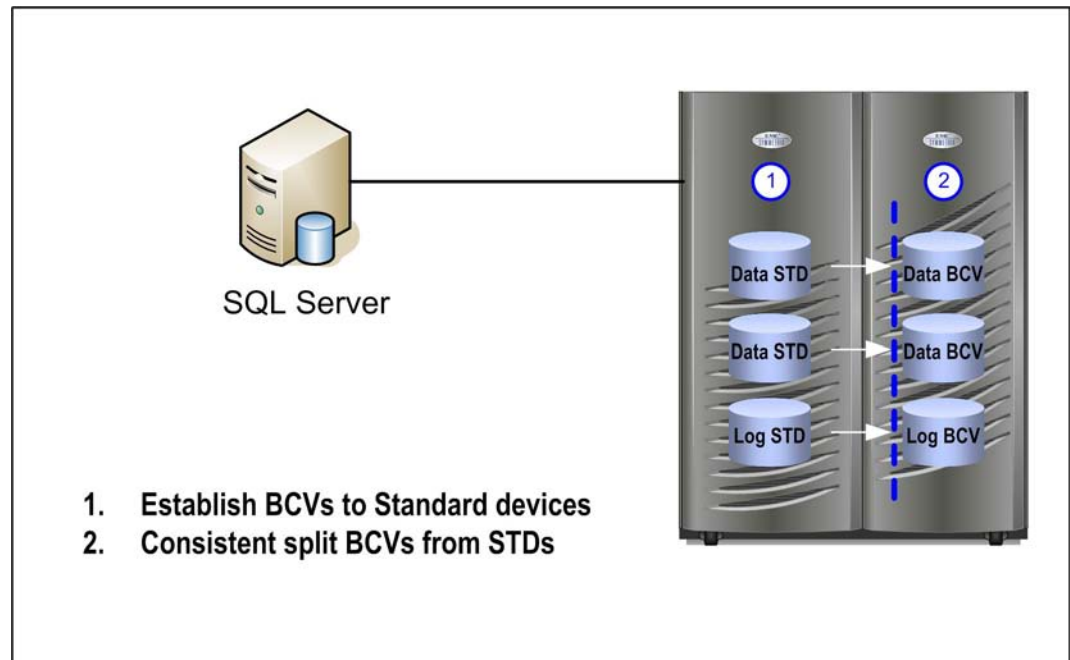


Figure 3-4 Copying a running SQL Server database with TimeFinder/Mirror

1. The first action is to establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed:

```
symmir -g <device_group> establish -full -noprompt
```

Note that the first iteration of the establish needs to be a full synchronization. Subsequent iterations are incremental and do not need the `-full` keyword. Once the command is issued, the array begins the synchronization process using only Symmetrix resources. Since this operation occurs independently from the host, the process must be interrogated to see when it completes. The command to interrogate the synchronization process is:

```
symmir -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete. Alternatively, synchronization can be verified using the following:

```
symmir -g <device_group> query
```

2. When the volumes are synchronized the split command can be issued.

```
symmir -g <device_group> split -consistent -noprompt
```

The `-consistent` keyword tells the Symmetrix to use Engineuity Consistency Assist (ECA) to momentarily suspend writes to the disks while the split is being processed.

The effect of this is to create a point-in-time copy of the database on the BCVs. It is similar to the image created when there is a power outage that causes the server to crash. This image is a restartable copy—a term which is defined previously. The database copy on the BCVs is then available for further processing.

Since there was no specific coordination between the database state and the execution of the consistent split, the copy is taken independent of the database activity. In this way, EMC Consistency technology can be used to make point-in-time copies of multiple systems atomically, resulting in a consistent point in time with respect to all applications and databases included in the consistent split.

3.4.2 Using TimeFinder/Clone

TimeFinder/Clone is an EMC software product that copies data internally in the Symmetrix array. A TimeFinder/Clone session is created between a source data volume and a target volume. The target volume needs to be equal to or greater in size than the source volume. The source and target for TimeFinder/Clone sessions can be any hypervolumes in the Symmetrix configuration.

TimeFinder/Clone devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Clone operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

Figure 3-5 depicts the necessary steps to make a copy of a running SQL Server database onto BCV devices using TimeFinder/Clone:

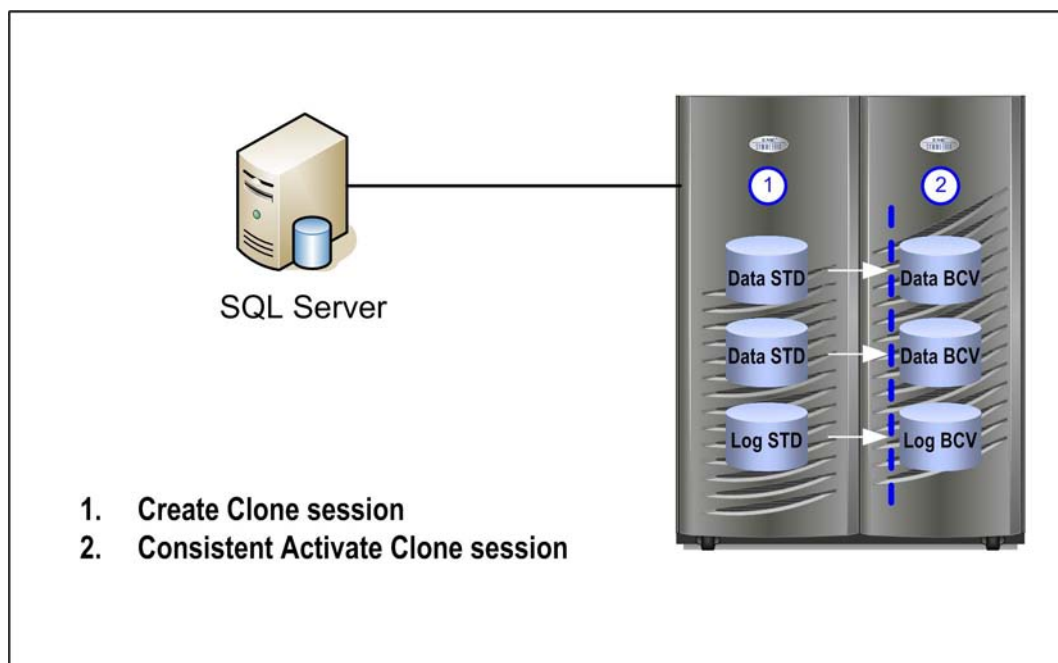


Figure 3-5 Copying a running SQL Server database with TimeFinder/Clone

1. The first action is to create the TimeFinder/Clone pairs. The following command creates the TimeFinder/Clone pairings and protection bitmaps. No data is copied or moved at this time:

```
symclone -g <device_group> create -noprompt
```

Unlike TimeFinder/Mirror, the TimeFinder/Clone relationship is created and activated when it is needed. No prior copying of data is necessary.

2. After the TimeFinder/Clone relationship is created it can be activated consistently.

```
symclone -g <device_group> activate -consistent -noprompt
```

The `-consistent` keyword tells the Symmetrix to use Engineuity Consistency Assist (ECA) to momentarily suspend writes to the source disks while the TimeFinder/Clone is being activated. The effect of this is to create a point-in-time copy of the database on the target volumes. It is a copy similar in state to that created when there is a power outage resulting in a server crash. This copy is a restartable copy—a term which is defined previously. After the activate command, the database copy on the TimeFinder/Clone devices is available for further processing.

Since there was no specific coordination between the database state and the execution of the consistent split, the copy is taken independent of the database activity. In this way, EMC Consistency technology can be used to make point-in-time copies of multiple systems atomically, resulting in a consistent point in time with respect to all applications and databases included in the consistent split.

Databases copied using TimeFinder/Clone are subject to Copy on First Write (COFW) and Copy on Access (COA) penalties. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the target volume, it must first be copied to the target volume.

3.4.3 Using TimeFinder/Snap

TimeFinder/Snap enables users to create complete copies of their data while consuming only a fraction of the disk space required by the original copy.

TimeFinder/Snap is an EMC software product that maintains space-saving, pointer-based copies of disk volumes using virtual devices (VDEVs) and save devices (SAVDEVs). The VDEVs contain pointers either to the source data (when it is unchanged) or to the SAVDEVs (when the data has been changed).

TimeFinder/Snap devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Snap operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

Figure 3-6 on page 3-13 depicts the necessary steps to make a copy of a running Microsoft SQL Server database using TimeFinder/Snap.

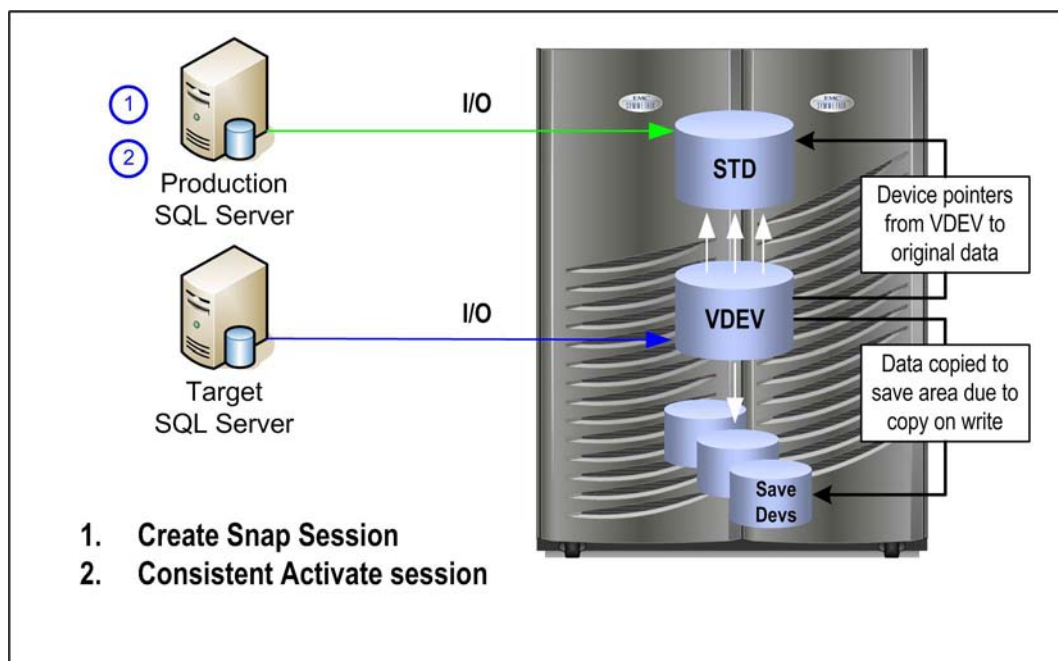


Figure 3-6 Copying a running SQL Server database with TF/Snap

1. The first action is to create the TimeFinder/Snap pairs. The following command creates the TimeFinder/Snap pairings and protection bitmaps. No data is copied or moved at this time:

```
symsnap -g <device_group> create -noprompt
```

After the TimeFinder/Snap is created, all the pointers from the VDEVs point at the source volumes. No data has been copied at this point. The snap can be activated consistently using the consistent activate command.

2. Once the create operation has completed, the activate command can be executed with the `-consistent` option. Execute the following command to perform the consistent snap:

```
symsnap -g <device_group> activate -consistent -noprompt
```

The `-consistent` keyword tells the Symmetrix to use Engineuity Consistency Assist (ECA) to momentarily suspend writes to the disks while the activate command is being processed. The effect of this is to create a point-in-time copy of the database on the VDEVs. It is similar to the state created when there is a power outage that causes the server to crash. This image is a restartable copy—a term which is defined previously. The database copy on the VDEVs is available for further processing.

Since there was no specific coordination between the database state and the execution of the consistent split, the copy is taken independent of the database activity. In this way, EMC Consistency technology can be used to make point-in-time copies of multiple systems atomically, resulting in a consistent point in time with respect to all applications and databases included in the consistent split.

Databases copied using TimeFinder/Snap are subject to a Copy on First Write (COFW) penalty while the snap is activated. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the snap save area, it must first be copied to the snap save area before the write from the host is acknowledged.

3.5 Copying the database using SQL Server Virtual Device Interface

Microsoft SQL Server 2000 and later support a programmatic interface, which provides the capability to use split mirror disk technology while a SQL Server database is online and create a recoverable database on the copied devices. During this process, the database is fully available for reads. Write operations are suspended during the Virtual Device Interface (VDI) split operation. When a transaction issues a commit, it is suspended until the VDI operation has been issued and the subsequent disk mirror is split.

In contrast to the executions of the previous section, there is no need to specify the `-consistent` option to any TimeFinder command, as SQL Server through the VDI operations, ensures database consistency is maintained.

The copy made in this manner is considered a valid full database backup of the target database(s). Microsoft SQL Server will indicate that a full valid backup has been completed in the system tables, and associated database files.

Because the VDI is a programmatic interface, it cannot be called directly by a user. Custom applications are typically created by storage vendors such as EMC which interface between the VDI and the storage array technology.

EMC provides a number of products that support this mechanism for creating valid backups for EMC Symmetrix. These products include the graphical Replication Manager and the command line interface-based TimeFinder SQL Server Integration Module (TF/SIM). Additionally, the Storage Resource Management (SRM) suite of tools provides the SYMIOCTL command line utility.

In the following examples, TF/SIM and SYMIOCTL are used to demonstrate the various forms of TimeFinder functionality. These examples assume local execution of the TF/SIM product on the production server. Remote execution of TF/SIM is beyond the scope of this chapter.

3.5.1 Using TimeFinder/Mirror

TimeFinder/Mirror is an EMC software product that allows an additional hardware mirror to be attached to a source volume. The additional mirror is a specially designated volume in the Symmetrix configuration called a business continuance volume, or BCV. The BCV is synchronized to the source volume through a process called an establish. While the BCV is established, it is not ready to all hosts. At an appropriate time, the BCV can be split from the source volume to create a complete point-in-time copy of the source data that can be used for multiple different purposes, including backup, decision support, and regression testing.

In general, those EMC products that utilize the SQL Server VDI also implement EMC's Storage Resource Management (SRM), which dynamically maps database files to array volumes. Thus, the operations of these tools become independent of SYMCLI device groups, which are used by administrators to monitor the volume states and relationships.

3.5.1.1 Executing TF/SIM

The process detailed in Figure 3-7 demonstrates the necessary steps required to create a split-mirror database backup of a Microsoft SQL Server database using TF/SIM to manager TimeFinder/Mirror operations:

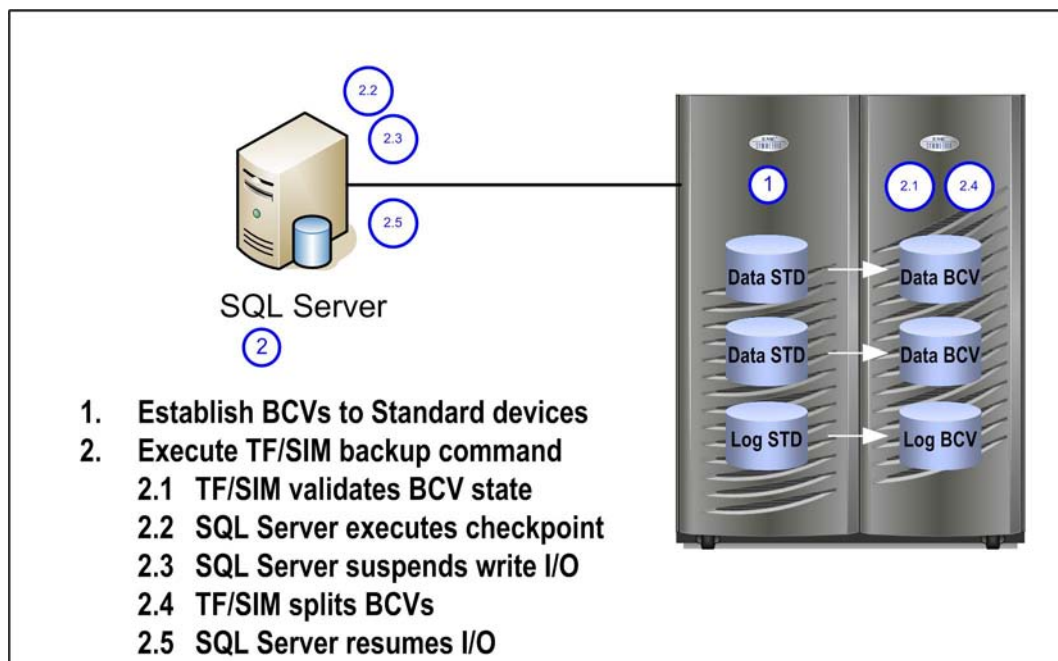


Figure 3-7 Creating a TimeFinder/Mirror VDI backup of a SQL Server database

1. The first action is to establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed.

```
symmir -g <device_group> establish -full -noprompt
```

Note that the first iteration of the establish needs to be a full synchronization. Subsequent iterations are incremental and do not need the `-full` keyword. Once the command is issued, the array begins the synchronization process using only Symmetrix resources. Since this is asynchronous to the host, the process must be interrogated to see when it is finished. The command to interrogate the synchronization process is:

```
symmir -g <device_group> verify
```

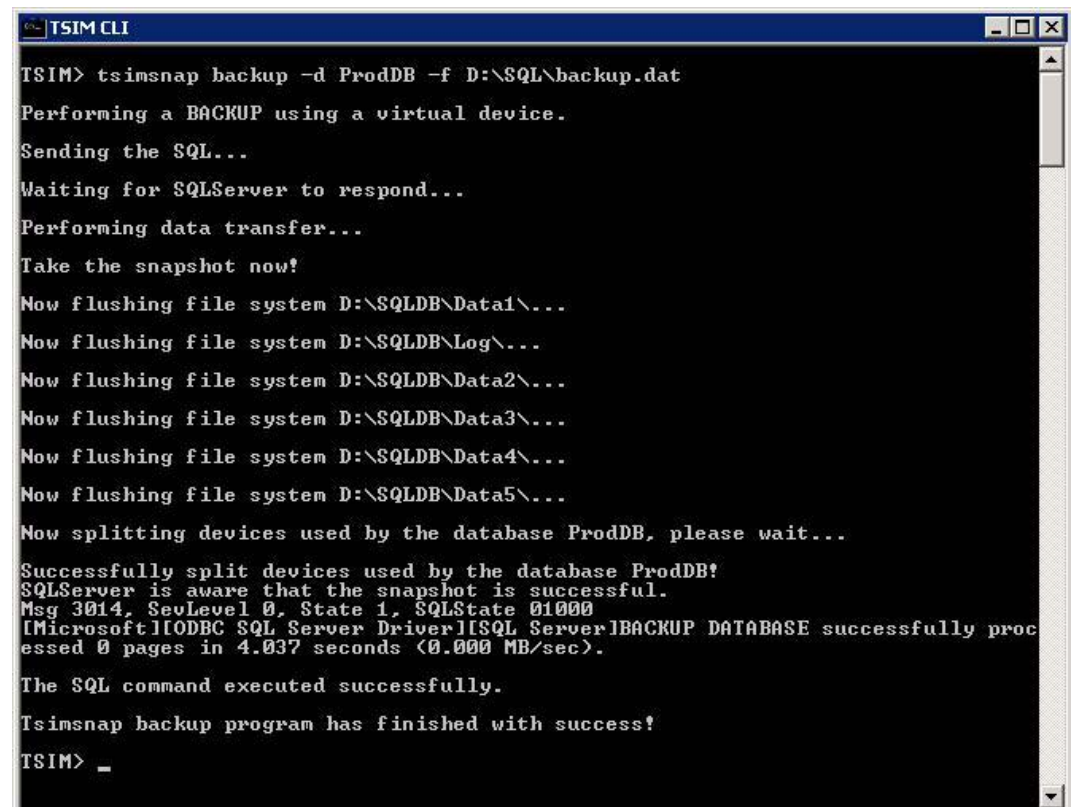
This command will return a 0 return code when the synchronization operation is complete.

2. When the volumes are synchronized, the TF/SIM backup operation may be executed.

TF/SIM does not actually use the device group to manage its internal operations, rather it examines relationships, which have been created between source and disk mirrors by administrators as a result of their use of various SYMCLI device group operations.

`tsimsnap backup -d <database> -f <location of metafile>`

In the example in Figure 3-8, the target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file.



```

TSIM CLI
TSIM> tsimsnap backup -d ProdDB -f D:\SQL\backup.dat
Performing a BACKUP using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Take the snapshot now!
Now flushing file system D:\SQLDB\Data1\...
Now flushing file system D:\SQLDB\Log\...
Now flushing file system D:\SQLDB\Data2\...
Now flushing file system D:\SQLDB\Data3\...
Now flushing file system D:\SQLDB\Data4\...
Now flushing file system D:\SQLDB\Data5\...
Now splitting devices used by the database ProdDB, please wait...
Successfully split devices used by the database ProdDB!
SQLServer is aware that the snapshot is successful.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]BACKUP DATABASE successfully proc
essed 0 pages in 4.037 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap backup program has finished with success!
TSIM> _

```

Figure 3-8 Sample TimeFinder/SIM backup execution with TimeFinder/Mirror

3.5.1.2 Executing SYMIOCTL

Similar to the TF/SIM execution, it is possible to utilize the SYMIOCTL command line utility to execute VDI processing for the SQL Server database. Unlike TF/SIM, however, all processing of the device group and state of the mirror devices within the group must be managed through command line operations. Also, there is no default ability to flush the filesystem buffers (note that in the preceding TF/SIM examples, TF/SIM ensures that file system buffers are also flushed). To implement this functionality with SYMIOCTL, it is necessary to use the Symmetrix Integration Utility (SIU) to perform the flush operation. The flush operation ensures that other file system-level operations are successfully written to the disk.

The execution of the SYMIOCTL proceeds in the following manner:

1. The first action is to establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed.

```
symmir -g <device_group> establish -full -noprompt
```

Note that the first iteration of the establish needs to be a full synchronization. Subsequent iterations are incremental and do not need the `-full` keyword. Once the command is issued, the array begins the synchronization process using only Symmetrix resources. Since this is asynchronous to the host, the process must be interrogated to see when it is finished. The command to interrogate the synchronization process is:

```
symmir -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete.

2. When the volumes are synchronized, the SYMIOCTL backup operation may be executed.

```
symioctl -type SQLServer begin snapshot <database>  
SAVEFILE <location of metafile> -noprompt
```

In this example, the SAVEFILE command line option specifies the location where SYMIOCTL will store the VDI metadata file.

3. Flush all database locations, either drive letters or mount points, in use by using SIU. In this example, the locations are mount points:

```
symntctl flush -path <mount point for volume>
```

Execute for each and every database file and log location.

4. Split the devices

```
symmir -g <device_group> split -instant -noprompt
```

In this case, the `-instant` option to the `symmir` command is used to facilitate a much faster execution of the split operation. Using this command, the Symmetrix array will maintain the logical state of the devices, and return control immediately to the command line.

5. Finalize the SYMIOCTL backup command and thaw I/O operations on the SQL Server database.

```
symioctl -type SQLServer end snapshot <database> -noprompt
```

It is the responsibility of the administrator to ensure that the SYMIOCTL end snapshot command is executed within the script—there is no automatic termination of the command. Leaving the database in snapshot state will suspend all user connections and may cause SQL Server to take the database offline. Any user-created process or script must cater for abnormal termination of the process or script itself, and ensure that the end snapshot is executed as a cleanup operation.

```

C:\>symioctl -type SQLServer begin snapshot ProdDB SAVEFILE D:\SQL\backup.dat -n
oprompt

C:\>symntctl flush -path D:\SQLDB\Data1
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data2
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data3
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data4
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data5
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Log
    Successfully flushed file system cache.

C:\>symmir -g sql_prod split -instant -noprompt
'Split' operation execution is in progress for
device group 'sql_prod'. Please wait...

'Split' operation successfully executed for device group
'sql_prod'.

C:\>symioctl -type SQLServer end snapshot ProdDB -noprompt
C:\>_
  
```

Figure 3-9 Sample SYMIOCTL execution with TimeFinder/Mirror

3.5.2 Using TimeFinder/Clone

TimeFinder/Clone is an EMC software product that copies data internally in the Symmetrix array. A TimeFinder/Clone session is created between a source data volume and a target volume. The target volume needs to be equal to or greater in size than the source volume. The source and target for TimeFinder/Clone sessions can be any hypervolumes in the Symmetrix configuration.

TimeFinder/Clone devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Clone operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

3.5.2.1 Executing TF/SIM

The TimeFinder SQL Server Integration Module only provides support for the Symmetrix *Clone Emulation* mode to facilitate backup operations for a SQL Server database. Thus, all TimeFinder/Mirror operations are mapped transparently to TimeFinder/Clone operations. TF/SIM does require that for local operation, the source and target volumes are preestablished. Therefore, it is still necessary to establish the devices. In the case of Clone Emulation mode, the symmir establish command will be mapped to the appropriate symclone operation.

Figure 3-10 depicts the necessary steps to make a VDI backup of a Microsoft SQL Server database onto BCV devices using TimeFinder/Clone.

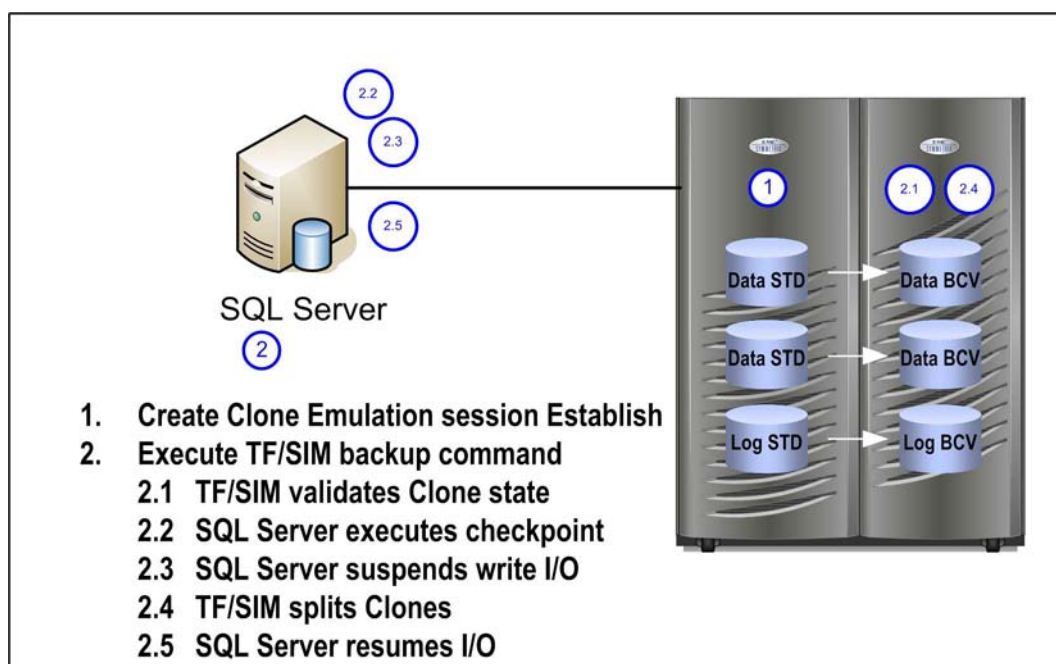


Figure 3-10 Creating a VDI backup of a SQL Server database with TimeFinder/Clone

1. The first action is to create and synchronize the TimeFinder/Clone pairs. The following command creates the TimeFinder/Clone pairings and protection bitmaps. Since this will be under a Clone Emulation mode, the appropriate environment variable must be set before executing the symmir command:

```
set SYMCLI_CLONE_EMULATION=ENABLED
```

```
symmir -g <device_group> establish -full -noprompt
```

TF/SIM requires that the clone is fully synchronized with the source volumes. As this is Clone Emulation mode, the standard symmir commands should be used with the environment variable set. The command to interrogate the process is:

```
symmir -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete.

2. When the volumes are synchronized, the TF/SIM backup operation may be executed:

```
tsimsnap backup -d <database> -f <location of metadata>
```

In this example, the target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file.

Databases copied using TimeFinder/Clone are typically subject to Copy on First Write (COFW) and Copy on Access (COA) penalties. However, in this instance, the clone devices are fully established in the first instance, and the clone becomes a fully independent image at that point. Subsequently, no Copy on First Write penalties will be suffered in this configuration.

The Copy on First Write penalty means that the first time a track is written to the source volume and it has not yet been copied to the target volume, it must first be copied to the target volume before the write from the host is acknowledged. Subsequent writes to tracks, which have already been copied, do not suffer from the penalty. Copy on Access means that if a track on a target volume is accessed before it has been copied, it must first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems.

3.5.2.2 Executing SYMIOCTL

Similar to the TF/SIM execution, it is possible to utilize the SYMIOCTL command line utility to execute VDI processing for the SQL Server database. Unlike TF/SIM, however, all processing of the device group and state of the mirror devices within the group must be managed through command line operations. Also, there is no default ability to flush the file system buffers (note that in the preceding TF/SIM examples, TF/SIM ensures that file system buffers are also flushed). To implement this functionality with SYMIOCTL, it is necessary to use the Symmetrix Integration Utility (SIU) to perform the flush operation. The flush operation ensures that other file system-level operations are successfully written to the disk. An example of the execution of this process is shown in Figure 3-11 on page 3-22.

Therefore, the execution of the SYMIOCTL process executes in the following manner:

1. The first action is to establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed.

```
symclone -g <device_group> create -noprompt
```

2. Once the clone session is created, the SYMIOCTL backup operation may be executed.

```
symioctl -type SQLServer begin snapshot <database>
SAVEFILE <location of metafile> -noprompt
```

In this example, the SAVEFILE command line option specifies the location where SYMIOCTL will store the VDI metadata file.

3. Flush all database locations, either drive letters or mount points, in use, by using SIU. In this example, the locations are mount points.

```
symntctl flush -path <mount point for volume>
```

Execute for each and every database file and log location.

4. Activate the clone session

```
symclone -g <device_group> activate -noprompt
```

5. Finalize the SYMIOCTL backup command and thaw I/O operations on the SQL Server database.

```
symioctl -type SQLServer end snapshot <database> -noprompt
```

It is the responsibility of the administrator to ensure that the SYMIOCTL end snapshot command is executed within the script—there is no automatic termination of the command. Leaving the database in snapshot state will suspend all user connections and may cause SQL Server to take the database offline. Any user-created process or script must cater for abnormal termination of the process or script itself, and ensure that the end snapshot is executed as a cleanup operation.

Databases copied using TimeFinder/Clone are subject to Copy on First Write (COFW) and Copy on Access (COA) penalties. The Copy on First Write penalty means that the first time a track is written to the source volume and it has not yet been copied to the target volume, it must first be copied to the target volume before the write from the host is acknowledged. Subsequent writes to tracks, which have already been copied, do not suffer from the penalty. Copy on Access means that if a track on a target volume is accessed before it has been copied, it must first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems.

```

C:\>symclone -g sql_prod create -noprompt
'Create' operation execution is in progress for
device group 'sql_prod'. Please wait...
'Create' operation successfully executed for device group
'sql_prod'.

C:\>symioctl -type SQLServer begin snapshot ProdDB SAVEFILE D:\SQL\backup.dat -n
oprompt

C:\>symntctl flush -path D:\SQLDB\Data1
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data2
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data3
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data4
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data5
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Log
    Successfully flushed file system cache.
C:\>symclone -g sql_prod activate -noprompt
'Activate' operation execution is in progress for
device group 'sql_prod'. Please wait...
'Activate' operation successfully executed for device group
'sql_prod'.

C:\>symioctl -type SQLServer end snapshot ProdDB -noprompt
C:\>

```

Figure 3-11 Sample SYMIOCTL execution with TF/Clone

3.5.3 Using TimeFinder/Snap

TimeFinder/Snap enables users to create complete copies of their data while consuming only a fraction of the disk space required by the original copy.

TimeFinder/Snap is an EMC software product that maintains space-saving pointer-based copies of disk volumes using virtual devices (VDEVs) and save devices (SAVDEVs). The VDEVs contain pointers either to the source data (when it is unchanged) or to the SAVDEVs (when the data has been changed).

TimeFinder/Snap devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Snap operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

3.5.3.1 Executing TF/SIM

Figure 3-12 depicts the necessary steps to make a VDI backup of a SQL Server database using TimeFinder/Snap:

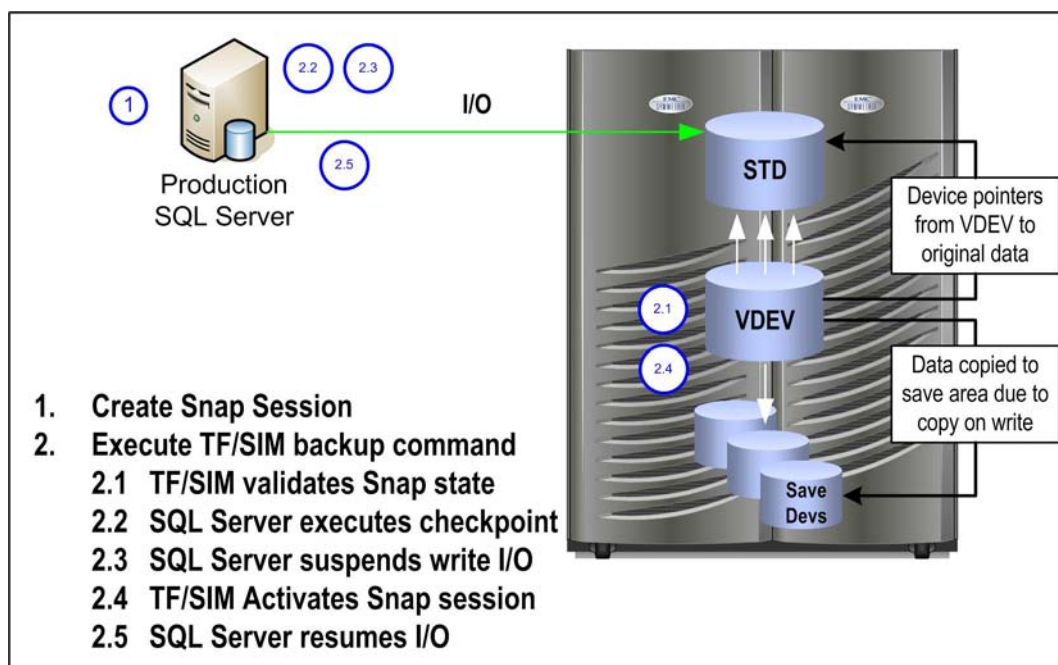


Figure 3-12 Creating a VDI backup of a SQL Server database with TimeFinder/Snap

1. The first action is to create the TimeFinder/Snap pairs. The following command creates the TimeFinder/Snap pairings and protection bitmaps. No data is copied or moved at this time:

```
symsnap -g <device_group> create -noprompt
```

Unlike TimeFinder/Mirror, the snap relationship is created and activated when it is needed. No prior copying of data is necessary. The create operation establishes the relationship between the standard devices and the virtual devices (VDEVs) and it also creates the protection metadata.

2. After the snap is created, the TF/SIM backup operation may be executed:

```
tsimsnap backup -d <database> -f <location of metafile> -snap
```

In this example, the target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file. An example of the execution of this process is shown in Figure 3-13 on page 3-24.

```

TSIM CLI
TSIM> tsimsnap backup -d Proddb -f D:\SQL\backup.dat -snap
Performing a BACKUP using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Take the snapshot now!
Now flushing file system D:\SQLDB\Data1\...
Now flushing file system D:\SQLDB\Log\...
Now flushing file system D:\SQLDB\Data2\...
Now flushing file system D:\SQLDB\Data3\...
Now flushing file system D:\SQLDB\Data4\...
Now flushing file system D:\SQLDB\Data5\...
Now activating snap devices used by the database Proddb, please wait...
Successfully activating snap devices used by the database Proddb!
SQLServer is aware that the snapshot is successful.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]BACKUP DATABASE successfully proc
essed 0 pages in 3.848 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap backup program has finished with success!
TSIM> _

```

Figure 3-13 Sample TF/SIM backup execution with TF/Snap

Databases copied using TimeFinder/Snap are subject to a Copy on First Write (COFW) penalty while the snap is activated. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the snap save area, it must first be copied to the save area before the write from the host is acknowledged.

3.5.3.2 Executing SYMIOCTL

Similar to the TF/SIM execution, it is possible to utilize the SYMIOCTL command line utility to execute VDI processing for the SQL Server database. Unlike TF/SIM, however, all processing of the device group and state of the mirror devices within the group must be managed through command line operations. Also, there is no default ability to flush the file system buffers (note that in the preceding TF/SIM examples, TF/SIM ensures that file system buffers are also flushed). To implement this functionality with SYMIOCTL, it is necessary to use the Symmetrix Integration Utility (SIU) to perform the flush operation. The flush operation ensures that other file system-level operations are successfully written to the disk.

To implement the SYMIOCTL process, the following steps should be executed:

1. The first action is to create the TimeFinder/Snap pairs. The following command creates the TimeFinder/Snap pairings and protection bitmaps. No data is copied or moved at this time:

```
symsnap -g <device_group> create -noprompt
```

2. Once the Snap session is created, the SYMIOCTL backup operation may be executed.

```
symioctl -type SQLServer begin snapshot <database>
SAVEFILE <location of metafile> -noprompt
```

In this example, the SAVEFILE command line option specifies the location where SYMIOCTL will store the VDI metadata file.

3. Flush all database locations, either drive letters or mount points, in use by using SIU. In this example, the locations are mount points.

```
symntctl flush -path <mount point for volume>
```

Execute for each and every database file and log location.

4. Activate the Snap session.

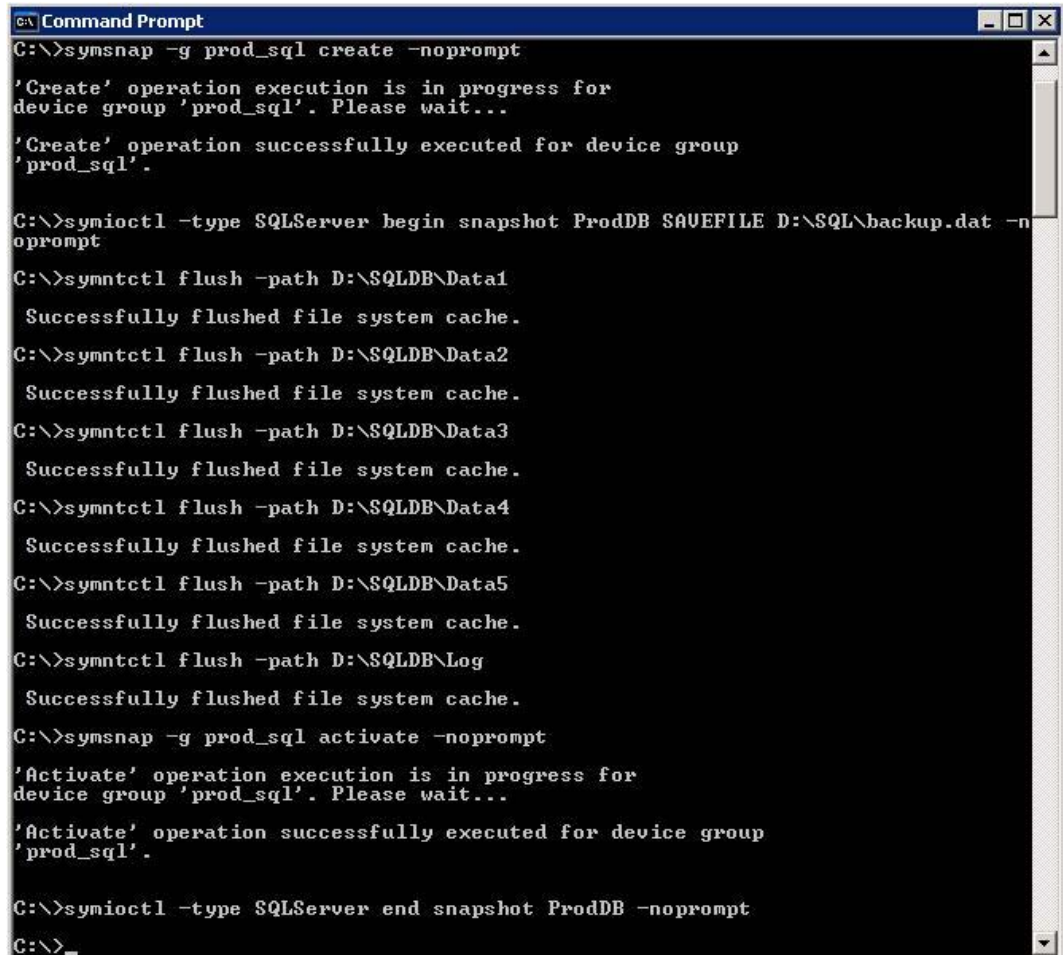
```
symsnap -g <device_group> activate
```

5. Finalize the SYMIOCTL backup command and thaw I/O operations on the SQL Server database.

```
symioctl -type SQLServer end snapshot <database> -noprompt
```

It is the responsibility of the administrator to ensure that the SYMIOCTL end snapshot command is executed within the script—there is no automatic termination of the command. Leaving the database in snapshot state will suspend all user connections and may cause SQL Server to take the database offline. Any user-created process or script must cater for abnormal termination of the process or script itself, and ensure that the end snapshot is executed as a cleanup operation.

Databases copied using TimeFinder/Snap are subject to a Copy on First Write (COFW) penalty while the snap is activated. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the snap save area, it must first be copied to the save area before the write from the host is acknowledged. An example of the execution of this process is shown in Figure 3-14 on page 3-26.



```

C:\>symsnap -g prod_sql create -noprompt
'Create' operation execution is in progress for
device group 'prod_sql'. Please wait...

'Create' operation successfully executed for device group
'prod_sql'.

C:\>symioctl -type SQLServer begin snapshot ProdDB SAVEFILE D:\SQL\backup.dat -n
oprompt

C:\>symntctl flush -path D:\SQLDB\Data1
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data2
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data3
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data4
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Data5
    Successfully flushed file system cache.
C:\>symntctl flush -path D:\SQLDB\Log
    Successfully flushed file system cache.
C:\>symsnap -g prod_sql activate -noprompt
'Activate' operation execution is in progress for
device group 'prod_sql'. Please wait...

'Activate' operation successfully executed for device group
'prod_sql'.

C:\>symioctl -type SQLServer end snapshot ProdDB -noprompt
C:\>

```

Figure 3-14 Sample SYMIOCTL execution with TF/SNAP

3.6 Copying the database using Replication Manager

EMC Replication Manager (RM) can be used to manage and control the TimeFinder copies of a SQL Server database. The RM product has a GUI and command line and provides the capability to:

- ◆ Auto-discover the standard volumes holding the database
- ◆ Identify the path name for all data and transaction log file locations

Using this information, RM can set up TimeFinder Groups with BCVs or VDEVs, schedule TimeFinder operations and manage the creation of database copies, expiring older versions as needed.

Figure 3-15 on page 3-27 demonstrates the steps performed by Replication Manager using TimeFinder/Mirror to create a database copy that can be used for multiple other purposes:

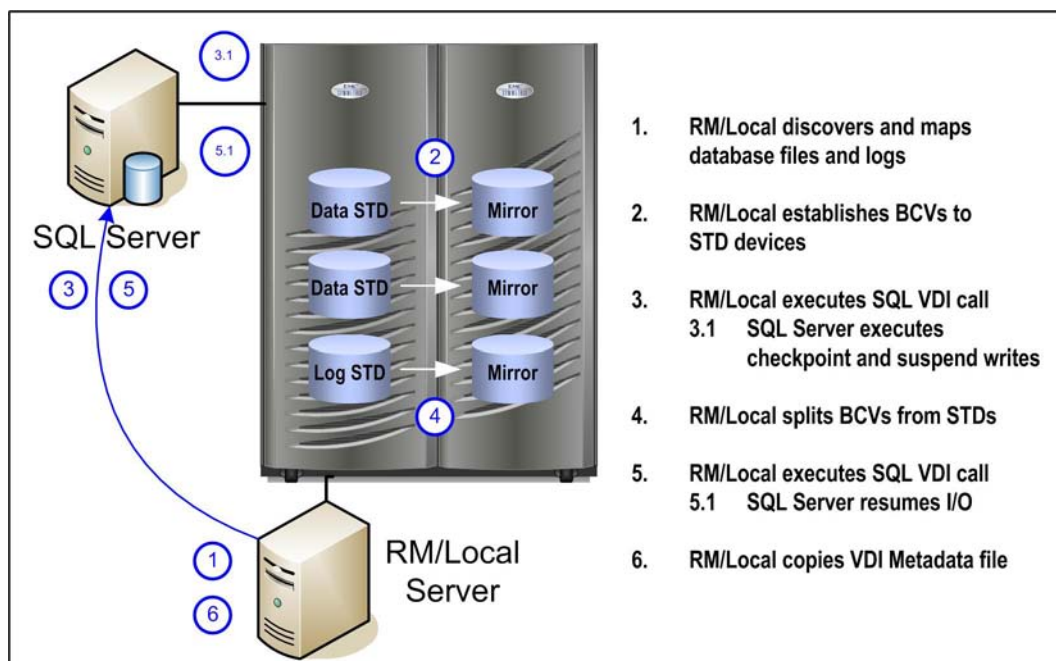


Figure 3-15 Using Replication Manager to make a TimeFinder Replica of a SQL Server database

1. In the first step Replication Manager maps the database locations of all the data files and logs on all Symmetrix devices. Note that the dynamic nature of this activity will handle the situation when extra volumes are added to the database. The procedure will not have to change.
2. Replication Manager then establishes the BCVs to the standard volumes in the Symmetrix. Replication Manager polls the progress of the establish until the BCVs are synchronized and then moves on to the next step.
3. Replication Manager executes a SQL Server VDI database call to checkpoint and write suspend the database.
4. Replication Manager then issues a TimeFinder split, to detach the BCVs from the standard devices.
5. Next, Replication Manager executes another SQL Server VDI call to resume all I/O operations to the database.
6. Finally, Replication Manager copies the VDI metadata file from the source host to an RM holding area for use later in a restore.

3.7 Transitioning disk copies to Microsoft SQL Server databases clones

How a database copy can be enabled for use depends on how the copy was created. A database copy created using the SQL Server VDI process can be processed by using the restore process based on the utility used for its creation. These utilities implement the appropriate SQL Server VDI processing to facilitate a restore procedure using the VDI metadata file.

If a copy of a running database was created using EMC Consistency technology without using the SQL Server VDI feature, it can only be restarted. Equally, an image created while the SQL Server Instance or database was shutdown cannot be processed with VDI functionality. In both of these instances, whether a consistent split of a running database or a copy of a shutdown database, no roll forward log apply is possible to a point in time after the time the copy was created.

The copy created with database shutdown or using EMC Consistency technology can be attached to any SQL Server instance which has appropriate connectivity to the Symmetrix array.

Care must be taken if the copy is to be re-presented to the source server especially in the case of the server being a node in a Microsoft Cluster configuration. This is because of duplicate signature issues for a Microsoft Cluster configuration. In general, presenting mirror devices representing source devices that are part of a Microsoft Cluster configuration is not supported unless specifically documented by the relevant product.

To attach the copy created from a shutdown database or using EMC Consistency technology, it is possible to utilize the SQL Server Transact-SQL `sp_attach_db` procedure. This procedure allows for the relocation of the data files and logs, and for the renaming of the copied database.

This section details how to use the `sp_attach_db` procedure, how to restart a database copy created using EMC Consistency technology, and how to deal with host-related issues when processing the database copy. Also discussed is the usage of SQL Server VDI processes to create database instances where the initial state was created using a VDI compliant utility.

3.7.1 Instantiating clones from consistent split or shutdown images

In most cases, when creating a copy of a database and utilizing it on a different server, the database manager on the new server must be made aware of the new database that is coming under its control. This is done by telling the database manager the locations of the copy files. In the SQL Server environment this may be facilitated by utilizing the SQL Server graphical user interface to attach the database, or by utilizing SQL Server Transact-SQL statements. The `sp_attach_db` stored procedure looks similar to this:

```
sp_attach_db <new_database_alias>,  
@filename1='<new_file_location>',  
@filename<n>='<new_file_location>'
```

Where `<new_file_location>` represents the locations of all database data files and logs for the database copy. Refer to SQL Server Books On Line for complete documentation on this Transact-SQL stored procedure.

In Figure 3-16 on page 3-29, a consistent split image is attached to a SQL Server instance. It assumes that all the mirrored devices, be they BCVs, clones, or snaps, are presented to the target server and mounted in appropriate locations. If the mount locations are identical to those on the production instance, then it is only necessary to specify the first file location (the SQL Server metadata file for the database). Since the metadata file contains information for the locations of all subsequent files and logs, the

stored procedure will use this information. If the file locations have been modified, then all the new locations must be specified.

SQL Server will perform necessary roll forward/roll back recovery on the database instance. Since the Write Ahead Logging functionality of SQL Server has been maintained, the resulting cloned instance will return to a transactional consistent state, and become a fully independent clone of the production instance. Note in Figure 3-16 that transaction roll-forward and roll-back operations are shown during the attach process as viewed through Query Analyzer.

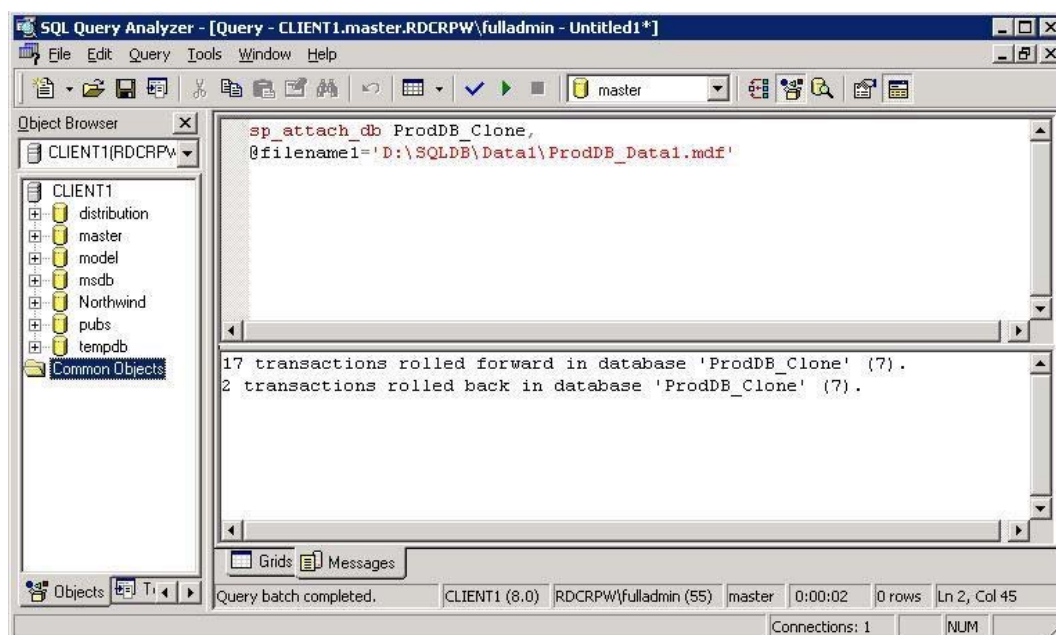


Figure 3-16 Attaching a consistent split image to SQL Server

3.7.2 Using SQL Server VDI to process the database image

A database copy of a SQL Server database created by using the SQL Server VDI split mirror backup process can be subsequently restored for use in a number of ways, using the respective VDI utility that was utilized in the initial creation. The SQL Server VDI process can initialize the database copy into a number of different modes:

- ◆ As a fully independent cloned instance
- ◆ As a standby database, providing read-only access

3.7.2.1 Creating a cloned database using VDI processing

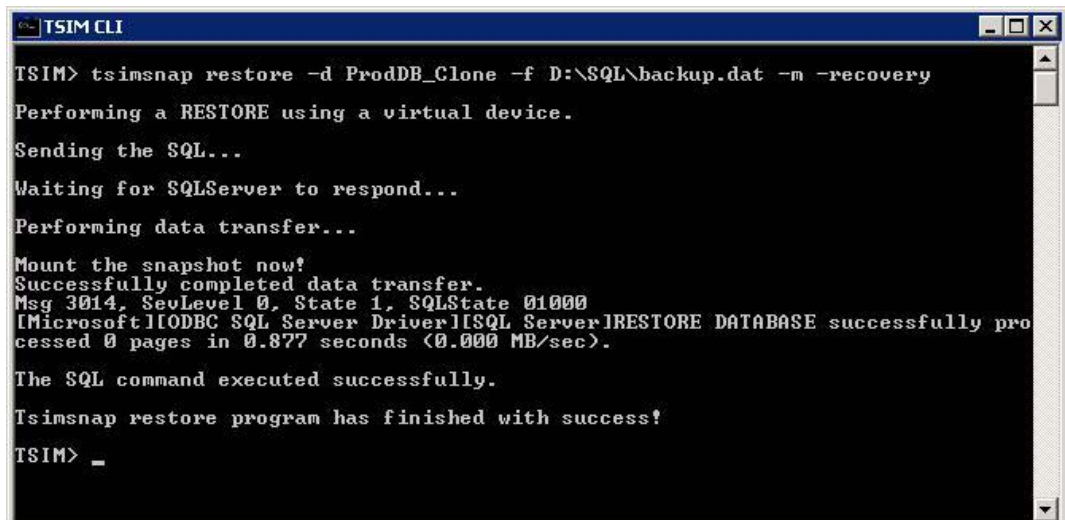
The command to create a cloned database instance on a separate target server using TF/SIM VDI processing is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -m -recovery
```

This process needs to be facilitated by the `-m` command line option, as it is considered a manual recovery process. As a result, this command expects that the file locations for the various data and log files are the same on the target server as they were on the production system. The VDI metadata file created during the backup process for TF/SIM, which relates to this mirrored device set, must be accessible on the host executing the restore operation, and identified using the `-f` option. The `-recovery` option indicates that SQL Server should perform any and all recovery processes on the data files and thus transform the database into a new instance. An example of the execution of this process is shown in Figure 3-17.

Using the mirror devices to instantiate a database clone will modify the state of the files on those devices. Once a VDI backup image has been modified in this way, it will no longer be possible to restore this specific instance back to the production system, and should no longer be considered a “backup” image.

This process results in the creation of a new transaction log chain. In-flight transactions that were active at the time the database copy was created are backed out to the last commit. This cloned database instance is now available for access and represents the state of the database at the point in time the VDI backup was created.



```

TSIM CLI
TSIM> tsimsnap restore -d Proddb_Clone -f D:\SQL\backup.dat -m -recovery
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 0.877 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _

```

Figure 3-17 TF/SIM and SQL Server VDI to create a clone

Equally, a VDI backup image created by using SYMIOCTL may also be restored in a similar manner by using the following command:

```

symioctl -type SQLServer restore snapshot
<database alias> SAVEFILE <file location> -noprompt

```

The VDI metadata file created during the backup process for SYMIOCTL, which relates to this mirrored device set, must be accessible on the host executing the restore operation and identified using the SAVEFILE option. An example of the execution of this process is shown in Figure 3-18 on page 3-31.

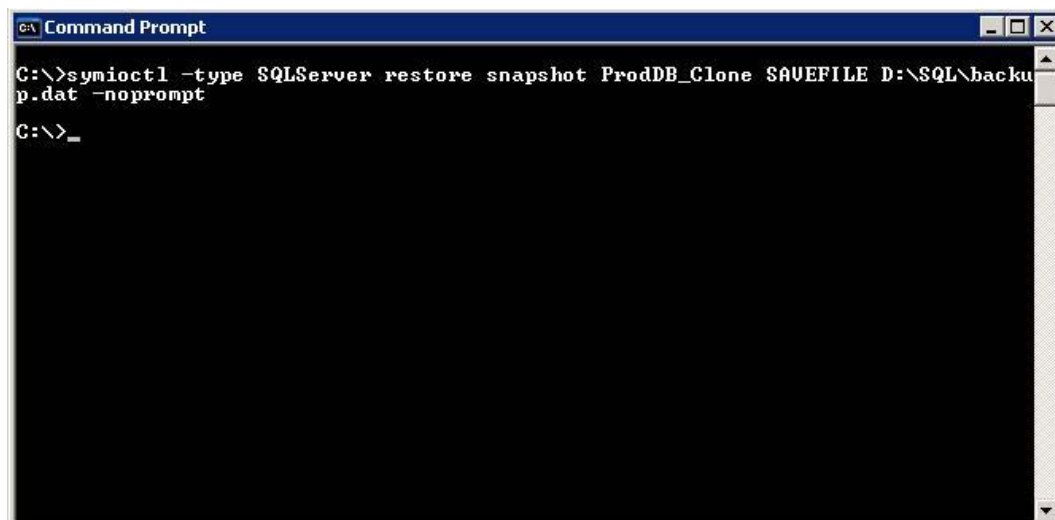


Figure 3-18 SYMIOCTL and SQL Server VDI to create a clone

3.7.2.2 Creating a STANDBY database using VDI processing

SQL Server supports the implementation of STANDBY database environment, where a target database instance created from a backup image of the production system may apply subsequent incremental transaction log backups. In the STANDBY state, the database is available for READ-ONLY access during periods of time while transaction logs are not applied to the standby instance. This process implements a data state, which is continually updated as transaction log backups are processed. Databases placed in this recovery mode are only accessible for read operations, and can therefore be used for those processes that need only access a clone of the production database for interrogation or data extraction purposes.

The following is an example of the TF/SIM command to create a standby database, which can be used for other purposes such as decision support and regression testing:

```
tsimsnap restore -d <database_alias> -f <metafile> -m standby
```

The standby keyword specifies that the standby database is to be used as a backup copy, which can be used to restore the primary database. After execution of this command, the cloned database is left in a READ-ONLY state, and may also have subsequent incremental transaction logs applied. An example of the execution of this process is shown in Figure 3-19 on page 3-32.



```

TSIM CLI
TSIM> tsimsnap restore -d ProdDB_Clone -f D:\SQL\backup.dat -m -standby
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 1.040 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _

```

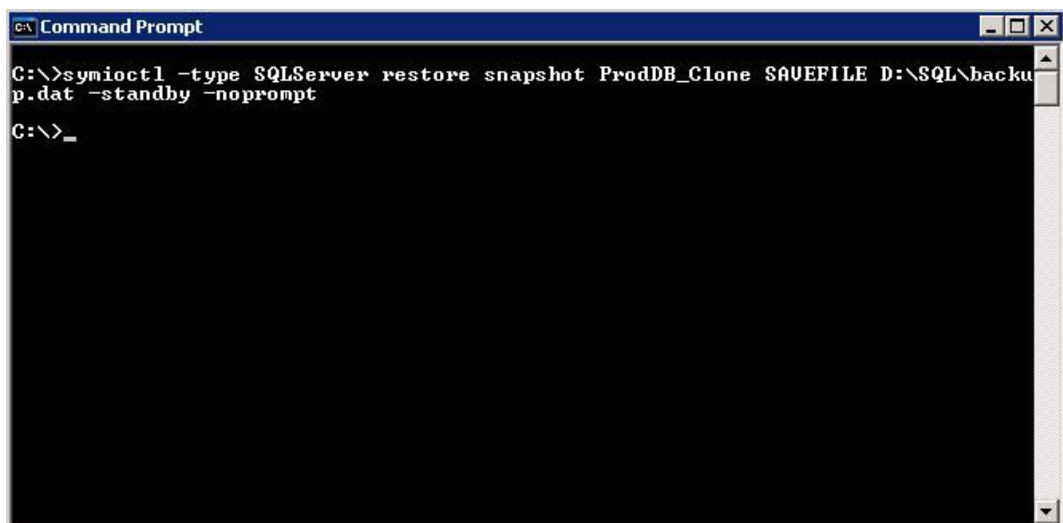
Figure 3-19 Using TF/SIM and SQL Server VDI to create a standby database

Similarly, a VDI backup created using SYMIOCTL may be restored in a manner which allows for subsequent transaction logs to be applied.

The SYMIOCTL command to create a recovering database from a VDI backup is:

```
symioctl -type SQLServer restore snapshot <database alias> SAVEFILE <metafile> -
standby -noprompt
```

The *-standby* option implements the same functionality as documented for the TF/SIM preceding command. The VDI metadata file created during the backup process for SYMIOCTL, which relates to this mirrored device set, must be accessible on the host executing the restore operation, and identified using the SAVEFILE option. An example of the execution of this process is shown in Figure 3-20.



```

C:\>symioctl -type SQLServer restore snapshot ProdDB_Clone SAVEFILE D:\SQL\backu
p.dat -standby -noprompt
C:\>_

```

Figure 3-20 Using SYMIOCTL and SQL Server VDI to create a standby database

As indicated, it is possible to apply incremental transaction log application to this standby database. In the context of creating a cloned environment, this may be useful to occasionally update the database state, and thereby provide more current information from the clone. Log shipping functionality and behavior is documented in Chapter 6.

3.7.2.3 Relocating a database copy

Relocating the data files and logs of the copied SQL Server database is a requirement if mounting the database back to the same server that hosts the source database, or if database locations have changed for whatever reason. Both the TF/SIM utility and the `sp_attach_db` Transact-SQL stored procedure support database relocation. The components that can be identified in this way are:

- ◆ Database name
- ◆ Data filepaths
- ◆ Log filepaths

For the `sp_attach_db` stored procedure, it is only necessary to provide the new locations of the new cloned database. This should be done for each logical database component. The example shown in Figure 3-21, attaches a new `ProdDB_Clone` database with relocated data and log file locations.

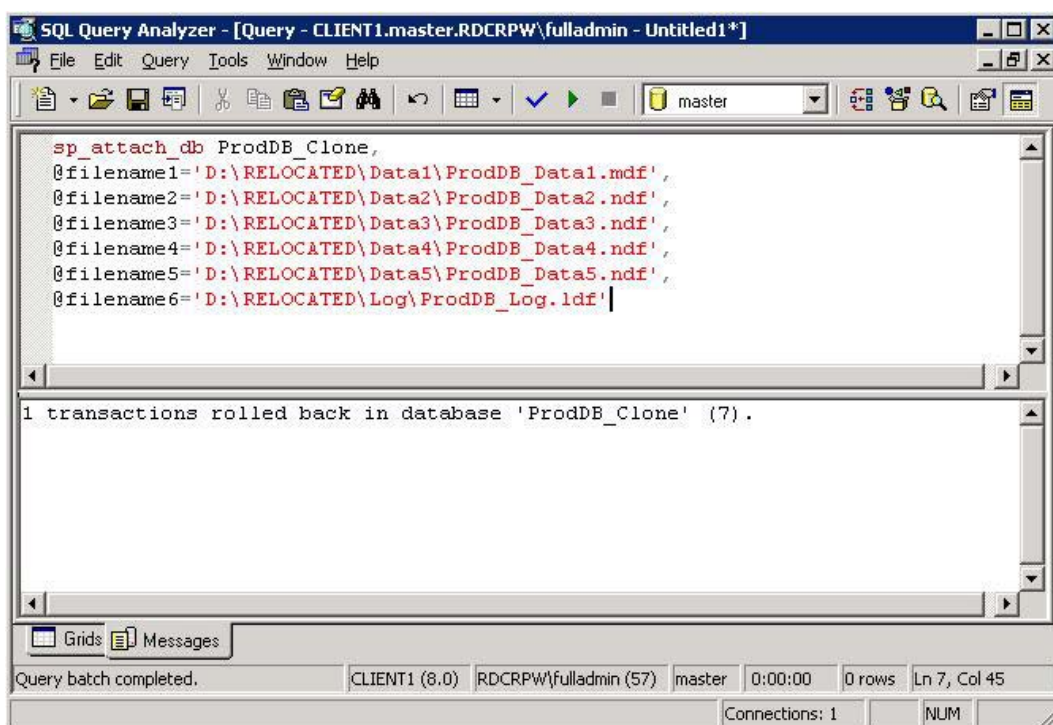


Figure 3-21 Attaching a cloned database with relocated data and log locations

In case of TF/SIM, the logical component names for the various devices are required. It is possible to interrogate the existing database to find the logical names by executing the `sp_helpdb` stored procedure.

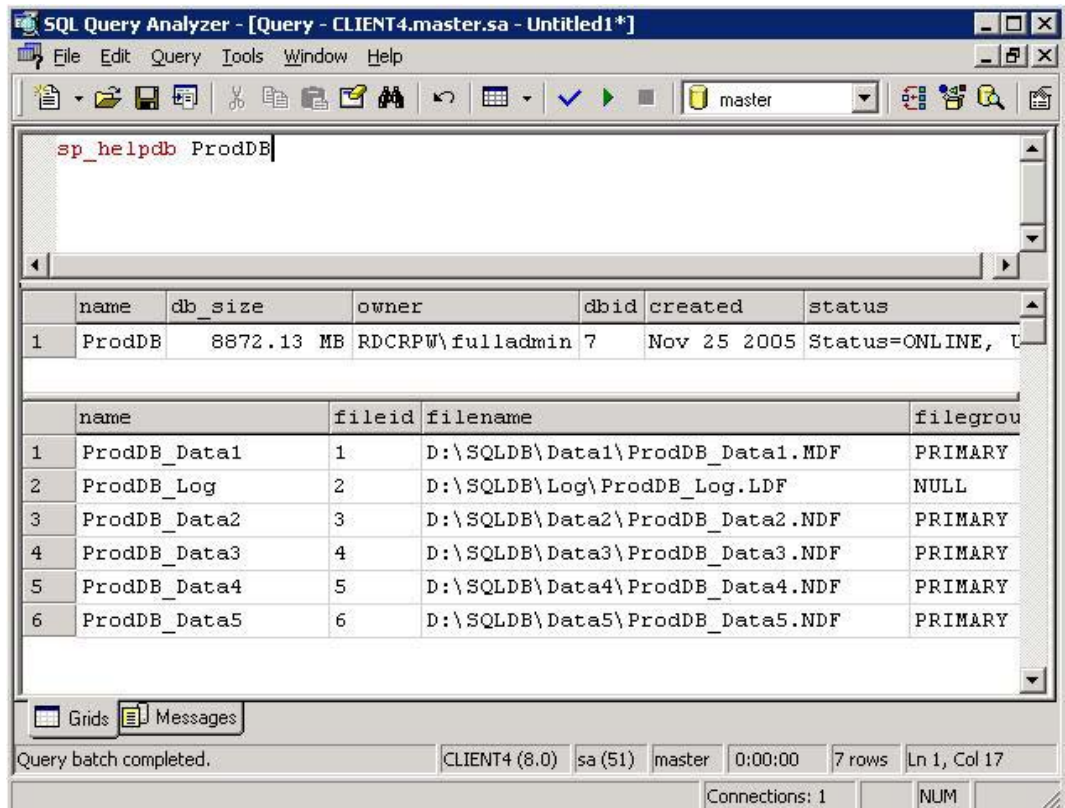


Figure 3-22 SQL Query Analyzer executing `sp_helpdb`

The logical SQL names for the physical files comprising the database are listed in the *name* column. Utilizing this list, it is possible to create a text file which lists each logical name with a new location, as shown in Figure 3-23.

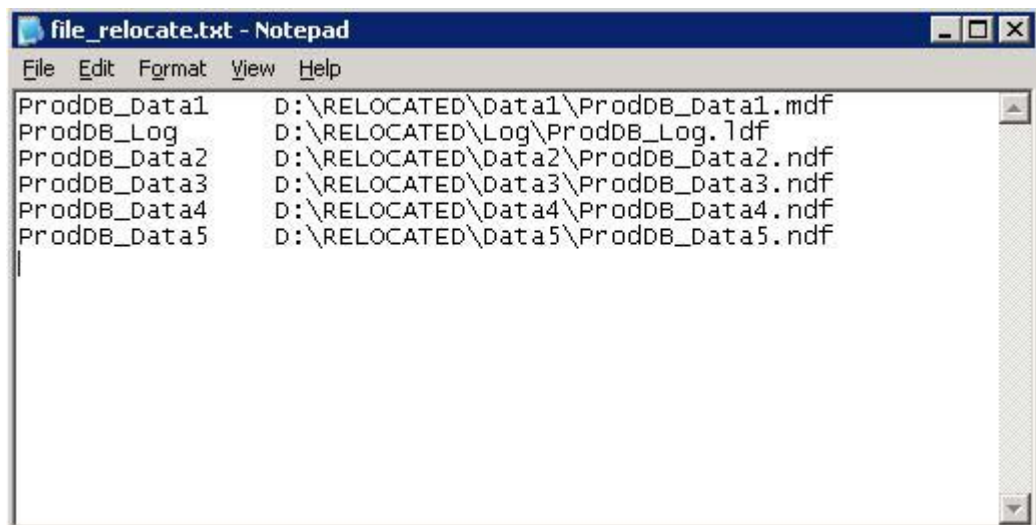
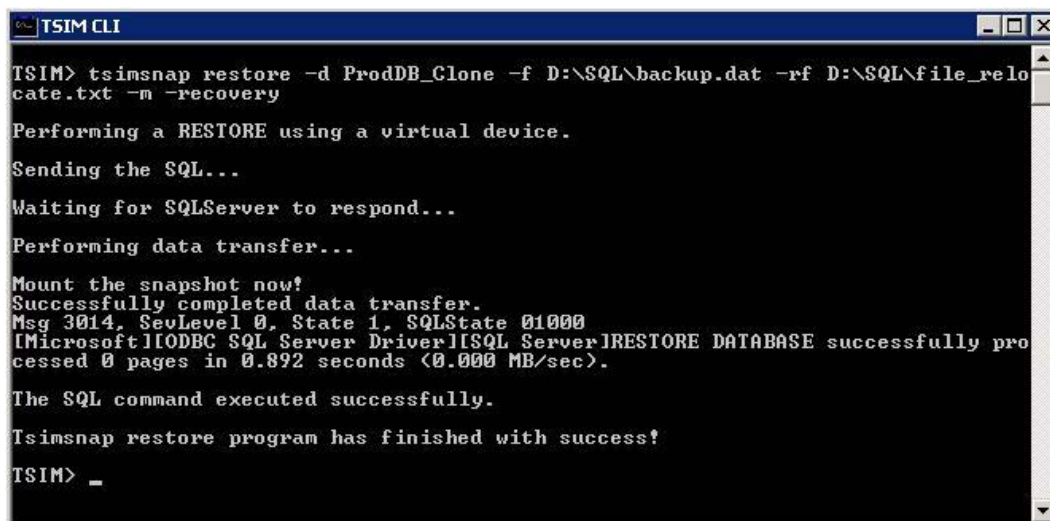


Figure 3-23 Mapping database logical components to new file locations

When relocating a database using TF/SIM, as shown in Figure 3-24, the command should be constructed as follows:

```
tsimsnap restore -d <database_alias> -f <metafile>
-rf <relocate mapping file> -m -<recovery style>
```



```
TSIM CLI
TSIM> tsimsnap restore -d ProdDB_Clone -f D:\SQL\backup.dat -rf D:\SQL\file_relo
cate.txt -m -recovery
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 0.892 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _
```

Figure 3-24 Using TF/SIM and VDI to restore database to a new location

3.8 Reinitializing the cloned environment

In many cases, it may be necessary to reinitialize the clone environment. To facilitate this process, it is necessary to reverse some of the processes that were executed to create the database clone using the specific process required.

A number of steps are required to remove the clone database from the target server so that a reinitialization may be processed.

1. Drop or detach the cloned database instance from the target server. This may be executed via the appropriate Management UI, or by executing the `sp_detach_db` Transaction-SQL statement.
2. Unmount the volumes from the target server using appropriate `symntctl` commands.
3. Reestablish BCV relationship or recreate sessions as appropriate.
4. Execute the appropriate split functionality as initially used.
5. Remount the volumes using appropriate `symntctl` commands.
6. Attach the cloned database in the same manner as originally processed.

3.9 Choosing a database cloning methodology

The replication technologies described in previous sections each have pros and cons with respect to their applicability to solve a given business problem. The following matrix

provides a contrast to the different methods that can be used and the differing attributes of those methods:

Table 3-1 A comparison of database cloning technologies

	TF/Snap	TF/Clone	TF/Mirror	Replication Manager
Maximum No. of copies	15	Incremental: 16 Non-inc: Unlimited	Incremental: 16 Non-inc: Unlimited	Incremental: 16 Non-inc: Unlimited
No. simultaneous Copies	15	16	2	2
Production Impact	COFW	COFW & COA	None	None
Scripting	Required	Required	Required	Automated
Database clone needed a long time	Not recommended	Recommended	Recommended	Recommended
High Write Usage to DB Clone	Not recommended	Recommended	Recommended	Recommended

COFW = Copy on First Write

COA = Copy on Access.

Table 3-2 shows examples of the choices you might make for database cloning based upon this matrix:

Table 3-2 Database cloning requirements and solutions

System Requirements	Replication Choices
The application on the source volumes is very performance sensitive and the slightest degradation will cause responsiveness of the system to miss SLAs	TimeFinder/Mirror Replication Manager
Space and economy are a real concern. Multiple copies are needed and retained only a short period of time, with performance not critical.	TimeFinder/Snap Replication Manager
More than 2 simultaneous copies need to be made. The copies will live for up to a month's time.	TimeFinder/Clone
Multiple copies are being made, some with production mount. The copies are reused in a cycle, expiring the oldest one first.	Replication manager

Chapter 4 Backing up Microsoft SQL Server Databases

This chapter presents these topics:

4.1	EMC Consistency technology and backup	4-2
4.2	SQL Server backup functionality	4-3
4.3	SQL Server log markers	4-8
4.4	EMC products for SQL Server backup.....	4-9
4.5	TF/SIM VDI backup.....	4-13
4.6	SYMIOCTL VDI backup	4-20
4.7	Replication Manager VDI backup	4-22
4.8	Saving the VDI backup to long-term media	4-23

To mitigate the possibility of complete or partial data loss of a production database, it is necessary to implement some form of backup and recovery process to ensure that a separate, logically valid copy of the database is always available. If the production database is rendered unusable, the logically valid copy may be restored to the production system. Clearly, the currency of the data state of the copied database is important, and backup processes typically implement functionality to ensure that there will be minimal data loss.

For most production databases the requirements for availability are such that the database is accessible 24 hours a day, seven days a week. Thus, all backup operations must be processed online with minimal impact to the production system. Inevitably, as the production database size grows, the ability to process a full backup becomes more time consuming. While online streaming backup processes can have minimal impact, their duration and timing become a substantial issue.

This chapter describes SQL Server backup processes, and how database administrators can leverage EMC technology in a backup strategy to:

- ◆ Reduce production system impact during backup processing
- ◆ Create consistent backup images
- ◆ Integrate SQL Server backup/recovery processes

4.1 EMC Consistency technology and backup

In Chapter 3, the use of EMC Consistency technology was described as a method to create restartable copies of a source SQL Server database. This type of technology is noninvasive to the production database operations and occurs entirely at the storage array level. The restartable images represent dependent-write consistent images of all related objects that have been defined within the consistency group.

Restartable images created using Consistency technology do have valuable usage when customers are concerned with federated environments, where the creation of a consistent restart point for all databases and other objects within the environment is the primary goal. Independently created recoverable images created of each individual component of a federated environment make it inordinately complex, if not impossible, to resolve a single business point of consistency across all systems.

It is possible to maintain local images or stream to longer term media these copies of a SQL Server database created from procedures which do not utilize SQL Server's BACKUP DATABASE Transact SQL statement, such as those created from a SQL Server shutdown state, or from EMC consistent split images.

The Recovery Point Objective (RPO) of such a restartable image is based on a static point in time, and will increase as the source production environment changes. It is only when the next point of consistency is created that the RPO can be reset to its initial state. During intervening times between cycles, the RPO point will vary. It is important to control the cycle of such a process to ensure that the solution falls within the maximum RPO.

4.2 SQL Server backup functionality

Microsoft SQL Server provides a number of standard tools and interfaces to facilitate backup processing. Within the standard product, administrators can use the built-in backup tools presented by both the SQL Server Enterprise Manager and the SQL Server T-SQL programmatic interface.

SQL Server Enterprise Manager provides a graphical user interface (GUI) to the major administrative functions within the database environment. In Figure 4-1, an example of the interface to the backup functionality is provided.

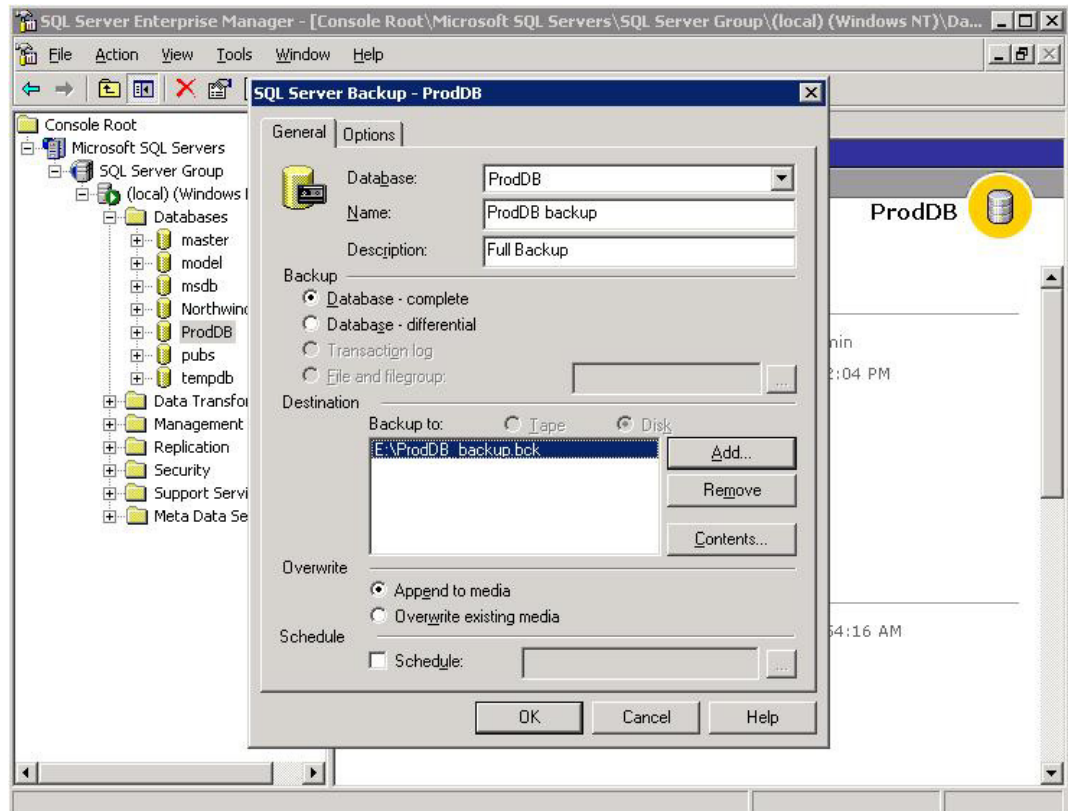


Figure 4-1 SQL Enterprise Manager backup interface

Standard SQL Server backup functionality typically allows for backup of a database to a tape device, or a file within a file system. In Figure 4-1, a disk file device is to be used for the backup operation. Additionally, the SQL Server Enterprise Manager provides a scheduling mechanism (and a maintenance plan), which allows for backup operations of this form to be processed. Within the Transact-SQL command interface, a similar process could be invoked through the BACKUP DATABASE statement.

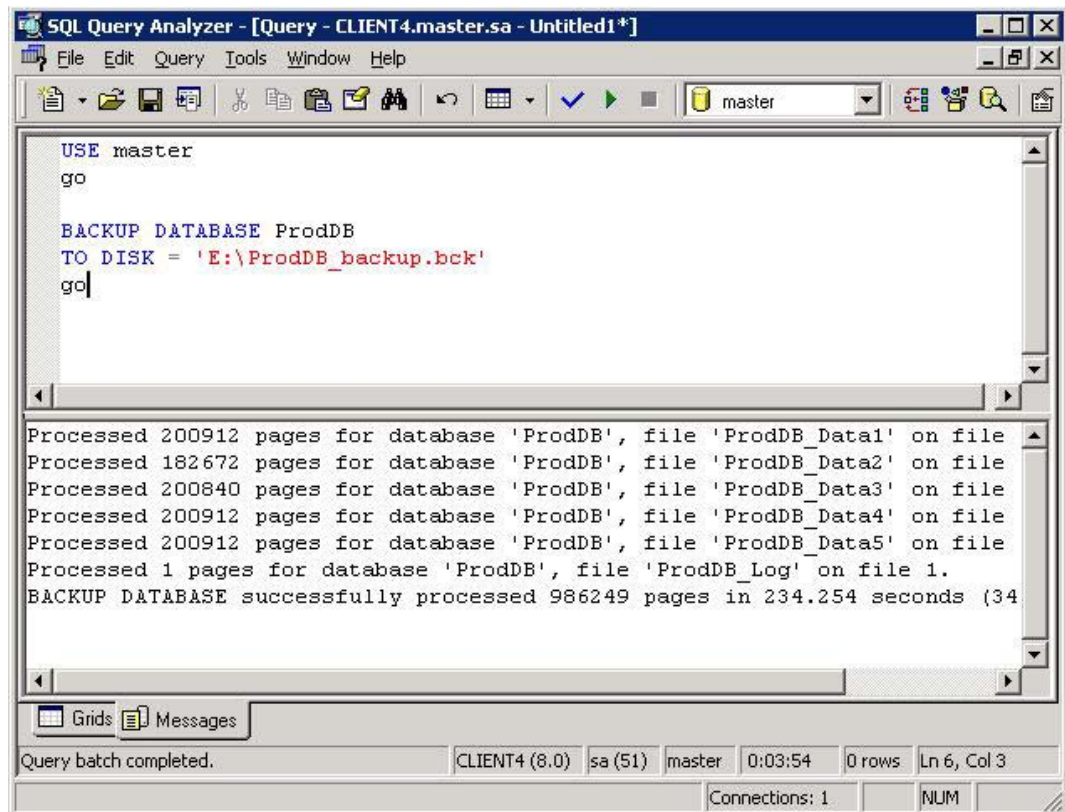


Figure 4-2 DATABASE BACKUP Transact-SQL execution

Executing regular backup processes, as shown in Figure 4-2, provides a recovery procedure if serious corruption or failure of the production server and/or the database itself. Backups can be executed in a number of ways, but at the core is the full database backup. It is from a full backup that recoveries are generally processed when required. Full backups also facilitate other incremental backup functions such as incremental transaction log backups. Restoration of these incremental backup processes then allow for a more efficient method of ensuring the currency of the state of the database.

Microsoft SQL Server fully supports online backup processes and these are utilized through the examples in this chapter. When a production database is relatively small, full online backup operations are very fast and provide little operational issue. As database environments grow, such operations can begin to impact the production system. Backups can begin to interfere with user activity, and this is when the value of split mirror disk technology becomes more relevant.

Before looking at the various EMC technologies applicable in the backup/recovery space for SQL Server, it is important to understand the various recovery modes available to a SQL Server database.

4.2.1 Microsoft SQL Server recovery models

The recovery models defined by SQL Server affect the behavior of the transaction log management for a database. SQL Server does not implement functionality in the same manner as other RDBMS environments in this respect. There is no functionality that

could be referred to an automatic “archive log”. The only mechanism that provides similar functionality is that of the execution of incremental transaction log backups. This behavior is controlled by the *recovery models*, as defined in the next sections.

The recovery models are important to understand as they affect backup, and thus recovery and operations.

4.2.1.1 SIMPLE recovery model

In this model, SQL Server attempts to manage the size of the transaction log by recovering transaction log space which is no longer required for transactions which have been committed. This reclaimed space is then used to record additional in-process transactions. Other RDBMS environments may refer to this as “circular logging”.

In this model, SQL Server can protect the database from events such as a server restart, but is unable to recover from a backup and roll forward. Transaction log backups are not required or supported in this mode. Therefore, there is no chain of transaction log backups to replay. The Recovery Point Objective (RPO) of this state in the event of complete system loss is that of the last full backup.

4.2.1.2 FULL recovery model

In this model, all updates to the database are recorded in the transaction log and the log is neither automatically truncated nor is the space within the transaction log re-used dynamically. Re-use is only implemented after incremental transaction log backups.

The use of the full recovery model ensures the creation of a continuous sequence of transaction log records that allows for recovery of a given database to essentially any required point. Incremental transaction log backups can be used in combination with full backups to recover from a total system loss to the point in time of the last transaction log backup.

In general, until a full database backup has been completed for a database in “full recovery” model, the database will behave like it is in the “simple recovery” model. After the first full backup, the full recovery functionality is implemented.

4.2.1.3 BULK-LOGGED recovery model

One major problem for customers is the size of the transaction log file when certain operations are executed. In many situations, data load processes may be executed in certain environments. When utilizing the full recovery model, the size of the transaction log may grow to a significant size. To mitigate this effect, certain operations can be minimally logged in the “bulk-logged” model. This provides a good deal of the recoverability of the full recovery model and addresses some of the transaction size issues under certain conditions.

It is possible to switch between the various recovery models. Microsoft documents the restrictions on this switching process in the Books On Line documentation under “Switching Recovery models”.

For the purposes of the remainder of this document, we will assume that the database is in the full recovery model since this is the most applicable form. Implementing the simple recovery model is not recommended or supported by EMC to be used with any production database.

The recovery model for a database may be set either through the SQL Server Enterprise Manager, by selecting the properties of the required database. In the case of our example, the ProdDB database has the full recovery model selected. In Figure 4-3, the other forms of recovery are displayed in the drop-down menu.

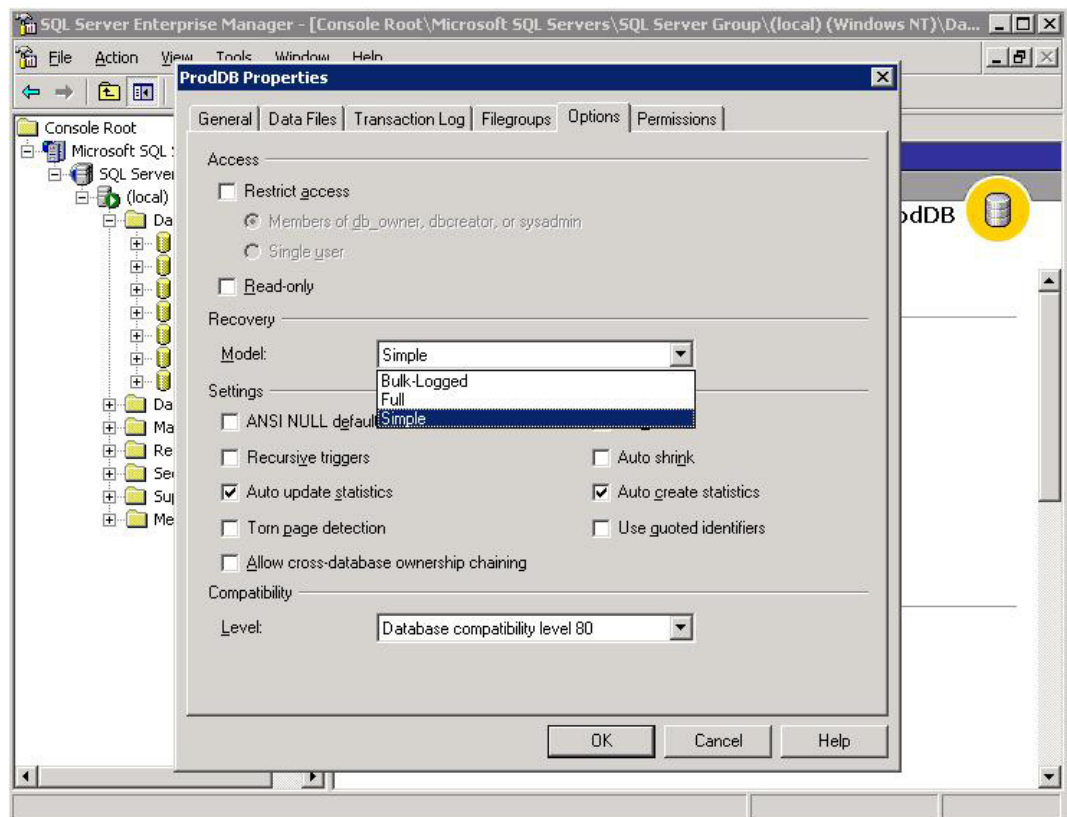


Figure 4-3 Recovery model options for a SQL Server database

Modification of the recovery model used for a given database is also supported via the Transact-SQL command ALTER DATABASE.

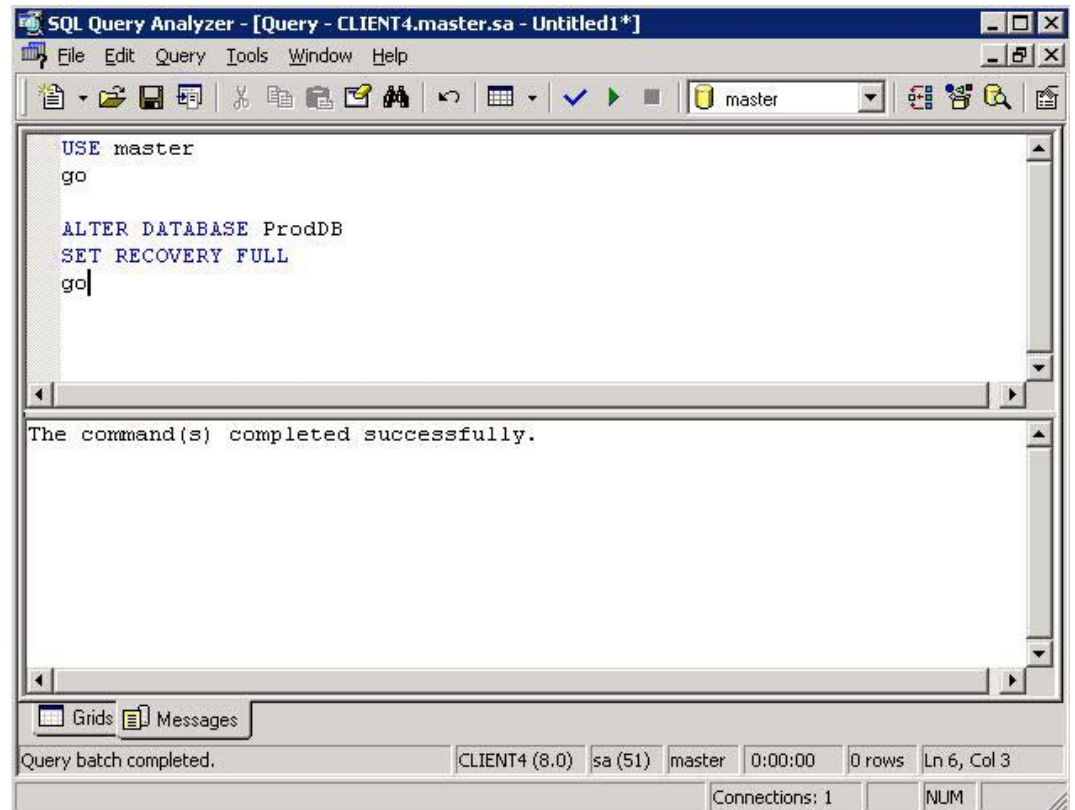


Figure 4-4 Setting recovery model via Transact-SQL

Customers may find the need to switch between the full recovery and bulk-logged model to cater for operational requirements and to mitigate transaction log growth rates during load operations. Switching between the full and bulk-logged model is fully supported and does not require any specific change in operational behavior. Customers should understand the limitations of the bulk-logged model of operations, and should utilize the full recovery model where possible.

4.2.2 Types of SQL Server backups

Microsoft SQL Server supports multiple forms of database backups. The simplest of all to consider is the full backup. This is a complete set of the data and transaction log files for the specified database. Clearly, this form of creating a full database backup can be expensive in terms of processing time and impact to the production environment. In the case of online backup to tape, SQL Server's database engine will require CPU and IO resources to stream all database pages out of the datafiles and transaction log..

In addition to the full backup, SQL Server supports a differential backup process. This form of backup copies only the changed data since the last full backup. It can be restored after a full backup to return the database to the state at the time the differential backup was executed. This form of backup can be used in combination with transaction log backups to return to a consistent state.

Incremental transaction log backups are the last form of backup discussed here. Transaction log backups copy all the current transactional activity recorded for a

database from the transaction log. Once the transaction log data has been backed up and is no longer required for transactional recovery, the SQL Server database engine can reuse space within the transaction log file.

As discussed in section 1.5, a physical transaction log is subdivided by SQL Server into a number of logical virtual log files. Once a virtual log is full and does not contain any information pertaining to those transaction log records which constitute the active portion of the transaction log, the virtual log may be marked for reuse. This occurs after an incremental transaction log backup has been executed, and the log records have been backed up.

The mechanisms that control this reuse are more complex than described here. Additional, extensive discussion of the transaction log, log records, virtual logs, and active log definition may be found in the SQL Server Books On Line documentation.

SQL Server allows for the restoration of a full database backup and application of incremental transaction log backups created since the last full backup. In this way, the amount of data loss is minimized, and any data loss will be related to the amount of time since the last incremental backup. The full chain of incremental logs must be replayed in the order in which they were created.

Both differential backups and incremental transaction log backups help to mitigate the processing time required for creating a logical backup process, this time relates directly to a requirement often referred to the recovery time objective (RTO). The RTO determines how much time is allowed to return to database service. This is independent of what actual process may be used. For example, RTO does not speak to whether a D/R solution is being used, or whether a restore to the production system is required. It is simply concerned with the amount of time required to restore service.

4.3 SQL Server log markers

SQL Server automatically maintains timestamp information for transaction log records. The timestamps are based on the system clock of the production system itself. This automatic inclusion of timestamps facilitates rolling forward to a specific point in time when subsequently applying incremental transaction log backups.

In addition to the timestamps, it is possible to record a “marked transaction” within the transaction log. This typically represents a logical point in time; a point in time that may reflect the point before some operational process, rather than one based on a specific time. For example, it may be useful to record the logical point before a large update process within the database. Therefore, in the event of a system failure, it is possible to return to that state before the start of the update.

Recording a marked transaction may be executed in a similar manner to the following example:

```
BEGIN TRANSACTION myUpdate WITH MARK 'Before_Update'  
go
```

```
COMMIT TRANSACTION myUpdate  
go
```

Restoration to a logical transaction marker is discussed in Chapter 5.

4.4 EMC products for SQL Server backup

In general, it can be considered that EMC products are positioned to facilitate the core full backup process implemented by SQL Server. Since the functionality of storage arrays is generally based on the logical unit number (LUN), they facilitate backup and restore operations at the full LUN level. This means that incremental backup functions, which are typically processed by the SQL Server engine, cannot be executed by a disk mirror environment.

However, it is possible to implement functions such as file group backup processes, which allow for the creation of backup images of only parts of the entire database. It should be noted that the backup of the file group is only implemented as a full backup of the file group itself, rather than an incremental backup of the file group. This process then allows for the separate restoration of a single file group should that be appropriate, rather than the restoration of the entire physical database.

It is expected that customers will incorporate EMC tools that create full backups with additional processes that serve to enhance backup and restore functions. Therefore, it is typical to expect that full backups created at a storage level are enhanced with regular transaction log backups used to mitigate any data loss should a restore be necessitated.

To create a valid disk mirror backup image of a SQL Server database, it is necessary to interface to the owning SQL Server instance itself. This functionality is required to quiesce the database and suspend I/O so that a split mirror image may be created. EMC has created a number of products and utilities to support the broad range of operational modes that various customer environment require.

These products and utilities range from graphical user applications, such as Replication Manager, to command line products such as TimeFinder SQL Server Integration Utility (TF/SIM). Additionally, within the Solutions Enabler product, command line utilities exist, which can initiate these processes in discrete steps. This range of products allows customers to select the best solution to match their specific requirements.

4.4.1 Integrating TimeFinder and Microsoft SQL Server

Integration of EMC's TimeFinder functionality with Microsoft SQL Server for backup and restore operations centers on the usage of the SQL Server Virtual Device Interface (VDI). Within SQL Server 2000 and SQL Server 2005, Microsoft provides the VDI functionality to support disk mirror snapshot backup operations, which are enabled by storage arrays. Utilizing EMC TimeFinder technology provides support for clones, snaps and traditional TimeFinder BCVs. These technologies are described in detail in Chapter 2.

Since the SQL Server VDI is a programmatic interface, users cannot use this functionality directly, and vendors are required to provide end-user tools to make use of the API. EMC has provided a number of end-user tools to integrate the functionality of the virtual device interface and TimeFinder operations.

Microsoft SQL Server VDI also provides support for a more granular form of file group-level operation. Many of the EMC products provide support for creation and subsequent restoration of these file groups where implemented. The ability to target specific file groups for backup and restore operations may be beneficial in configurations in which only a specific file group may need to be targeted for backup and restore operations. Operationally, it would be more efficient to only designate that file group and exclude the remainder of the file groups, which may be static and unchanging from daily backup cycles. Using file group operations, administrators could optimize backup and restore operations to implement backups against a specific target file group. It is also possible to specify a particular file group for restore operations from a disk mirror backup based on a full backup of the database. This allows for maximum flexibility for all customer requirements.

The tools typically fall into two categories: command line interface (CLI) tools and graphical user interface (GUI) tools. In the CLI area, EMC provides the TimeFinder SQL Server Integration Module (TF/SIM) as a separate product, and the SYMIOCTL tool as a part of the Solutions Enabler package. In the GUI area, users can take advantage of Replication Manager (RM) to control the creation and management of backup images (referred to as replicas). Additionally, RM allows for the automation and scheduling of replica creation.

Both TF/SIM and RM provide interfaces to backup and restore databases using database snapshot backup/restore functionality through the SQL Server Virtual Device Interface (VDI). They both provide an integrated method for quickly creating point-in-time backups of databases, which can then be used for decision support systems, database maintenance commands, and disaster recovery.

4.4.1.1 Disk mirror backups vs. streamed backups

A VDI snapshot database backup differs from a streamed backup because the actual data pages that are normally streamed out through SQL Server itself during a standard backup do not have to be processed in this manner. This is because the disk mirror (BCV/snap or clone) results in an identical copy of the database. During the VDI snapshot creation, SQL Server ensures that the image on the production volume (and therefore on the relevant disk mirrors) are placed in a valid backup state. Creation of this valid image does require SQL Server to execute a checkpoint to flush all updated pages to the data files, and then suspend all updates. As all updates are suspended for the snapshot creation, all write activity to the transaction log is also suspended. This provides a logically consistent disk state for the database because the transaction log and data file states will be flushed to disk.

The time it takes to checkpoint the database, write the metadata file to disk, and split or activate the disk mirror device from their source volume determines the total duration of the entire snapshot backup process. In comparison, the duration of a traditional backup is determined by the time that it takes to write every allocated data page in the database to disk or tape.

The duration of the I/O suspension is only for the disk mirror split or activate process.

During the VDI snapshot backup, some additional data, referred to as metadata, is recorded during the execution. This metadata is usually in the order of tens of kilobytes

(KBs) in total size and contains information about the various files that make up the database and some additional log information. This metadata can be used during a restoration phase to automate much of the processing.

4.4.1.2 Effects of SQL Server VDI backups on SQL Server databases

A VDI snapshot backup generally has minimal impact on database performance during its execution. Most importantly, user connections to the SQL Server are not broken during this process. Read access is maintained while write operations are suspended. Those transactions, which execute a commit operation while the VDI backup is being processed, may be suspended because of their write requirement. Most VDI backup operations will execute within a matter of seconds, although the VDI implementation itself does not implement a timeout value.

4.4.1.3 Effects of VSS backups on SQL Server databases

The VSS implementation provided with SQL Server 2005 provides a structured framework for executing SQL Server disk-based backup operations. The VSS framework provides an implementation that has similar requirements to that of VDI. Operationally, the effects on I/O operations are similar to that of the VDI, although the VSS framework does implement a threshold value for processing the backup. The threshold value for VSS operations is 10 seconds. If a disk mirror-based backup exceeds this timing, the backup will abort and I/O operations will proceed as normal.

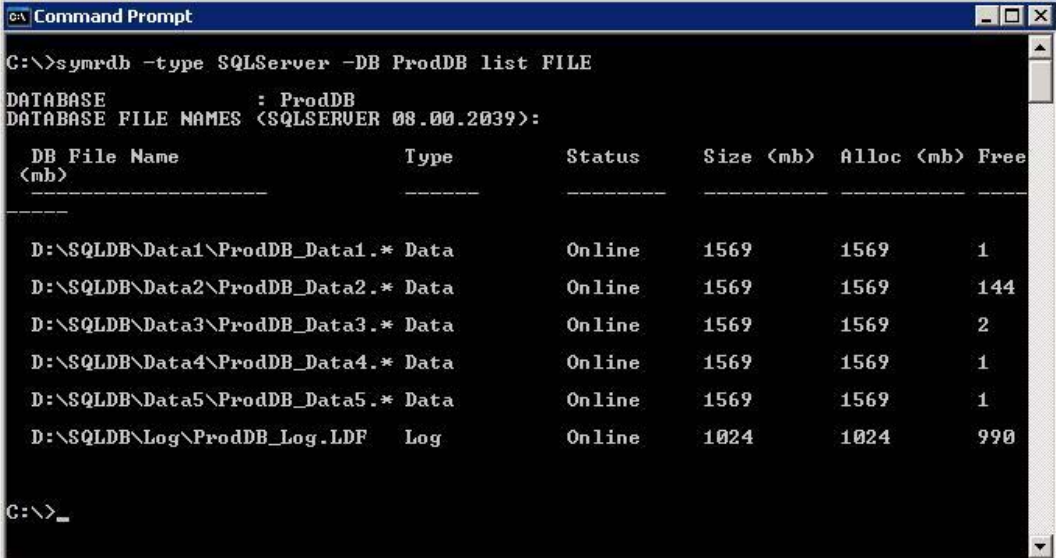
4.4.2 EMC Storage Resource Management

EMC provides additional tools to facilitate creation and management of TimeFinder configurations when using a database environment such as Microsoft SQL Server. A number of those tools are available within the EMC Solutions Enabler package as described in Chapter 2, and form part of the Storage Resource Management (SRM) infrastructure. These tools will be discussed in this section to provide a more complete picture of the various operations. Products such as RM/Local use similar mechanisms, but may not display the underlying operations to the user.

The utilization of the SRM commands is not a specific requirement, but is used here to demonstrate functionality. Products such as Replication Manager monitor and manage required devices in a much more dynamic manner, and do not require that administrators utilize device groups.

As a part of the SRM utility set, the symrdb command line executable provides specific functionality for discovering, managing, and reporting on Microsoft SQL Server Instances, as well as other database platforms.

In Figure 4-5 on page 4-12, the symrdb utility is used to list the database files used by the ProdDB database. In this case, we use SRM's mapping functionality to show the file locations for the database itself. In this form, the utility should provide the same locations as those through the Enterprise Manager GUI and through the Transact-SQL `sp_helpdb` stored procedure.



```

C:\>symrdb -type SQLServer -DB ProdDB list FILE
DATABASE : ProdDB
DATABASE FILE NAMES <SQLSERVER 08.00.2039>:

```

DB File Name <mb>	Type	Status	Size <mb>	Alloc <mb>	Free
D:\SQLDB\Data1\ProdDB_Data1.*	Data	Online	1569	1569	1
D:\SQLDB\Data2\ProdDB_Data2.*	Data	Online	1569	1569	144
D:\SQLDB\Data3\ProdDB_Data3.*	Data	Online	1569	1569	2
D:\SQLDB\Data4\ProdDB_Data4.*	Data	Online	1569	1569	1
D:\SQLDB\Data5\ProdDB_Data5.*	Data	Online	1569	1569	1
D:\SQLDB\Log\ProdDB_Log.LDF	Log	Online	1024	1024	990

```

C:\>_

```

Figure 4-5 SYMRDB listing SQL database file locations

The *symrdb* command line utility can also be used to define any required TimeFinder device groups to be used for managing a split mirror environment. SRM, through the mapping functions, is able to relate SQL Server data and transaction log files to Symmetrix volumes, and then use this information to populate the respective device groups.

In the following example, *symrdb* is used to define a new device group for those Symmetrix devices used by the data and transaction log files of the specified database. Alternate manual methods to create the device groups and allocate volumes are also possible, although utilizing SRM ensures that all devices are appropriately captured.

```
symrdb -type SQLServer -db <database> rdb2dg <device_group>
```

It is not generally possible to use data files independently of the current transaction log, nor is the transaction log independently a source of significant value. Backup/restore processes typically will require both the data and transaction log files to be available together. Therefore, creating and managing a single device group is recommended.

The final step is to add those specific types of mirror devices to the device group to implement the execution of split mirror functionality. BCV devices must be the same size as the source (STD) devices to support TimeFinder operations. Clone devices may be larger than the source STD volumes. For utilization of TimeFinder/Snap functionality, the relevant VDEV devices must be added to the device group.

Details regarding adding BCV, Clone and Snap devices, and establishing TimeFinder relationships can be found documented in the relevant EMC documentation as listed in Appendix A.

The disk mirror devices are also typically presented through the SAN to a backup or target host, which is then able to mount or use the BCV devices in some manner.

Mapping and presenting these split mirror devices is beyond the scope of this document, and may be found in the relevant EMC technical documentation.

Once the relevant mirror devices have been selected, mapped to the appropriate alternate host and synchronized, it is possible to obtain information regarding the volumes in use. Using the respective SYMCLI command, it is possible to query the status of the device group, as shown in Figure 4-6. In this instance, the symmir command is utilized, but it would also be possible to interrogate using the symclone or symsnap command line utilities, should their respective style of devices be used.

```

C:\>symmir -g sql_prod que

Device Group (DG) Name: sql_prod
DG's Type           : REGULAR
DG's Symmetrix ID    : 000184600216

  Standard Device          BCU Device          State
-----
Logical      Sym   Inv. Tracks Logical      Sym   Inv. Tracks STD <=> BCU
-----
DEU001      0039    0 BCU002      006B *    0 Synchronized
DEU002      005A    0 BCU001      0065 *    0 Synchronized
Total
Track(s)              0
MB(s)                 0.0

Legend:
(*) : The paired BCU device is associated with this group.

```

Figure 4-6 SYMMIR query of a device group

In general, before initiating any synchronization, resynchronization, or restore process, you should ensure that any secondary target or backup system is shut down or has the volumes unmounted. Failure to do so leads to a corrupted file system image on the secondary system, and this may result in unpredictable results for the database clone image.

For the initial establish process of the BCU's to the source volumes, a full establish is required to copy all data from the production disks to the BCU's. For example:

```
symmir -g <device_group> -full establish -noprompt
```

During resynchronization (reestablish) operations, it may be more efficient to execute an incremental establish operation (note the removal of the `-full` option):

```
symmir -g <device_group> establish -noprompt
```

4.5 TF/SIM VDI backup

The TimeFinder SQL Server Integration Utility is a command line utility, which interfaces with the SQL Server Virtual Device Interface. Additionally, TF/SIM ensures the appropriate storage-level states for the supported split mirror devices.

It should be noted that the TimeFinder SQL Server Integration Utility can be configured for local or remote modes of execution. The discussions in this chapter refer to only the local form of backup operation, where execution is done on the production server itself.

The different forms of execution have different preconditions for the mirror devices being used. For example, local mode does require TimeFinder/Mirror devices to be synchronized, whereas remote mode requires that they are not currently established. Please refer to the product guide for the TimeFinder SQL Server Integration Module which will list precondition requirements, and execution modes. Refer to Appendix A for information on related documentation.

4.5.1 Using TimeFinder/Mirror

TimeFinder/Mirror is an EMC software product that allows an additional hardware mirror to be attached to a source volume. The additional mirror is a specially designated volume in the Symmetrix configuration called a business continuance volume, or BCV. The BCV is synchronized to the source volume through a process called an establish. While the BCV is established, it is not ready to all hosts. At an appropriate time, the BCV can be split from the source volume to create a complete point-in-time copy of the source data that can be used for multiple different purposes, including backup, decision support, and regression testing.

In general, those EMC products that utilize the SQL Server VDI also implement EMC's Storage Resource Management (SRM), which dynamically map database files to array volumes. Thus, the operations of these tools become independent of SYMCLI device groups which are used by administrators to monitor the volume states and relationships. However, the device groups may be used to manage the TimeFinder state when required for certain preconditions, which may be required. For example, local execution of TF/SIM on production database servers requires that TimeFinder/Mirror devices are in a synchronized state. Device groups would be used to ensure that state is created.

The following process demonstrated in Figure 4-7 on page 4-15 shows the necessary steps required to create a split-mirror database backup of a Microsoft SQL Server database using TimeFinder/Mirror and TF/SIM being executed in local mode on the production server.

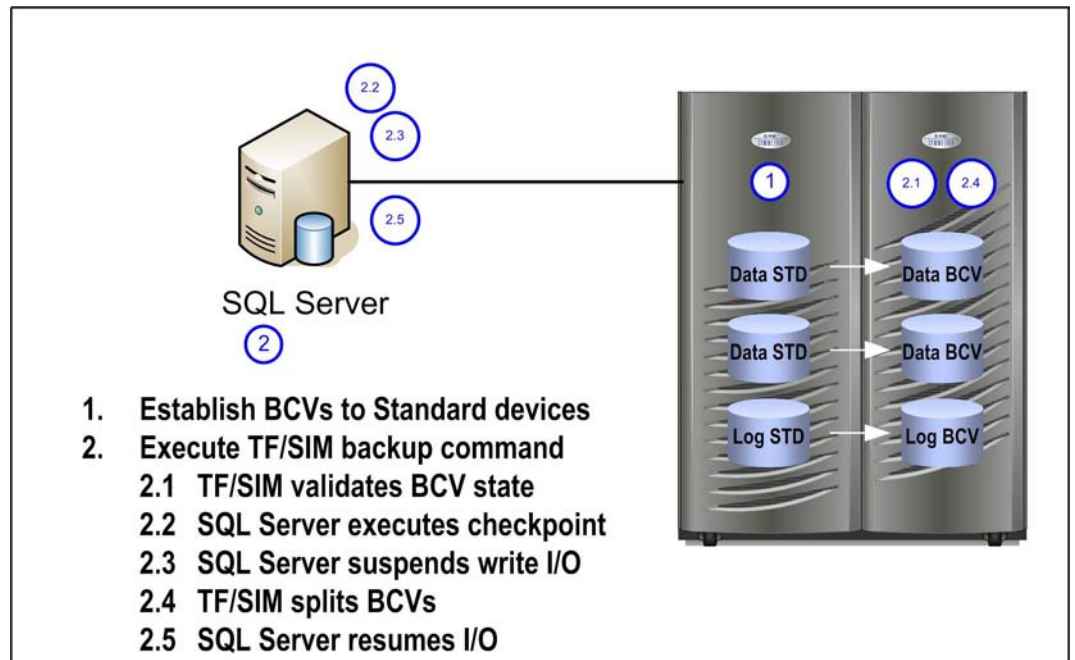


Figure 4-7 Creating a TimeFinder/Mirror VDI backup of a SQL Server database

1. The first action is to establish the BCVs to the standard devices. This operation occurs in the background and should be executed in advance of when the BCV copy is needed.

```
symmir -g <device_group> establish -full -noprompt
```

Note that the first iteration of the establish needs to be a full synchronization. Subsequent iterations are incremental and do not need the `-full` keyword. Once the command is issued, the array begins the synchronization process using only Symmetrix resources. Since this is asynchronous to the host, the process must be interrogated to see when it is finished. The command to interrogate the synchronization process is:

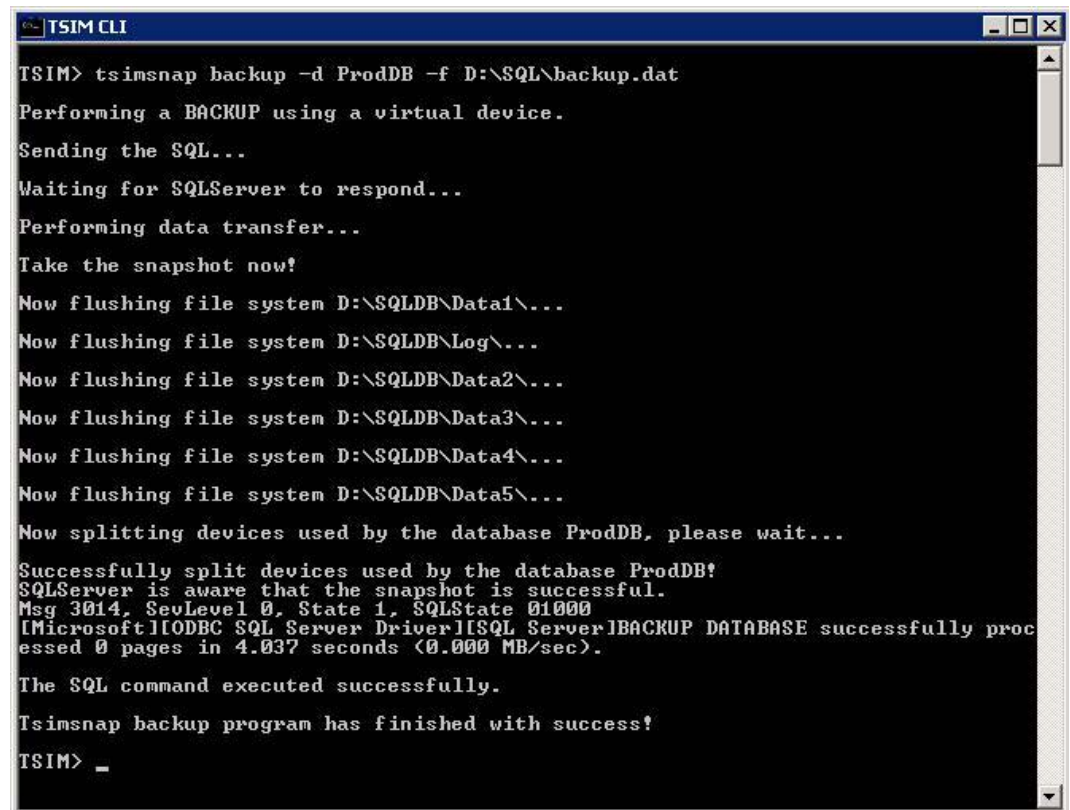
```
symmir -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete.

2. When the volumes are synchronized, the TF/SIM backup operation may be executed:

```
tsimsnap backup -d <device_group> -f <location for metadata>
```

The target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file.



```

TSIM CLI
TSIM> tsimsnap backup -d Proddb -f D:\SQL\backup.dat
Performing a BACKUP using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Take the snapshot now!
Now flushing file system D:\SQLDB\Data1\...
Now flushing file system D:\SQLDB\Log\...
Now flushing file system D:\SQLDB\Data2\...
Now flushing file system D:\SQLDB\Data3\...
Now flushing file system D:\SQLDB\Data4\...
Now flushing file system D:\SQLDB\Data5\...
Now splitting devices used by the database Proddb, please wait...
Successfully split devices used by the database Proddb!
SQLServer is aware that the snapshot is successful.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]BACKUP DATABASE successfully proc
essed 0 pages in 4.037 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap backup program has finished with success!
TSIM> _

```

Figure 4-8 TF/SIM backup execution with TimeFinder/Mirror

In the execution example shown in Figure 4-8, Windows Server mount points are being used to locate the various data and log file locations. Specifically, the mount point root is off the D:\SQLDB directory. Alternatively, locations could be using drive letter assignments.

4.5.2 Using TimeFinder/Clone

TimeFinder/Clone is an EMC software product that copies data internally in the Symmetrix array. A TimeFinder/Clone session is created between a source data volume and a target volume. The target volume needs to be equal to or greater in size than the source volume. The source and target for TimeFinder/Clone sessions can be any hypervolumes in the Symmetrix configuration.

TimeFinder/Clone devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Clone operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix A.

The TimeFinder SQL Server Integration Module utilizes the Symmetrix *Clone Emulation* mode to facilitate backup operations for a SQL Server database. Thus, all TimeFinder/Mirror operations are mapped transparently to TimeFinder/Clone operations. TF/SIM does require that for local operation, the source and target volumes are preestablished. This can be done transparently by setting the appropriate environment variable to enable Clone Emulation where required.

Figure 4-9 depicts the necessary steps to make a VDI backup of a Microsoft SQL Server database onto BCV devices using TimeFinder/Clone.

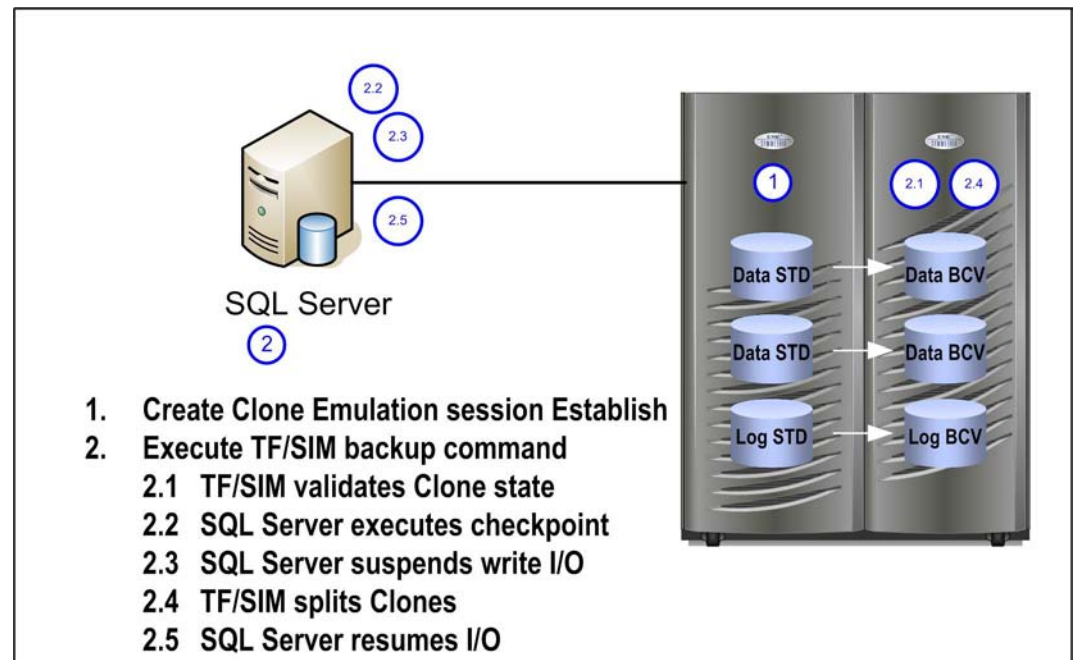


Figure 4-9 Creating a VDI backup of a SQL Server database with TimeFinder/Clone

1. The first action is to create and synchronize the TimeFinder/Clone pairs. The following command creates the TimeFinder/Clone pairings and protection bitmaps:

```
set SYMCLI_CLONE_EMULATION=ENABLED
```

```
symmir -g <device_group> establish -noprompt
```

TF/SIM requires that the clone is fully synchronized with the source volumes. The command to interrogate the precopy process is:

```
symclone -g <device_group> verify
```

This command will return a 0 return code when the synchronization operation is complete.

2. When the volumes are synchronized, the TF/SIM backup operation may be executed:

```
tsimsnap backup -d <database> -f D:\SQL\backup.dat
```

In this example, the target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file.

Databases copied using TimeFinder/Clone are subject to Copy on First Write (COFW) and Copy on Access (COA) penalties. The Copy on First Write penalty means that the first time a track is written to the source volume and it has not yet been copied to the target volume, it must first be copied to the target volume before the write from the host is acknowledged. Subsequent writes to tracks, which have already been copied, do not suffer from the penalty. Copy on Access means that if a track on a target volume is accessed before it has been copied, it must first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems.

4.5.3 Using TimeFinder/Snap

TimeFinder/Snap enables users to create complete copies of their data while consuming only a fraction of the disk space required by the original copy.

TimeFinder/Snap is an EMC software product that maintains space-saving pointer-based copies of disk volumes using virtual devices (VDEVs) and save devices (SAVDEVs). The VDEVs contain pointers either to the source data (when it is unchanged) or to the SAVDEVs (when the data has been changed).

TimeFinder/Snap devices are managed together using SYMCLI device or composite groups. Solutions Enabler commands are executed to create SYMCLI groups for TimeFinder/Snap operations. If the database spans more than one Symmetrix, a composite group is used. Examples of these commands can be found in Appendix B.

Figure 4-10 depicts the necessary steps to make a VDI backup of a SQL Server database using TimeFinder/Snap.

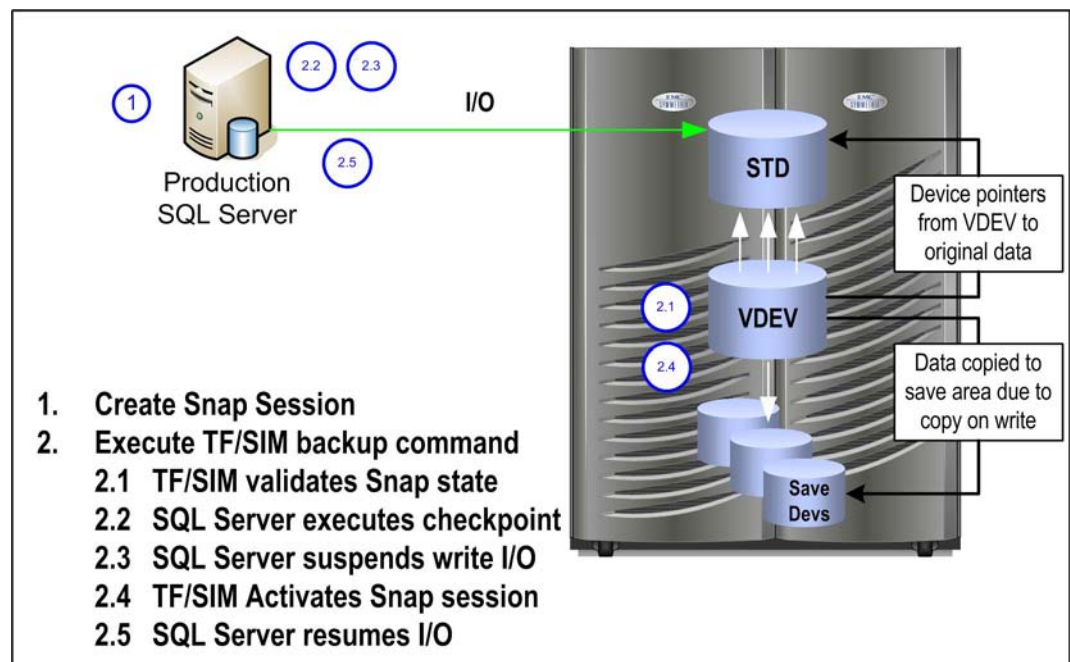


Figure 4-10 Creating a VDI backup of a SQL Server database with TimeFinder/Snap

1. The first action is to create the TimeFinder/Snap pairs. The following command creates the TimeFinder/Snap pairings and protection bitmaps. No data is copied or moved at this time:

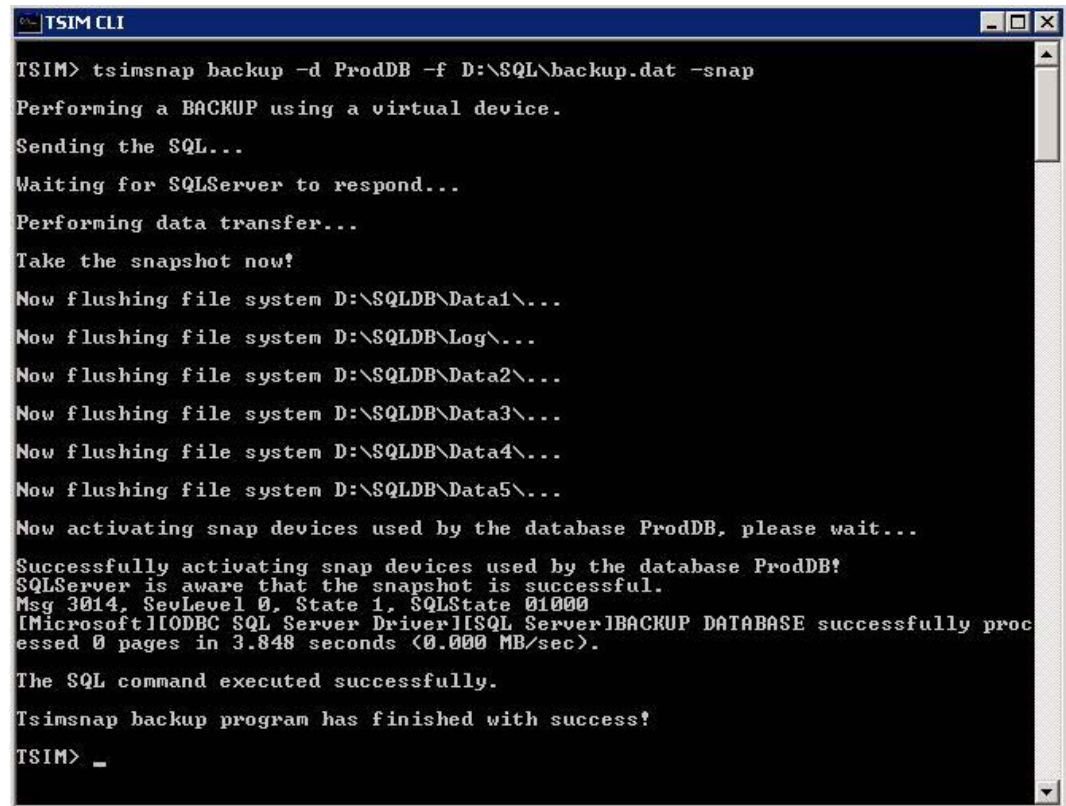
```
symsnap -g <device_group> create -noprompt
```

Unlike TimeFinder/Mirror, the snap relationship is created and activated when it is needed. No prior copying of data is necessary. The create operation establishes the relationship between the standard devices and the virtual devices (VDEVs), and it also creates the protection metadata.

2. After the snap is created, the TF/SIM backup operation may be executed:

```
tsimsnap backup -d <database> -f D:\SQL\backup.dat -snap
```

In this example, the target production database is provided after the `-d` command line option. The `-f` command line option specifies the location where TF/SIM will store the VDI metadata file. A sample execution of this TF/SIM process is shown in Figure Figure 4-11.



```

TSIM CLI
TSIM> tsimsnap backup -d ProdDB -f D:\SQL\backup.dat -snap
Performing a BACKUP using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Take the snapshot now!
Now flushing file system D:\SQLDB\Data1\...
Now flushing file system D:\SQLDB\Log\...
Now flushing file system D:\SQLDB\Data2\...
Now flushing file system D:\SQLDB\Data3\...
Now flushing file system D:\SQLDB\Data4\...
Now flushing file system D:\SQLDB\Data5\...
Now activating snap devices used by the database ProdDB, please wait...
Successfully activating snap devices used by the database ProdDB!
SQLServer is aware that the snapshot is successful.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]BACKUP DATABASE successfully proc
essed 0 pages in 3.848 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap backup program has finished with success!
TSIM> _

```

Figure 4-11 TF/SIM backup execution with TimeFinder/Snap

Databases copied using TimeFinder/Snap are subject to a Copy on First Write (COFW) penalty while the snap is activated. The Copy on First Write penalty means that if a track is written to the source volume and it has not yet been copied to the snap save area, it must first be copied to the save area before the write from the host is acknowledged.

4.6 SYMIOCTL VDI backup

SYMIOCTL is a part of the EMC Storage Resource Management framework, which is a separately licensed EMC product. Implemented as a command line interface, SYMIOCTL does provide the ability to initiate VDI snapshot functionality for SQL Server environments into discrete steps to allow for specific actions to be implemented between “begin” and “end” snapshot commands. This is in contrast to products such as TF/SIM and Replication Manager, which provide structured approaches for managing known styles of execution.

Serious consideration should be given to solutions based on this style of functionality, and great care should be taken with user created scripts. Consider the issues such as the abnormal termination of a user-generated script after the “begin snapshot” and before the “end snapshot”. Such an abnormal termination would leave the target database in a VDI-suspended state and result in possible termination of the database itself by the SQL Server instance.

Products such as TF/SIM and Replication Manager implement functionality in a well structured and controlled manner to ensure that impact to production databases is mitigated.

4.6.1 Using TimeFinder/Mirror

The SYMIOCTL command line utility allows for execution of VDI processing against SQL Server databases in a TimeFinder environment. Unlike TF/SIM, however, all processing of the device group and state of the mirror devices within the group must be managed through command line operations. Since the execution of SYMIOCTL is somewhat independent of any TimeFinder operations, it can simply be executed at the appropriate time to place the database in the required state.

Additionally, there is no default ability to flush the file system buffers (note that in the preceding TF/SIM examples, TF/SIM ensures that file system buffers are also flushed). To implement this functionality with SYMIOCTL, it is necessary to use the Symmetrix Integration Utility (SIU) to perform the flush operation. The flush operation ensures that other file system-level operations are successfully written to the disk.

Therefore, the execution of the SYMIOCTL process is in the following manner, and assumes that the device group has been correctly defined, and that requisite mirror devices (BCVs, clones, or snap VDEVs) have been implemented.

1. The first action is to implement the required disk mirror device state.

For TimeFinder/Mirror devices, the BCVs should be synchronized with the standard volumes utilizing the following TimeFinder command:

```
symmir -g <device_group> establish [-full] -noprompt
```

For TimeFinder/Clone devices, the clone session should be created by using the following TimeFinder command:

```
symclone -g <device_group> create -noprompt
```

For TimeFinder/Snap devices, the snap session should be created by using the following TimeFinder command:

```
symsnap -g <device_group> create -noprompt
```

2. When the volumes are synchronized and/or sessions have been created, the SYMIOCTL backup operation may be executed.

```
symioctl -type SQLServer begin snapshot <database>  
SAVEFILE <location of metafile> -noprompt
```

In this example, the SAVEFILE command line option specifies the location where *SYMIOCTL* will store the VDI metadata file.

3. Flush all database locations, either drive letters or mount points, in use by using SIU. In this example, the locations are mount points.

```
symntctl flush -path <mount point for volume>
```

Execute for each and every database file and log location.

4. Split the devices

For TimeFinder/Mirror:

```
symmir -g <device_group> split -instant -noprompt
```

For TimeFinder/Clone:

```
symclone -g <device_group> activate -noprompt
```

For TimeFinder/Snap:

```
symsnap -g <device_group> activate -noprompt
```

5. Finalize the SYMIOCTL backup command and thaw I/O operations on the SQL Server database.

```
symioctl -type SQLServer end snapshot <database> -noprompt
```

It is the responsibility of the administrator to ensure that the SYMIOCTL end snapshot command is executed within the script—there is no automatic termination of the command. Leaving the database in snapshot state will suspend all user connections and may cause SQL Server to take the database offline. Any user-created process or script must cater for abnormal termination of the process or script itself, and ensure that the end snapshot is executed as a cleanup operation.

An example of the execution of the SYMIOCTL command using TimeFinder/Mirror is shown in Figure 4-12 on page 4-22.

```

C:\>symioctl -type SQLServer begin snapshot ProdDB SAVEFILE D:\SQL\backup.dat -n
oprompt

C:\>symntctl flush -path D:\SQLDB\Data1
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data2
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data3
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data4
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Data5
    Successfully flushed file system cache.

C:\>symntctl flush -path D:\SQLDB\Log
    Successfully flushed file system cache.

C:\>symnir -g sql_prod split -instant -noprompt
'Split' operation execution is in progress for
device group 'sql_prod'. Please wait...
'Split' operation successfully executed for device group
'sql_prod'.

C:\>symioctl -type SQLServer end snapshot ProdDB -noprompt
C:\>_

```

Figure 4-12 Sample SYMIOCTL execution with TF/Mirror

4.7 Replication Manager VDI backup

EMC Replication Manager (RM) can be used to manage and control the TimeFinder copies of a SQL Server database. The RM product has a GUI and command line and provides the capability to:

- ◆ Auto-discover the standard volumes holding the database
- ◆ Identify the path name for all data and transaction log file locations

Using this information, RM can set up TimeFinder groups with BCVs or VDEVs, schedule TimeFinder operations, and manage the creation of database copies, expiring older versions as needed.

Figure 4-13 on page 4-6 demonstrates the steps performed by Replication Manager using TimeFinder/Mirror to create a database copy that can be used for multiple other purposes:

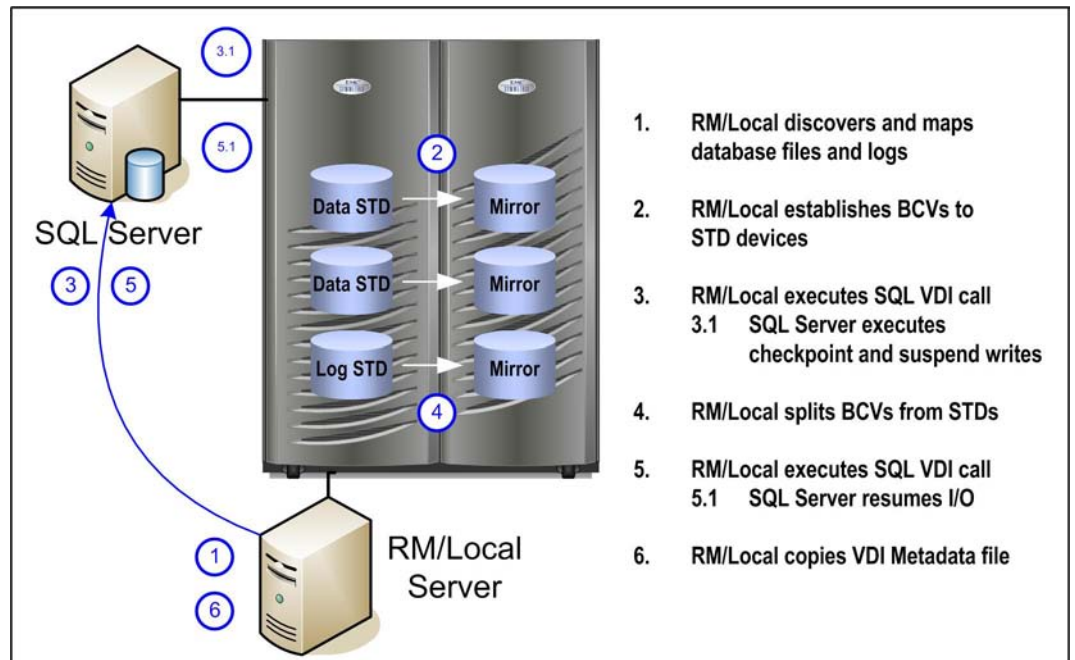


Figure 4-13 Using RM to make a TimeFinder replica of a SQL Server database

1. Replication Manager maps the database locations of all the data files and logs. These Symmetrix standard volumes holding those locations are discovered. Note that the dynamic nature of this activity will handle the situation when extra volumes are added to the database. The procedure will not have to change.
2. Replication Manager then establishes the BCVs to the standard volumes in the Symmetrix. Replication Manager polls the progress of the establish until the BCVs are synchronized and then moves on to the next step.
3. Replication Manager executes a SQL Server VDI database call to checkpoint and write suspend the database.
4. Replication Manager then issues a TimeFinder split, to detach the BCVs from the standard devices.
5. Next, Replication Manager executes another SQL Server VDI call to resume all I/O operations to the database.
6. Finally, Replication Manager copies the VDI metadata file from the source host to an RM holding area for use later in a restore.

4.8 Saving the VDI backup to long-term media

Once the VDI backup has completed, the result is a consistent set of data and transaction log files on the mirror devices. These devices are typically presented to a secondary (or backup) server, where they can be mounted as valid NTFS volumes as either Drive letter locations or mount points, as required.

Note that to retain the validity of the backup state, the files themselves should not be mounted to a SQL Server instance to facilitate any further processing. Instead, the file system objects (the files) themselves should be copied to whatever long-term storage media is required. The backup state represented by the files themselves should be maintained until the backup is required to be restored or processed in some manner.

It is also important to maintain a copy of the VDI metadata file created during the VDI backup process. Each VDI metafile is required to facilitate recovery processes against the volumes containing the VDI backup image. The VDI metafile should be stored in a location from which it is accessible for the host executing the recovery process.

Chapter 5 Restoring and Recovering Microsoft SQL Server Databases

This chapter presents these topics:

5.1	SQL Server restore functionality	5-2
5.2	EMC Products for SQL Server recovery	5-6
5.3	EMC Consistency technology and restore.....	5-7
5.4	TF/SIM VDI restore	5-9
5.5	SYMIOCTL VDI restore.....	5-28
5.6	Replication Manager VDI restore.....	5-37
5.7	Applying logs up to timestamps or marked transactions	5-38

Recovering a production database is an event that all DBAs hope is never required. Nevertheless, database administrators must be prepared for unforeseen events such as media failures or user errors, which require database recovery operations. The keys to a successful database recovery include:

- ◆ Identifying database recovery time objectives
- ◆ Planning the appropriate recovery strategy based upon the backup type (full, incremental)
- ◆ Documenting the recovery procedures
- ◆ Validating the recovery process

Microsoft SQL Server database recovery depends upon the back-up methodology used. With the appropriate backup procedures in place, a SQL Server database can be recovered to any point in time between the end of the full backup and the point of failure using a combination of the full back up data and log files, as well as those recovery structures including any incremental transaction log backups.

Recovery typically involves copying the previously backed up files into their appropriate locations and, if necessary, performing recovery operations to ensure that the database is recovered to the appropriate point in time and to ensure that the database is consistent.

To mitigate the amount of data loss sustained to a production database, the first step is to ensure that a complete chain of database backups and incremental log backups are maintained. To this end, the first process before any restoration procedures to a production system is to create an incremental backup of the current transaction log state. This is typically the last log sequence available, and will be required to minimize data loss.

This section assumes that EMC technology has been used in the backup process as described in Chapter 4.

5.1 SQL Server restore functionality

Microsoft SQL Server provides a number of standard tools and interfaces to facilitate restore processing. Within the standard product, administrators can use the built-in backup tools presented by both the SQL Server Enterprise Manager and the SQL Server T-SQL programmatic interface.

SQL Server Enterprise Manager provides a graphical user interface (GUI) to the major administrative functions within the database environment. In Figure 5-1 on page 5-3, an example of the interface to the restore functionality is provided.

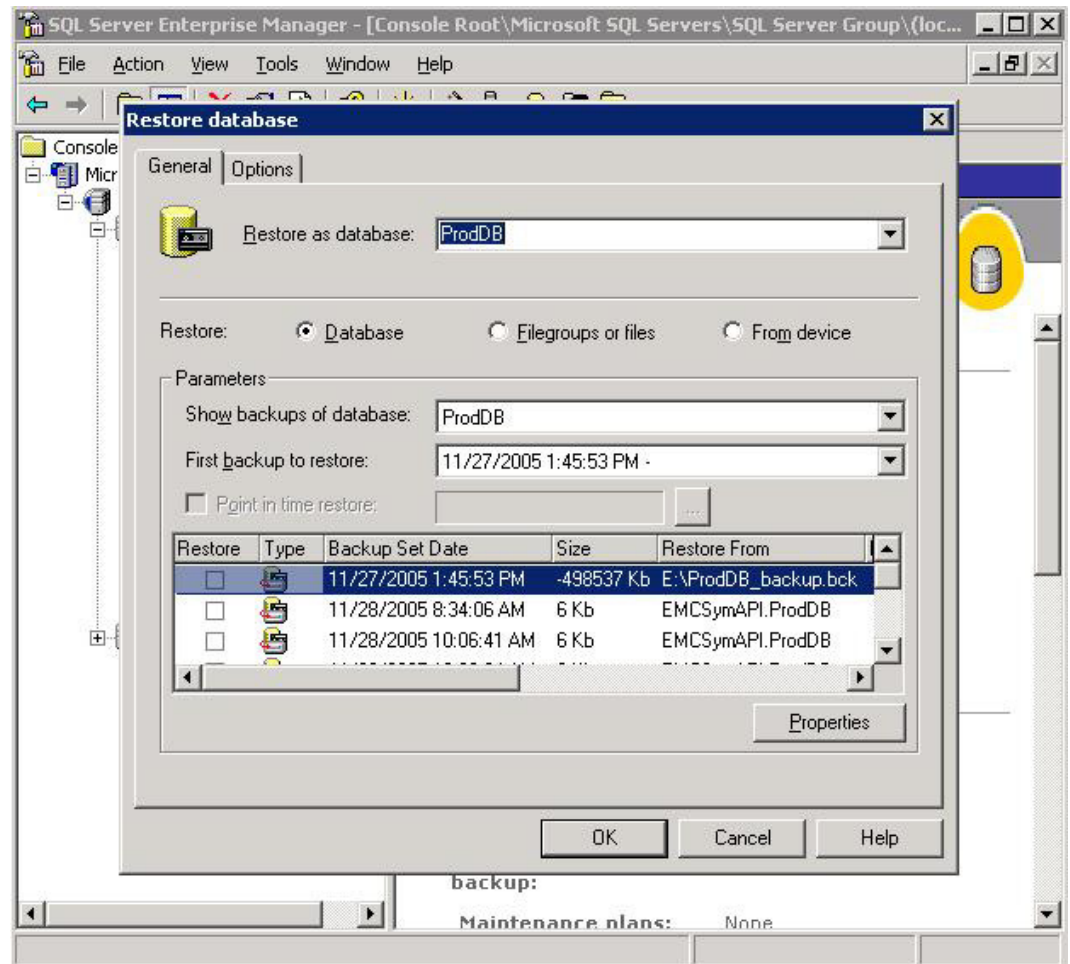


Figure 5-1 SQL Enterprise Manager restore interface

Restore operations do present a significantly larger array of alternatives than backup operations. These options are available to implement such functions as partial restore operations, for example, filegroups within the database. It is also possible, in the case of restoring a full database backup, to relocate the files during the restore operation. Additionally, since a full database backup may have subsequently facilitated incremental transaction log backups, options are provided to allow for subsequent restoration of these transaction log backups against the recovered database to mitigate data loss. Figure 5-2 on page 5-4, shows the “Options” tab of the restore window within SQL Server Enterprise Manager, displaying these abilities.

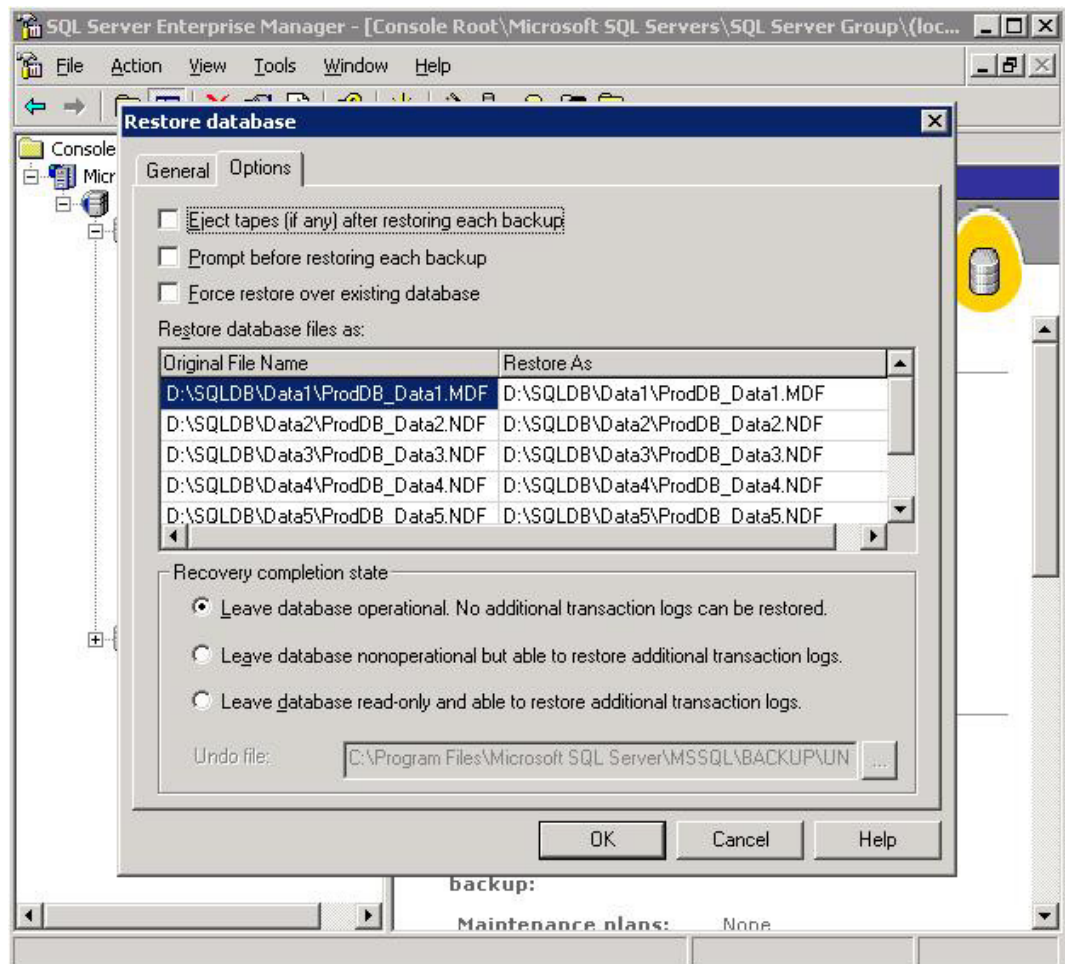


Figure 5-2 Additional Enterprise Manager restore options

Of special note in Figure 5-2 is the Recovery Completion State. These completion states facilitate the ability to apply subsequent incremental transaction logs to the restored database. These options are described next.

5.1.1 SQL Server – RESTORE WITH RECOVERY

This is the “Leave database operational” option shown in Figure 5-2.

When the WITH RECOVERY keywords are used during a SQL Server restore operation, this implies that no subsequent transaction log backups are to be applied to this restored database. SQL Server will then execute internal roll forward and roll back operations to ensure a transactionally consistent state of the database. Once completed, the database will be placed in a standard online state.

5.1.2 SQL Server – RESTORE WITH NORECOVERY

This is the “Leave database nonoperational” option shown in Figure 5-2 on page 5-4.

When the WITH NORECOVERY keywords are used during a SQL Server restore operations, this implies that subsequent incremental transaction logs will be applied to the database being restored. Therefore, at the end of the database restore, or subsequent incremental transaction log restores, SQL Server will not carry out any rollback recovery operations. Any database which is placed in a NORECOVERY state is not available for any user access. Finally, any sequence of restore operations executed with NORECOVERY will be concluded with a RESTORE ... WITH RECOVERY statement. This will then indicate that the process outlined in section 5.1.1 be initiated.

5.1.3 SQL Server – RESTORE WITH STANDBY

This is the “Leave database read-only” option shown in Figure 5-2 on page 5-4.

Unlike the NORECOVERY process, which does not allow for access to the database being restored, the STANDBY mode implements additional steps to allow for read only access to the database during those times when incremental transaction logs are not being applied. The RESTORE LOG WITH STANDBY does require file locations in which SQL Server may store those pages whose state has been rolled back because the transaction which changed the data has not yet executed a commit as represented by the last log. The database state at the end of this “undo” process is one of a transactionally consistent point in time. Users can read from the database while logs are not being applied. At the start of the application of a subsequent incremental transaction log, the saved page states are restored from the standby files, and the subsequent incremental transaction log backup is applied. This cycle proceeds until such time as a WITH RECOVERY clause is executed. At that point, SQL Server will initiate the process details in section 5.1.1.

The goal of the NORECOVERY and STANDBY recovery modes is to limit any potential loss of work (or data) between the time at which the full backup occurred and the time of the database failure which necessitated the restore. Clearly, the STANDBY mode of operation does allow for additional functionality.

The Transact-SQL statement RESTORE DATABASE facilitates all of these operations. The full syntax for the RESTORE DATABASE statement can be found in the Microsoft SQL Server Books Online documentation. Only a subset of the available options will be used in this chapter for illustration purposes.

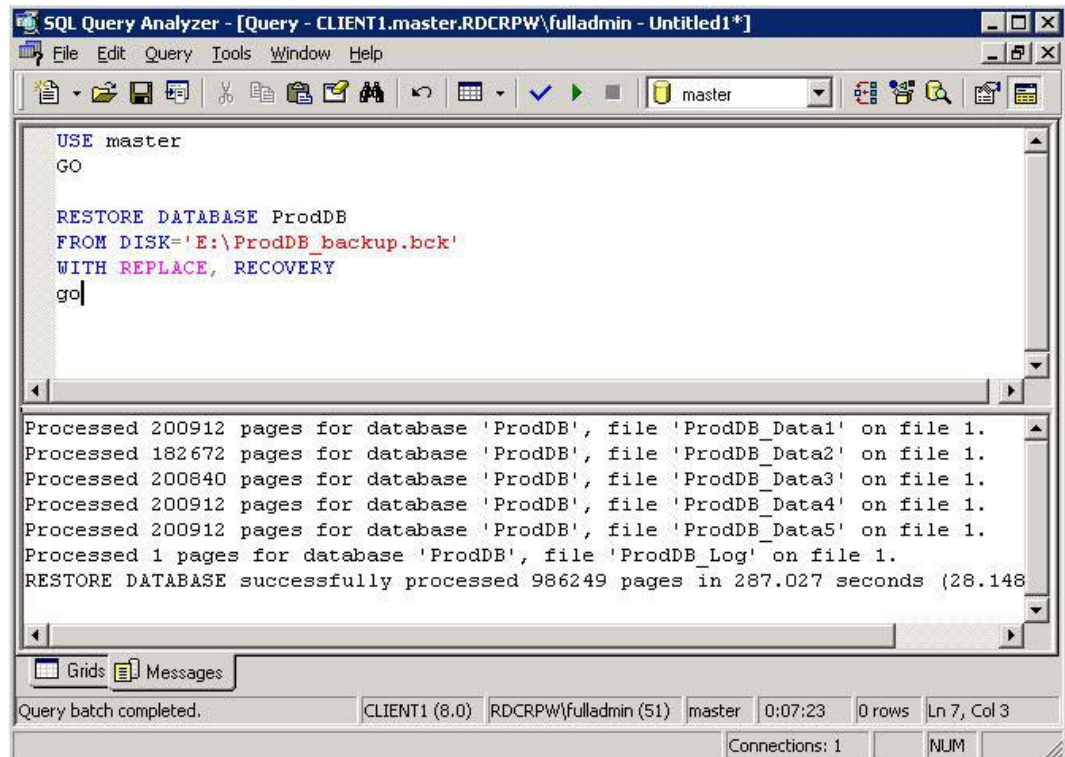


Figure 5-3 DATABASE RESTORE Transact-SQL execution

Generally, a restore operation will necessitate the SQL Server instance taking the database offline during the restore process, such as that process shown in Figure 5-3. As of SQL Server 2005, certain restore operations may be executed while the database remains online, though typically any given filegroup may need to be taken offline while the remainder of the database is accessible. Clearly, taking a production database offline will impact service-level agreements which may be in place. In all cases, the primary concern is to mitigate any impact to the production system. This includes facilitating the fastest possible restoration process through to reducing any potential data loss exposure.

In almost all cases, disk mirror operations which facilitate a restore process will be significantly more expeditious than a comparable streaming restore operation, be that from a disk based backup file as in the example above, or from streaming tape. It is for this single reason that in general, disk mirror backup operations become appealing. The ability to almost instantaneously restore a corrupted database, and return it to an operational state is much more appealing than requiring to wait until a streaming restore has been streamed from disk or tape. Until the restore is complete and any applicable incremental transaction log backups are replayed, the production database is considered offline.

5.2 EMC products for SQL Server recovery

Disk mirror-based functionality of storage arrays can significantly decrease the amount of time required to execute restore operations. As documented within Chapter 4, EMC provides a number of products that can be used to facilitate restore operations.

EMC products that facilitate SQL Server backup operations currently utilize the SQL Server VDI and ultimately will also provide support for the VSS operations made available with Microsoft SQL Server 2005.

Because disk mirror restore operations will necessitate the reverse synchronization of the mirror device back to the standard devices, a number of issues need to be addressed. Restoration of this type at the LUN level cannot be an online operation. The LUN must be dismounted before the reverse synchronization (restore). This is to ensure that Windows does not retain a prereset view of the specific LUNs. Should a restore be executed with the LUN mounted, Windows will mark the volume as questionable, and possible data corruption may result. In all circumstances, it is necessary to follow the guidelines provided by the specific EMC product being used to execute the restore.

In the case of Microsoft Cluster configurations, this restoration may become more complex because of the dependencies built into the Cluster Service itself. In general, disk resources are one of the dependencies for the SQL Server instance resource. To facilitate a restore operation, the disk resources will need to be taken offline before the TimeFinder restore operation. Failure to do so may result in Cluster Service inaccurately detecting a disk resource failure and commencing a failover operation. However, before taking the disk resources offline, it would be necessary to delete the database targeted for restore.

5.3 EMC Consistency technology and restore

EMC provides a number of solutions built around a foundation of Consistency technology. Using this technology, it is possible to create valid restartable images of a SQL Server database environment using dependent-write principles. These styles of solutions are described in Chapter 3.

It is entirely possible to utilize the images created through Consistency technology to facilitate the ability to create valid restart points. It is also possible to stream these images to tape or other media for longer term storage and subsequent restoration. However, while these images represent valid restart points, they do not facilitate any of the SQL Server recovery forms such as NORECOVERY or STANDBY, which are described in the remainder of this chapter. That is, you are not able to apply subsequent incremental transaction logs to a restartable image, only to a recoverable one.

Restartable images created using Consistency technology do have valuable usage when customers are concerned with federated environments, where the creation of a consistent restart point for all databases and other objects within the environment is the primary goal. Independently created recoverable images created of each individual component of a federated environment make it inordinately complex, if not impossible, to resolve a single business point of consistency across all systems.

The Recovery Point Objective (RPO) of a restartable image is based on a static point in time, and will increase as changes to the source production environment proceed. It is only when the next point of consistency is created that the RPO can be reset to its initial state. During intervening times between cycles, the RPO point will vary, it is important to control the cycle of such a process to ensure that the solution falls within the maximum RPO.

The RTO of such Federated environments is significantly easier to manage, since they typically only require restart processes, which have minimal administrative overheads. This combination of controlled RPO and very low RTO when considered across the scope of all systems within the federation makes an exceptionally compelling business restart solution.

Restore operations for images created by Consistency Technology require five major steps:

1. Drop or detach the production database using `sp_detach_db`
2. Unmount the filesystems used as locations for the database data and log files.
3. Use the appropriate restore operation for the Mirror environment be utilized
4. Mount the volumes
5. Reattach the database files using `sp_attach_db`.

The `sp_attach_db` stored procedure looks similar to this:

```
sp_attach_db <database>,  
@filename1='<file_location>',  
@filename<n>='<file_location>'
```

Where `<file_location>` represents the locations of all database data files and logs for the database copy, typically these will be the same location from which the original consistent split was taken. Please refer to SQL Server Books On Line for complete documentation on this Transact-SQL stored procedure.

In the following example, a Consistent Split image is attached to the SQL Server instance. It assumes that all the mirrored devices, be they BCVs, Clones or Snaps are restored appropriately to the STD devices, and re-mounted in appropriate locations. If the mount locations are identical to those on the production instance, then it is only necessary to specify the first file location (the SQL Server Metadata file for the database). Since the Metadata file contains information for the locations of all subsequent files and logs, the stored procedure will use this information. If the file locations have been modified, then all the new locations must be specified.

SQL Server will perform necessary roll forward/roll back recovery on the database instance. Since the Write Ahead Logging functionality of SQL Server has been maintained, the resulting cloned instance will return to a transactional consistent state, and become a fully independent clone of the production instance. Note in the following example of Figure 5-4, that transaction roll forward and roll back operations are shown during the attach process as viewed through Query Analyzer. In this instance, 17 transactions were rolled forward, and 2 were rolled back.

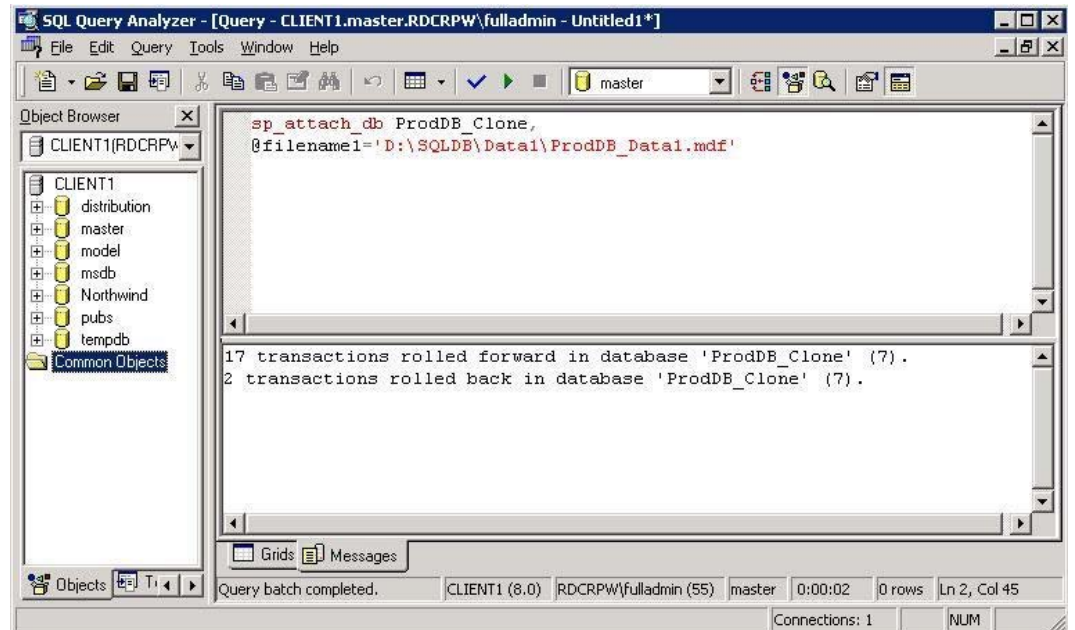


Figure 5-4 Attaching a Consistent split image to SQL Server

5.4 TF/SIM VDI restore

The following examples assume that a restoration is being executed for the production server, to restore a full backup image to its original location.

Prior to restoring any production database on the production server it is always recommended to maintain an existing copy of the database where possible, and to make every effort to create a final transaction log backup.

It will be necessary to restore the VDI metadata file created as a result of the backup operation so that it too is accessible for the TF/SIM command execution.

5.4.1 Using TimeFinder/Mirror

The diagram in Figure 5-5 depicts the necessary steps to execute a VDI restore of a Microsoft SQL Server database using TimeFinder/Mirror.

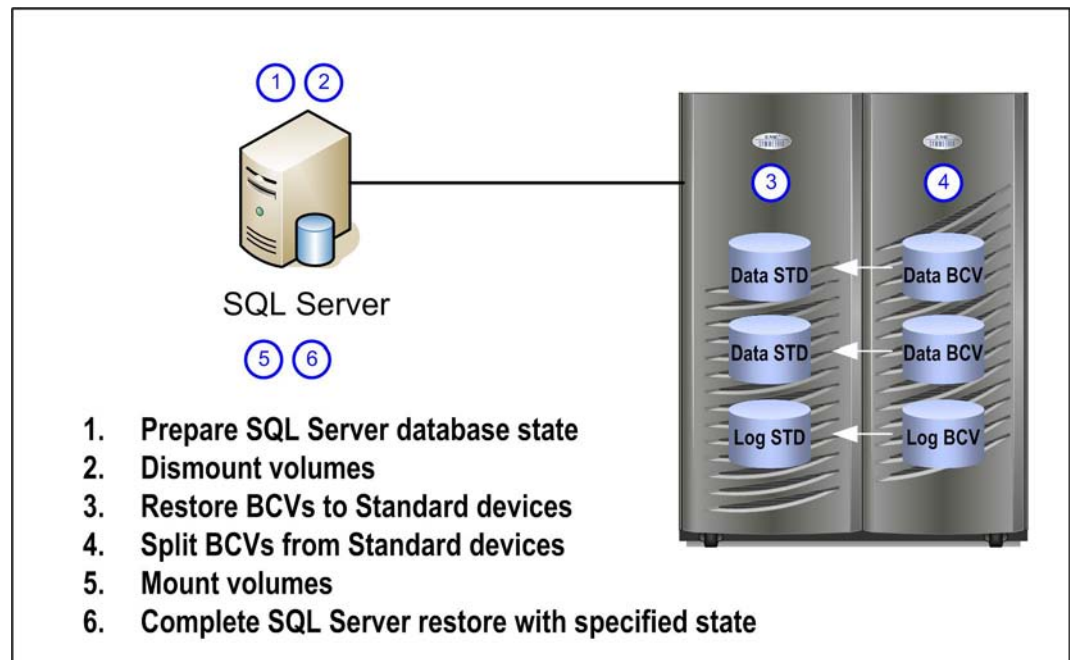


Figure 5-5 TF/SIM restore process using TimeFinder/Mirror

1. Communication is initiated with the SQL Server instance, and a restore operation is executed for the specified database.
2. The volumes or mount points are dismounted on the production host in preparation of the disk mirror restore operation.
3. The disk mirrors are incrementally restored to the standard devices. Only those track which were modified on the standard volumes will be copied back from the BCV devices
4. Once all changes have been restored, the BCV devices are spit from the standards to protect the state of the image on the BCVs
5. The volumes are remounted on the production hosts
6. SQL Server then processes the final restore operation and places the database in the appropriate state as defined on the initial call.

The specified state for the TF/SIM restore operation is one of NORECOVERY, STANDBY or RECOVERY. The option selected will depend on the specific requirements for the restore process. The options are detailed in the following sections.

5.4.1.1 Restore with NORECOVERY

In general, the NORECOVERY mode is utilized in the instance where a chain of logs is known to follow a full database backup. This allows for the restoration of the data and transaction logs of the database, and then the subsequent application of transaction log backups. It is possible to restore a backup and not require the application of transaction logs, this is the RECOVERY option, which is discussed in a subsequent section. The database that is being restored is not available for user processes until such time as the backup procedures and any log applications have been completed.

The command to restore a database instance back to the production server using TF/SIM VDI processing is:

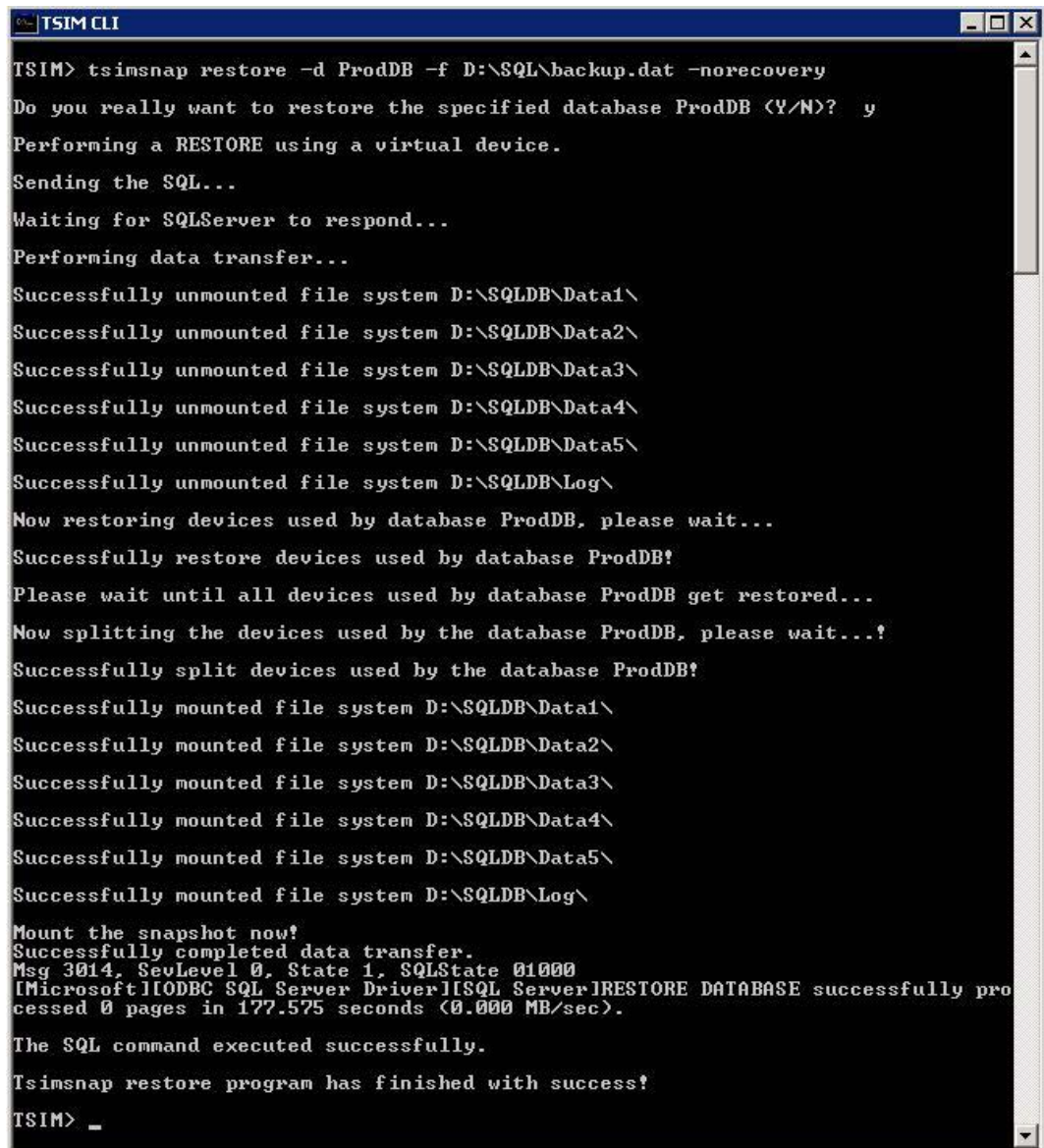
```
tsimsnap restore -d <DB_alias> -f <metafile> -norecovery
```

This process results in the BCV being restored to the state created by the backup operation last executed for the BCV. Additionally, TF/SIM manages a number of other processes, including the unmount operation of the LUN devices used by the database data and log files. The execution of this command is shown in Figure 5-6 on page 5-12.

It is also possible to use the `-protbvcvrest` option the TF/SIM call to optimize the restore operation. With the use of this option, the BCV restore operation begins as a background task and access to the restored state on the STD is immediately available. If a host process attempts a reference to a track marked to be restored from the BCV, the track is copied immediately, and returned to the calling process. Moreover, updates do not flow to the BCV devices. In this way, the validity of the BCV backup image is maintained. Once completely restored, the BCV devices may be split from the STDs.

Refer to the Product Guide for the TF/SIM product for preconditions and usage of the `-protbvcvrest` option. Details on the location of the Product Guide are provided in Appendix A.

To maintain the backup validity of the BCV image when the `-protbvcvrest` option is not used or is not available, the BCV devices are split after all required tracks are restored and before the mount operation. This ensures that the fidelity of the backup image on the BCV devices is maintained.



```

TSIM CLI
TSIM> tsimsnap restore -d ProdDB -f D:\SQL\backup.dat -norecovery
Do you really want to restore the specified database ProdDB (Y/N)? y
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Successfully unmounted file system D:\SQLDB\Data1\
Successfully unmounted file system D:\SQLDB\Data2\
Successfully unmounted file system D:\SQLDB\Data3\
Successfully unmounted file system D:\SQLDB\Data4\
Successfully unmounted file system D:\SQLDB\Data5\
Successfully unmounted file system D:\SQLDB\Log\
Now restoring devices used by database ProdDB, please wait...
Successfully restore devices used by database ProdDB!
Please wait until all devices used by database ProdDB get restored...
Now splitting the devices used by the database ProdDB, please wait...!
Successfully split devices used by the database ProdDB!
Successfully mounted file system D:\SQLDB\Data1\
Successfully mounted file system D:\SQLDB\Data2\
Successfully mounted file system D:\SQLDB\Data3\
Successfully mounted file system D:\SQLDB\Data4\
Successfully mounted file system D:\SQLDB\Data5\
Successfully mounted file system D:\SQLDB\Log\
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 177.575 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _

```

Figure 5-6 TF/SIM restore production database with TimeFinder/Mirror - NORECOVERY

The resulting state of the database after such an operation is one of a **LOADING** database. A **LOADING** database is not available for access directly, and is therefore different to a read-only state as created by a process such as a **STANDBY**. The state of the database does allow for subsequent transaction log backup files to be restored into the database instance. This application of incremental log backups allows for the minimization of data loss between the time the original database backup occurred and the time at which the failure occurred, which necessitated the restore operation.

The database state can easily be seen by using SQL Server Enterprise Manager, as shown in Figure 5-7.

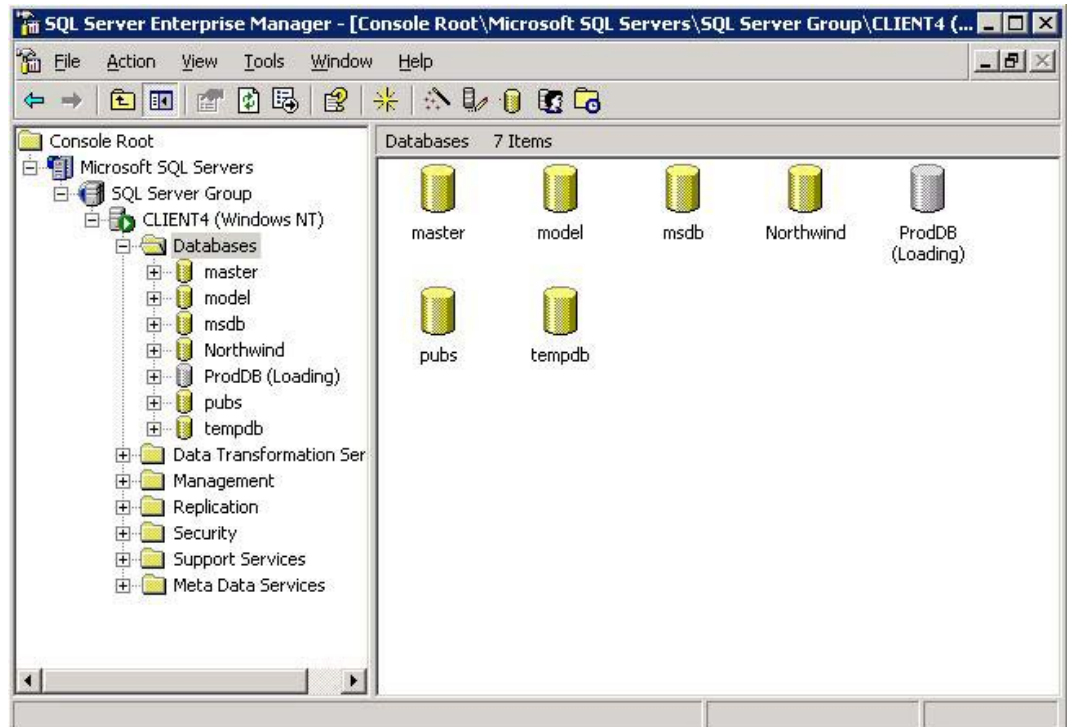


Figure 5-7 SQL Enterprise Manager view of a LOADING database

Transaction log backups need to be applied to this database in sequence, specifically in the order in which they were created. The time required to apply the incremental transaction log backups will relate directly to the amount of change which occurred on the production database during normal operations.

Clearly, a restoration of the production database in this manner will necessitate an outage. While much of time requirement may be mitigated by using disk mirror technology to restore the last full backup image, the incremental transaction logs will still be require to be replayed against the restored database. The time required to restore the database and apply transaction logs is referred to as the Recovery Time Interval. Figure 5-8 on page 5-14 demonstrates the Transact-SQL statement execution for the application of an incremental transaction log backup. The NORECOVERY option used indicates that further logs may be applied to this restoring database.

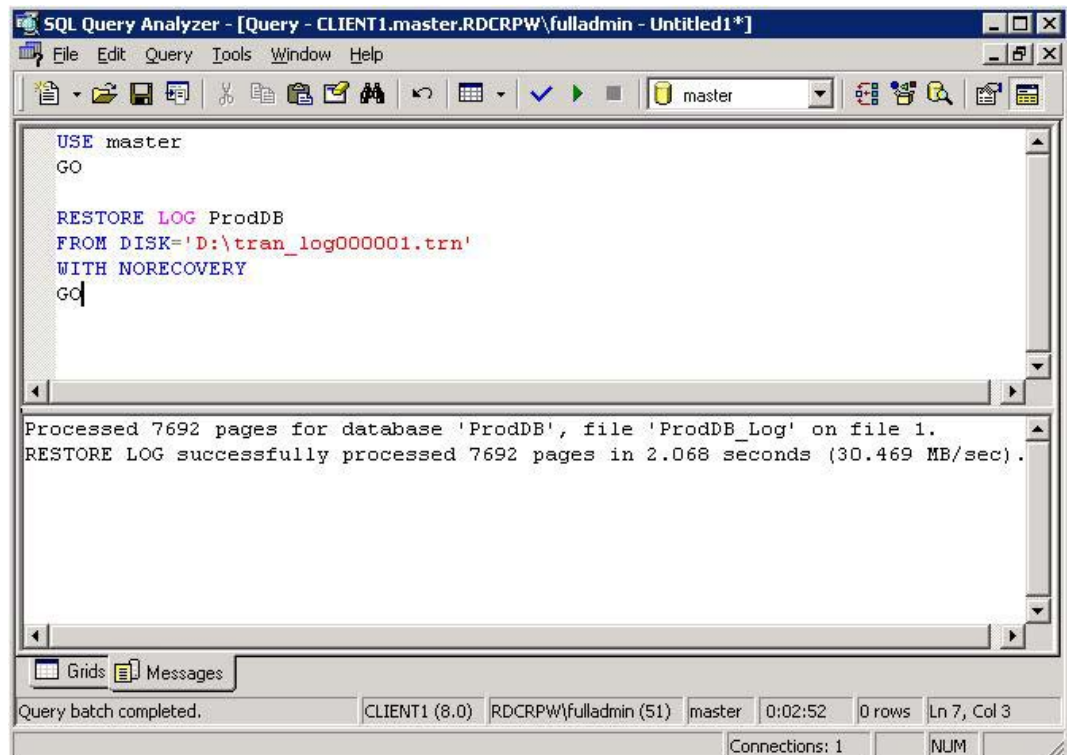


Figure 5-8 Restore of incremental transaction log with NORECOVERY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, please refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

5.4.1.2 Restore with STANDBY

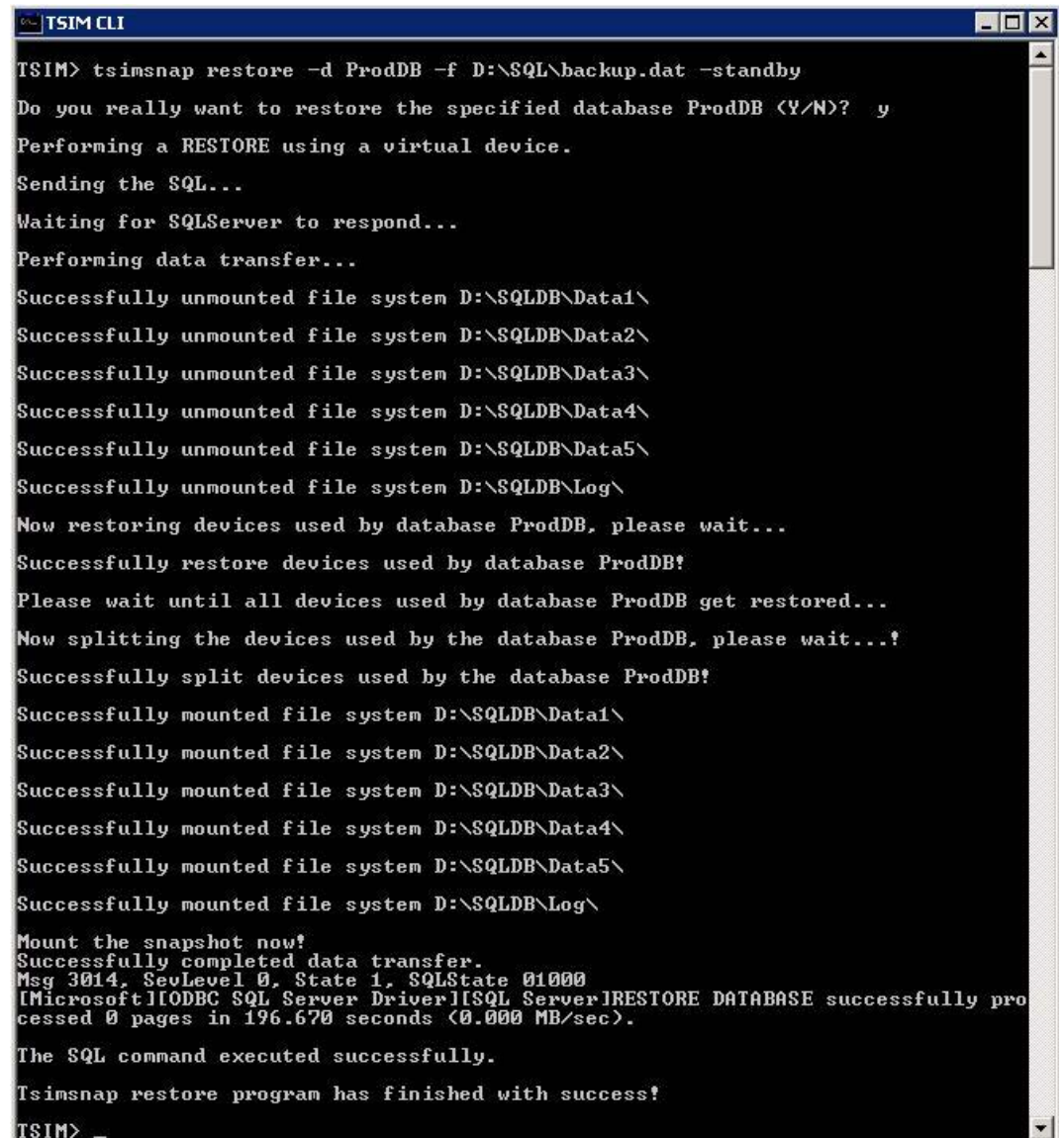
The STANDBY mode is not a typical mode used for restoration of a production database. Typically, either the NORECOVERY or RECOVERY modes are used because they result in the required states. However, the STANDBY mode of recovery does allow for the creation of a read-only state and it is possible that a database

administrator may want to query the state of the database during intervals between incremental transactions log applications. For that purpose, it is discussed here.

To facilitate the application of incremental transaction logs in a similar manner to that when the NORECOVERY option is utilized, but also to provide read only access to the restored database, the STANDBY option may be used. In this way, the restored database is available for those operations that do not attempt to update the target database. Any attempt to execute updates will return a failure, as the database is in a read-only state.

The command to restore a database instance back to the production server using TF/SIM VDI processing, as shown in Figure 5-9, is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -standby
```



```

TSIM CLI
TSIM> tsimsnap restore -d ProdDB -f D:\SQL\backup.dat -standby
Do you really want to restore the specified database ProdDB (Y/N)? y
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Successfully unmounted file system D:\SQLDB\Data1\
Successfully unmounted file system D:\SQLDB\Data2\
Successfully unmounted file system D:\SQLDB\Data3\
Successfully unmounted file system D:\SQLDB\Data4\
Successfully unmounted file system D:\SQLDB\Data5\
Successfully unmounted file system D:\SQLDB\Log\
Now restoring devices used by database ProdDB, please wait...
Successfully restore devices used by database ProdDB!
Please wait until all devices used by database ProdDB get restored...
Now splitting the devices used by the database ProdDB, please wait...!
Successfully split devices used by the database ProdDB!
Successfully mounted file system D:\SQLDB\Data1\
Successfully mounted file system D:\SQLDB\Data2\
Successfully mounted file system D:\SQLDB\Data3\
Successfully mounted file system D:\SQLDB\Data4\
Successfully mounted file system D:\SQLDB\Data5\
Successfully mounted file system D:\SQLDB\Log\
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 196.670 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM>

```

Figure 5-9 TF/SIM restore production database with TimeFinder/Mirror - STANDBY

During the application of incremental transaction logs, it is necessary, however, to terminate any access from other client connections. Exclusive access to the database is required to be able to apply an incremental transaction log created from the production system.

The STANDBY state leaves the target database in a read-only state, which is shown through SQL Server Enterprise Manager in Figure 5-10.

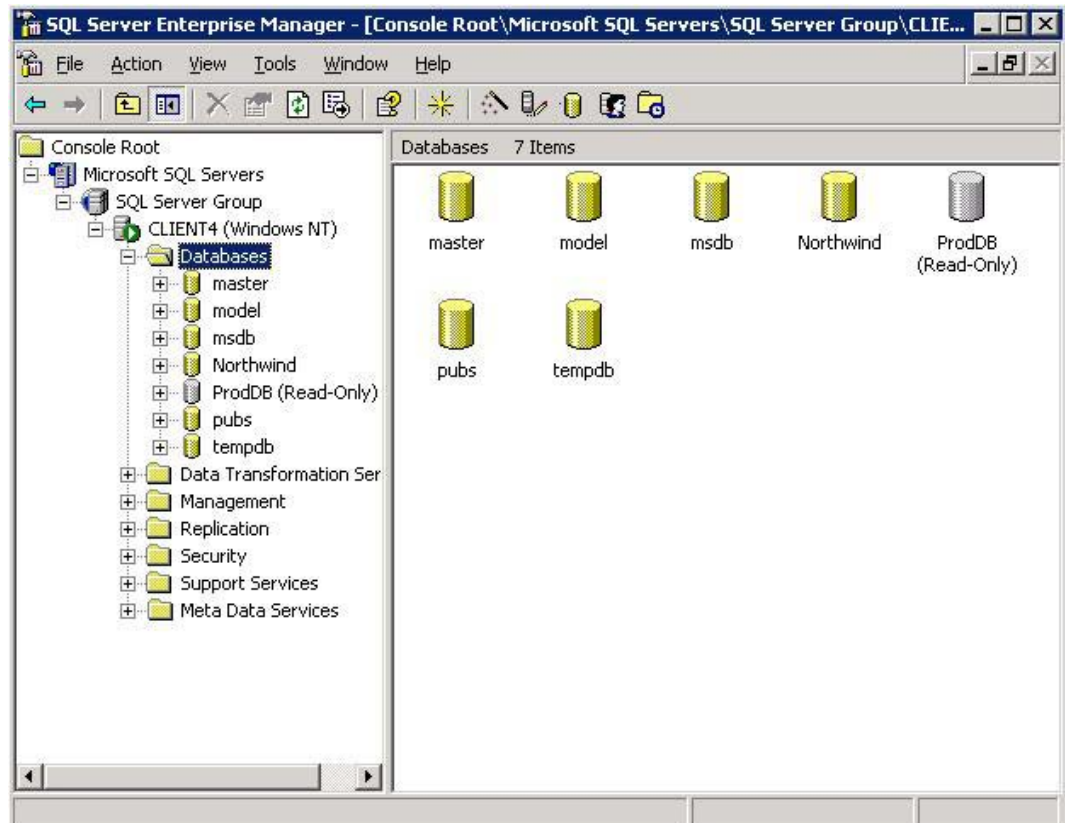


Figure 5-10 SQL Enterprise Manager view of a STANDBY (read-only) database

Similar to the application of incremental transaction logs to a NORECOVERY state, incremental transaction logs may be applied to the STANDBY database. However, to maintain the STANDBY state it is also necessary to specify the STANDBY option on the Transact-SQL RESTORE LOG statement. It is also required that the location for an undo file is provided. The undo file is used to record those pages rolled back for uncommitted transactions, after the application of a transaction log. The roll back operation ensures that a transactionally consistent view is available of the database state.

Before the application of subsequent incremental transaction logs, the undo file changes are reapplied to the database. Figure 5-11 on page 5-17 demonstrates the use of the STANDBY option and the inclusion of the standby file location.

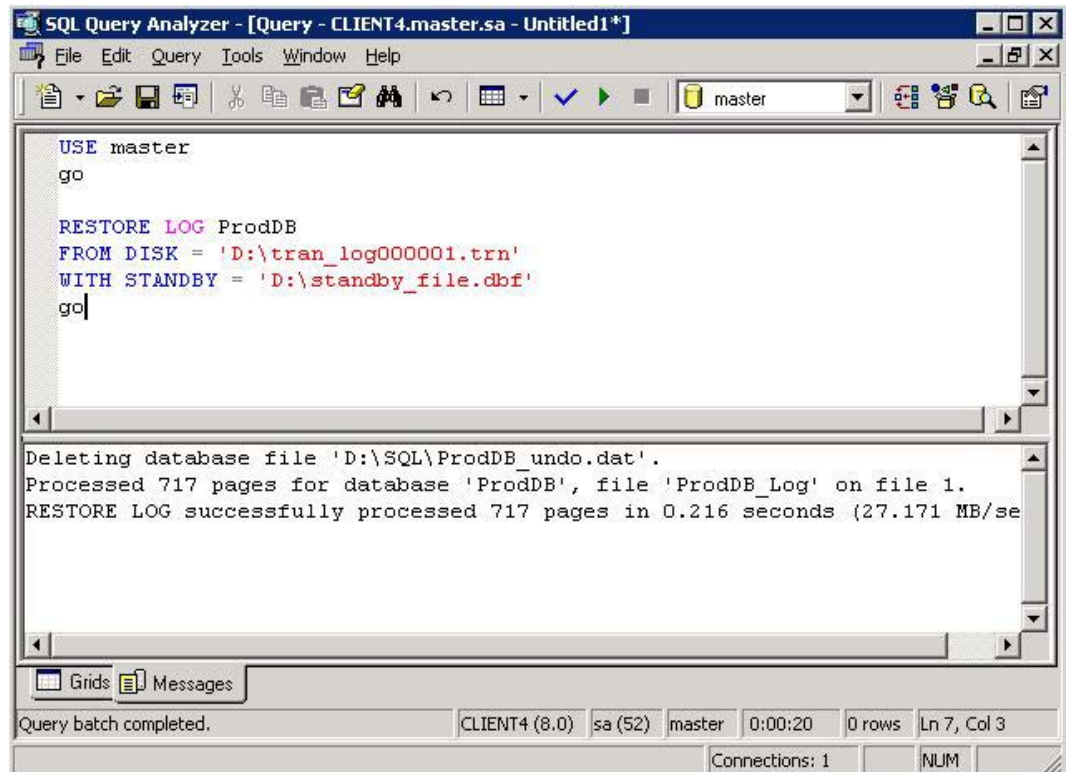


Figure 5-11 Restore of incremental transaction log with STANDBY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database in this state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

5.4.1.3 Restore with RECOVERY

In situations where there are no incremental transaction log backups available for replay against the full database backup, it may be appropriate to simply restore the last full backup and allow SQL Server to bring the database online. In this case, the end state of the operation will be of an online, fully accessible database. No additional incremental

transaction logs can be applied to this database instance once recovered, as a new chain of transaction logs will be initiated.

This process is identical to that executed in section 5.4.1.1, except for the use of the RECOVERY option.

The command to restore a database instance back to the production server using TF/SIM VDI processing is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -recovery
```

5.4.2 Using TimeFinder/Clone

The TF/SIM utility only implements Clone Emulation mode, subsequently restore operations are processed in each the same manner as those detailed in section 5.4.1.

5.4.3 Using TimeFinder/Snap

The execution of TF/SIM restore operations from TimeFinder/Snap devices differs slightly in the manner of execution from that of TimeFinder/Mirror. Once the TimeFinder/Snap restore process is started, the STD devices become available. A restore session is then created for this process, and the original copy session remains intact. Any changed tracks recorded since the start of the TF/Snap session are restored to the STD devices.

Figure 5-12 depicts the necessary steps to make a VDI backup of a Microsoft SQL Server database using TimeFinder/Snap.

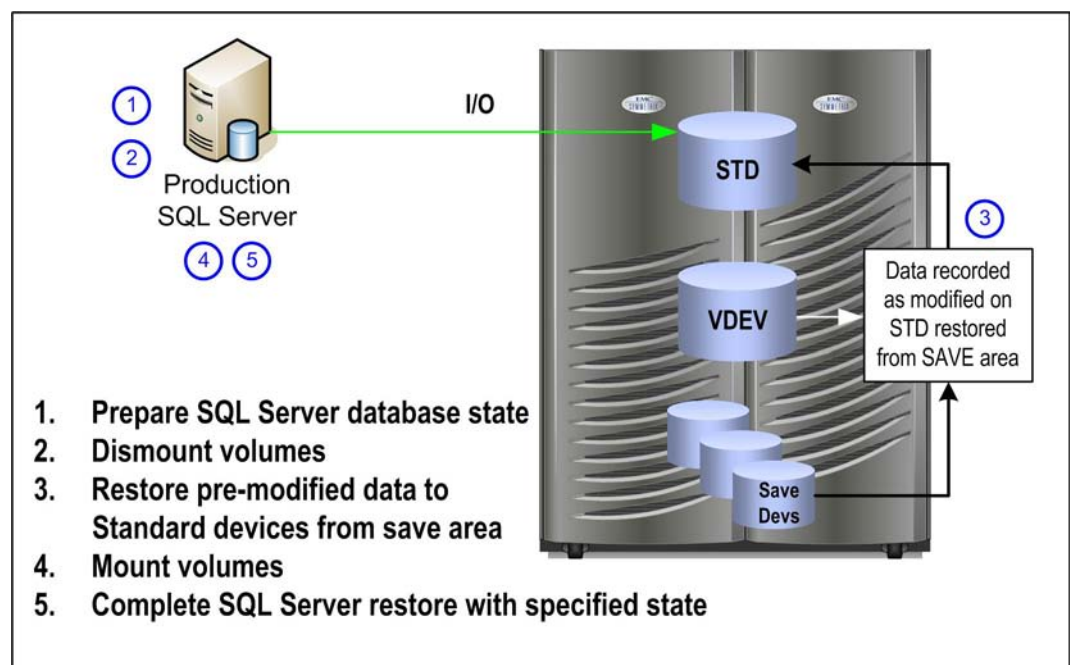


Figure 5-12 TF/SIM restore using TimeFinder/Snap

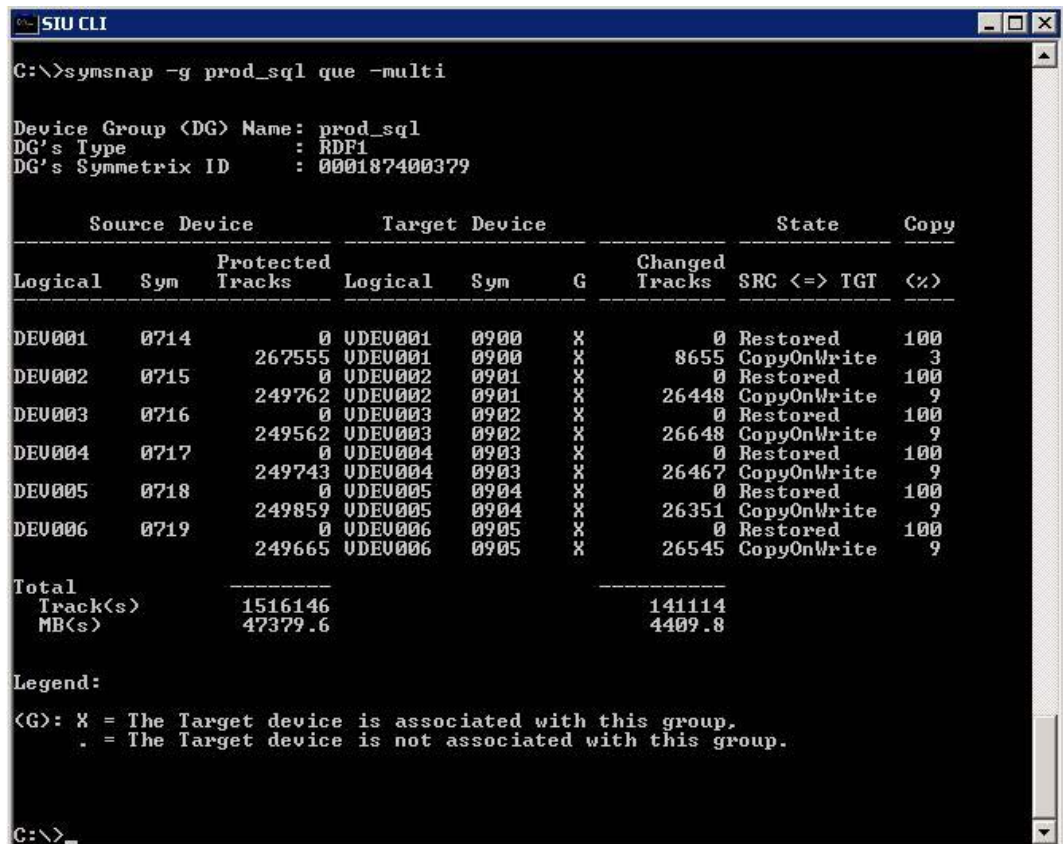
1. Communication is initiated with the SQL Server instance, and a restore operation is executed for the specified database
2. The volumes or mount points are dismounted on the production host in preparation of the disk mirror restore operation.
3. The restore process is initiated by the creation of a new session, which makes the view of the restored state immediately available.
4. The volumes are remounted on the production hosts
5. SQL Server then processes the final restore operation and places the database in the appropriate state as defined on the initial call.

After the execution of the TF/SIM restore operation using TimeFinder/Snap, the database will become immediately available in the state which was designated. This immediate access is provided while the actual copy of tracks is processed as a background operation. The state of the STD will be seen to be that state at the time of TimeFinder/Snap activation. If data is referenced on a track which has not yet been restored from the SAVE area, it will be immediately restored, and then the data will become accessible to the host. Validity of the state is protected and will be consistent.

In actuality, TimeFinder/Snap creates a new restore session to revert the changed tracks. It is possible to view the two states using the following command:

```
symsnap -g <device_group> query -multi
```

The `-multi` option lists all device sessions. This is shown in Figure 5-13.



```

C:\>symsnap -g prod_sql que -multi

Device Group <DG> Name: prod_sql
DG's Type           : RDF1
DG's Symmetrix ID    : 000187400379

  Source Device      Target Device      State      Copy
-----
Logical  Sym  Protected Logical  Sym  G  Changed
Tracks
DEU001   0714    0  UDEV001  0900  X    0  Restored    100
      267555  UDEV001  0900  X    8655 CopyOnWrite    3
DEU002   0715    0  UDEV002  0901  X    0  Restored    100
      249762  UDEV002  0901  X   26448 CopyOnWrite    9
DEU003   0716    0  UDEV003  0902  X    0  Restored    100
      249562  UDEV003  0902  X   26648 CopyOnWrite    9
DEU004   0717    0  UDEV004  0903  X    0  Restored    100
      249743  UDEV004  0903  X   26467 CopyOnWrite    9
DEU005   0718    0  UDEV005  0904  X    0  Restored    100
      249859  UDEV005  0904  X   26351 CopyOnWrite    9
DEU006   0719    0  UDEV006  0905  X    0  Restored    100
      249665  UDEV006  0905  X   26545 CopyOnWrite    9

Total
Track(s)      1516146      141114
MB(s)         47379.6      4409.8

Legend:
<G>: X = The Target device is associated with this group.
     . = The Target device is not associated with this group.

```

Figure 5-13 TimeFinder/Snap listing restore session

In the example shown in 5-13, each source STD device is listed in the first column. Two VDEV devices are associated with each STD device; one is shown with a Restored state in the second last column, and the other is listed with a CopyOnWrite state. Those devices listed with the Restored state are the VDEVs created by the restore operation.

Once completely restored, the session provides no specific value in this environment, and the VDEV devices created by the restore session may be terminated with the following command:

```
symsnap -g <device_group> terminate -restored
```

The specified state for the TF/SIM restore operation is one of NORECOVERY, STANDBY or RECOVERY. The option selected will depend on the specific requirements for the restore process. The options are detailed in the following sections.

5.4.3.1 Restore with NORECOVERY

In general, the NORECOVERY mode is utilized in the instance where a chain of logs is known to follow a full database backup. This allows for the restoration of the data and transaction logs of the database, and then the subsequent application of transaction logs.

It is possible to restore a backup and not require the application of transaction logs, this is the RECOVERY option, and is discussed in a subsequent section.

The command to restore a database instance back to the production server using TF/SIM VDI processing with TF/Snap is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -snap -norecovery
```

This process results in tracks, which were copied to the save area as a result of being updated on the STD, being restored to their preupdated state. Ultimately, this will initiate a restore process against the STD devices to return them back to the state at which the TF/Snap session was activated—that of the backup. Additionally, TF/SIM manages a number of other processes, including the unmount operation of the STD LUN devices used by the database data and log files. An example of the execution process of the TF/SIM command is shown in Figure 5-14.

```

TSIM CLI
TSIM> tsimsnap restore -d ProdDB -f D:\SQL\backup.dat -snap -norecovery
Do you really want to restore the specified database ProdDB (Y/N)? y
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Successfully unmounted file system D:\SQLDB\Data1\
Successfully unmounted file system D:\SQLDB\Data2\
Successfully unmounted file system D:\SQLDB\Data3\
Successfully unmounted file system D:\SQLDB\Data4\
Successfully unmounted file system D:\SQLDB\Data5\
Successfully unmounted file system D:\SQLDB\Log\
Now restoring devices used by database ProdDB, please wait...
Successfully restore devices used by database ProdDB!
Successfully mounted file system D:\SQLDB\Data1\
Successfully mounted file system D:\SQLDB\Data2\
Successfully mounted file system D:\SQLDB\Data3\
Successfully mounted file system D:\SQLDB\Data4\
Successfully mounted file system D:\SQLDB\Data5\
Successfully mounted file system D:\SQLDB\Log\
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 14.222 seconds (0.000 MB/sec).
The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _

```

Figure 5-14 TF/SIM restore production database with TimeFinder/SNAP– NORECOVERY

The resulting state of the database immediately after the TF/SIM snap restore operation is one of a **LOADING** database. A **LOADING** database is not available for access directly, and is therefore different to a read-only state as created by a process such as a **STANDBY**. The state of the database does allow for subsequent transaction log backup files to be restored into the database instance. This application of incremental log backups allows for the minimization of data loss between the time the original database backup occurred and the time at which the failure occurred, which necessitated the restore operation.

The database state can easily be seen by using SQL Server Enterprise Manager, as shown in Figure 5-15.

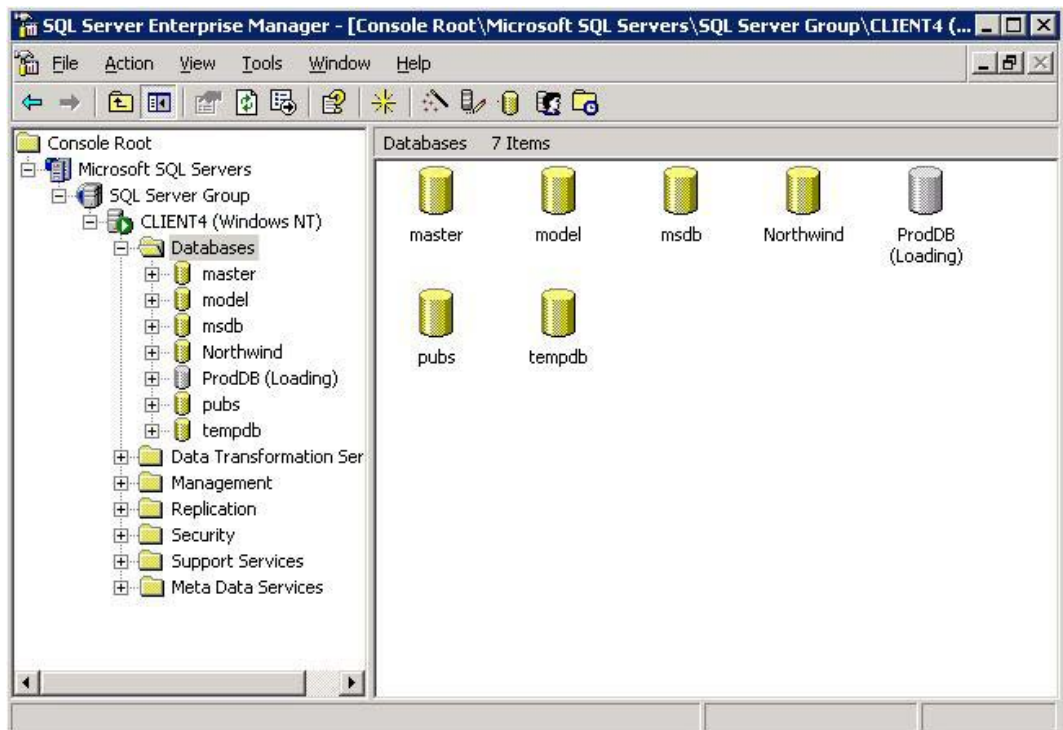


Figure 5-15 SQL Enterprise Manager view of a **LOADING database**

Transaction log backups need to be applied to this database in sequence, specifically in the order in which they were created. The time required to apply the incremental transaction log backups will relate directly to the amount of change which occurred on the production database during normal operations.

Clearly, a restoration of the production database in this manner will necessitate an outage. While much of time requirement may be mitigated by using disk mirror technology to restore the last full backup image, the incremental transaction logs will still be require to be replayed against the restored database. The time required to restore the database and apply transaction logs is referred to as the **Recovery Time Interval**. Typically, customers state this as a **Recovery Time Objective** or that time period that has been dictated by their service-level agreements (SLAs) for which the production database to be unavailable in the event of failure, for restoration purposes.

Application of transaction logs is typically a requirement of any recovery process for a production database. An example of the application of a transaction log increment backup executed from within Query Analyzer is shown in Figure 5-16.

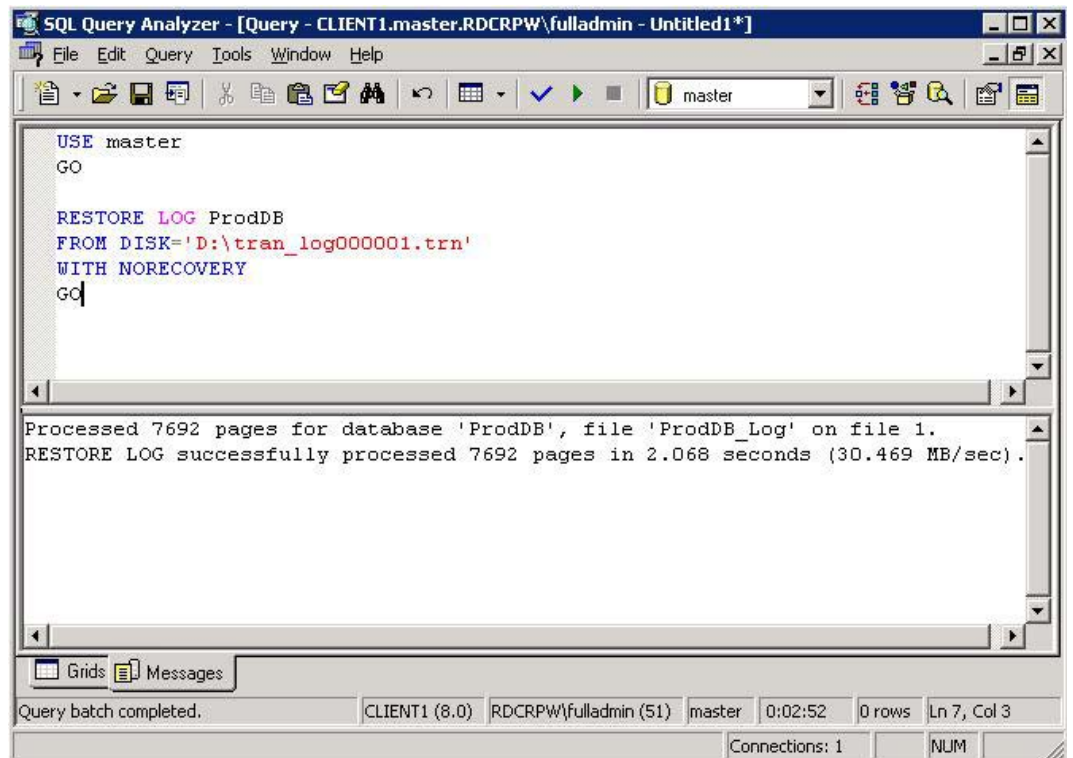


Figure 5-16 Restore of incremental transaction log with NORECOVERY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database in this state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

Remember to terminate the TimeFinder/Snap Restore Session once it has been completely restored.

5.4.3.2 Restore with STANDBY

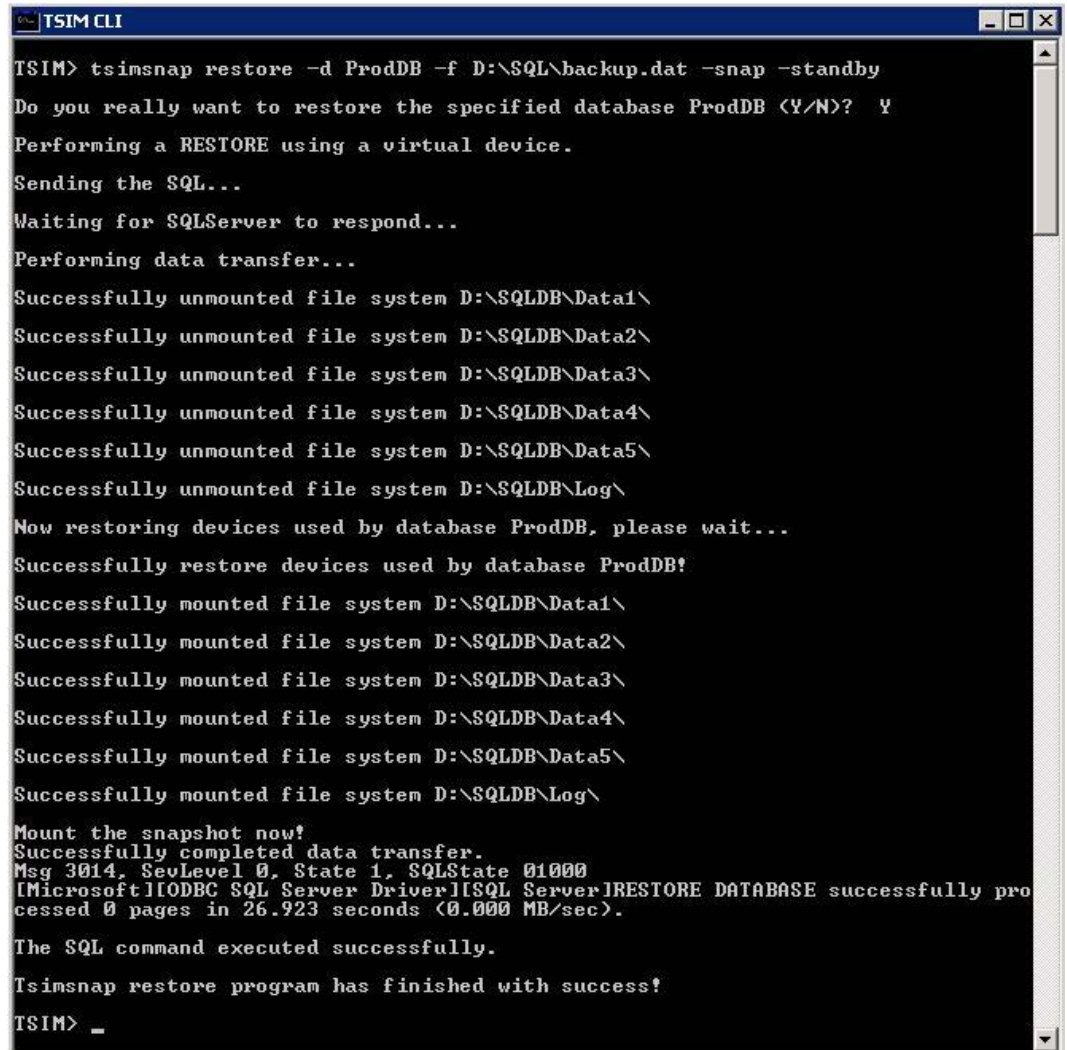
The STANDBY mode is not a typical mode used for restoration of a production database. Typically, either the NORECOVERY or RECOVERY modes are used because they result in the required states. However, it is possible that a database administrator may want to query the state of the database during intervals between incremental transactions log applications. For that purpose, it is discussed here.

To facilitate the application of incremental transaction logs in a similar manner to that when the NORECOVERY option is utilized, but also to provide read only access to the restored database, the STANDBY option may be used. In this way, the restored database is available for those operations that do not attempt to update the target database. Any attempt to execute updates will return a failure, as the database is in a read-only state.

The command to restore a database instance back to the production server using TF/SIM VDI processing is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -snap -standby
```

An example of the execution of the TF/SIM command is shown in Figure 5-17 on page 5-25.



```

TSIM> tsimsnap restore -d ProdDB -f D:\SQL\backup.dat -snap -standby
Do you really want to restore the specified database ProdDB (Y/N)? Y
Performing a RESTORE using a virtual device.
Sending the SQL...
Waiting for SQLServer to respond...
Performing data transfer...
Successfully unmounted file system D:\SQLDB\Data1\
Successfully unmounted file system D:\SQLDB\Data2\
Successfully unmounted file system D:\SQLDB\Data3\
Successfully unmounted file system D:\SQLDB\Data4\
Successfully unmounted file system D:\SQLDB\Data5\
Successfully unmounted file system D:\SQLDB\Log\
Now restoring devices used by database ProdDB, please wait...
Successfully restore devices used by database ProdDB!
Successfully mounted file system D:\SQLDB\Data1\
Successfully mounted file system D:\SQLDB\Data2\
Successfully mounted file system D:\SQLDB\Data3\
Successfully mounted file system D:\SQLDB\Data4\
Successfully mounted file system D:\SQLDB\Data5\
Successfully mounted file system D:\SQLDB\Log\
Mount the snapshot now!
Successfully completed data transfer.
Msg 3014, SevLevel 0, State 1, SQLState 01000
[Microsoft][ODBC SQL Server Driver][SQL Server]RESTORE DATABASE successfully pro
cessed 0 pages in 26.923 seconds (0.000 MB/sec).

The SQL command executed successfully.
Tsimsnap restore program has finished with success!
TSIM> _

```

Figure 5-17 TF/SIM restore production database with TimeFinder/SNAP - STANDBY

During the application of incremental transaction logs it is necessary, however, to terminate any access from other client connections. Exclusive access to the database is required to be able to apply an incremental transaction log created from the production system.

Since the STANDBY state does leave the target database in a read-only state, this can be seen through SQL Server Enterprise Manager, as shown in Figure 5-18 on page 5-26.

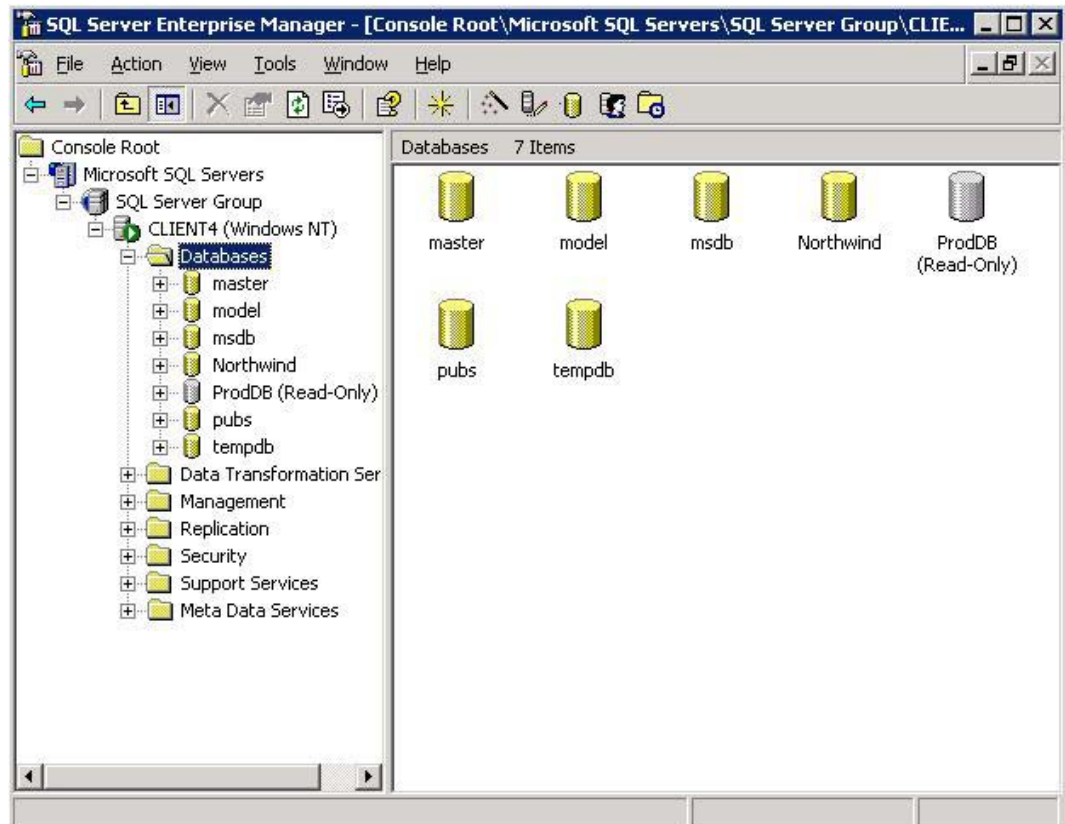


Figure 5-18 SQL Enterprise Manager view of a STANDBY (read-only) database

Similar to the application of incremental transaction logs to a NORECOVERY state, incremental transaction logs may be applied to the STANDBY database, as shown in Figure 5-19 on page 5-27. However, to maintain the STANDBY state it is also necessary to specify the STANDBY option on the Transact-SQL RESTORE LOG statement. It is also required that the location for an undo file is provided. The undo file is used to record those pages rolled back for uncommitted transactions, after the application of a transaction log. The roll back operation ensures that a transactionally consistent view is available of the database state.

Before the application of subsequent incremental transaction logs, the undo file changes are reapplied to the database.

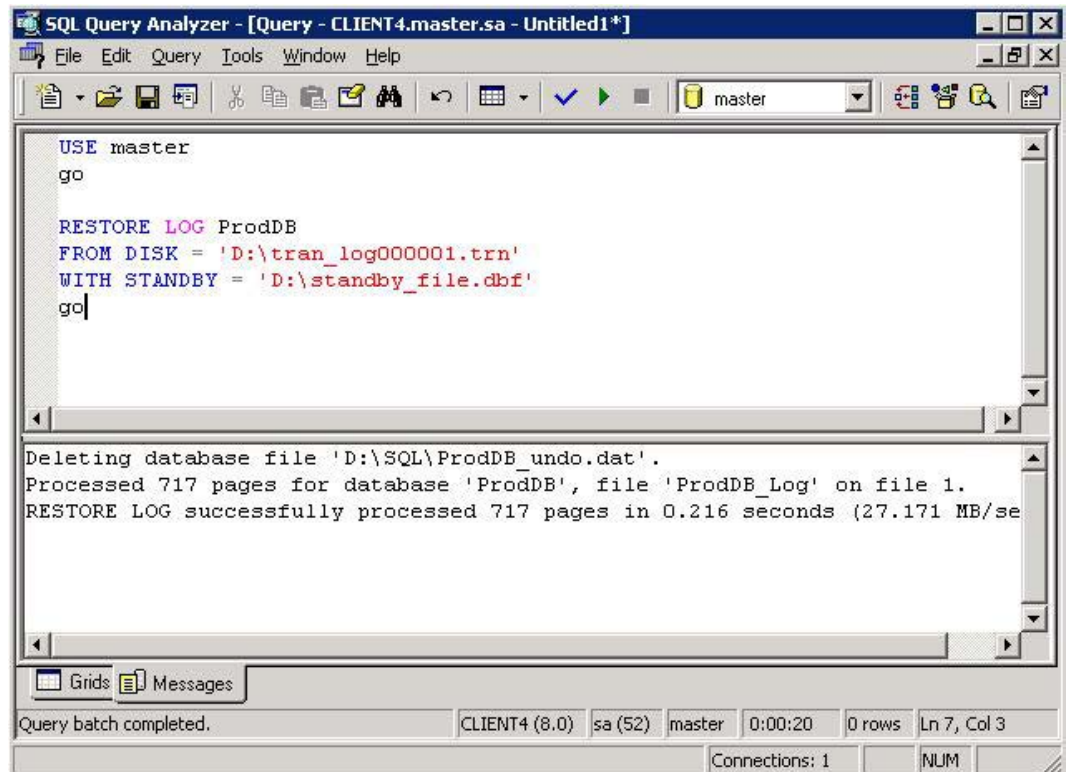


Figure 5-19 Restore of incremental transaction log with STANDBY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, please refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

Remember to terminate the TimeFinder/Snap Restore Session once it has been completely restored.

5.4.3.3 Restore with RECOVERY

In situations where there are no available incremental transaction log backups available for replay against the full database backup, it may be appropriate to simply restore the last full backup and allow SQL Server to bring the database online. In this case, the end state of the operation will be of an online, fully accessible database. No additional incremental transaction logs will be able to be applied to this database instance once recovered, as a new chain of transaction logs will be initiated.

This process is identical to that executed in section 5.4.3.1 except for the use of the RECOVERY option.

The command to restore a database instance back to the production server using TF/SIM VDI processing is:

```
tsimsnap restore -d <DB_alias> -f <metafile> -snap -recovery
```

Remember to terminate the TimeFinder/Snap Restore Session once it has been completely restored.

5.5 SYMIOCTL VDI restore

Backup images created using SYMIOCTL may also be used to restore the production database instance. However, SYMIOCTL provides no automation of activities for the restore process itself and the SYMIOCTL execution is entirely independent of the TimeFinder operations.

In all cases, it is necessary to remove the database object from SQL Server as the first step. This is required, as it is necessary to unmount the file system objects before restoring the BCV image.

Before restoring any production database on the production server, it is always recommended to maintain an existing copy of the database where possible, and to make every effort to create a final transaction log backup.

Figure 5-20 on page 5-29 depicts the following steps which are required for any restore using SYMIOCTL.

1. Detach or delete the target database
2. Unmount drives or mount points which are to be restored from mirror devices
3. Execute restore operation for TimeFinder
4. Mount drives or mount points
5. Execute SYMIOCTL restore operation
6. Apply logs where applicable

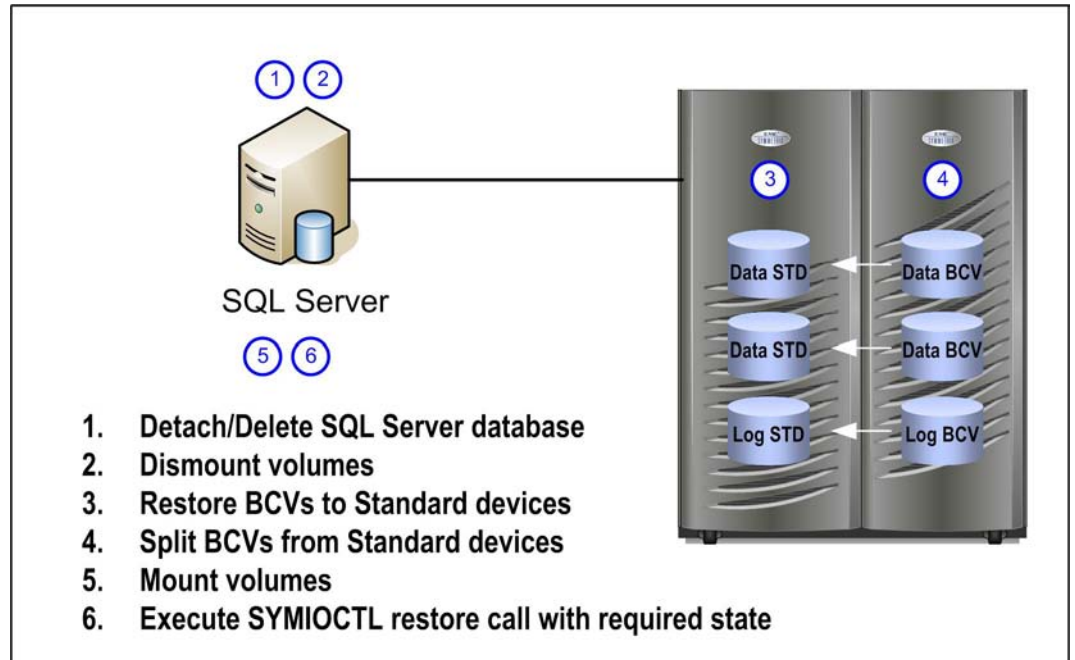


Figure 5-20 Using SYMIOCTL for TimeFinder/Mirror restore of production database

It will be necessary to restore the VDI metadata file created as a result of the backup operation so that it too is accessible for the SYMIOCTL command execution.

The following sections only detail the actual execution of the SYMIOCTL command instance. Detaching and/or deleting the database instance can be managed by the SQL Server Enterprise Manager or by executing the appropriate `sp_detach_db`, `DROP DATABASE` Transact-SQL statement. For correct usage and syntax, please refer to the Microsoft SQL Server Books Online documentation.

It is also assumed that the Symmetrix Integration Utility (SIU) command line process SYMNTCTL will be used for managing the unmount and mount operations. Examples of the execution of SYMNTCTL are provided in Chapter 3. Additional documentation on the SIU can be found in the product guide for the TimeFinder Integration Modules. Refer to Appendix A for details on locating relevant documentation.

5.5.1 Executing TimeFinder restore

Once production database has been detached or otherwise removed from the SQL Server instance, and volumes have been dismounted to facilitate the restore operation, it will be necessary to select the appropriate restore process for the style of TimeFinder operations used to create the initial backup. The three forms of TimeFinder restore covered here will be TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap.

5.5.1.1 TimeFinder/Mirror restore

The restore operation is a standard TimeFinder/Mirror process of the form:

```
symmir -g <device_group> restore -noprompt
```

The execution results in all changed tracks recorded for the STD devices being rolled back to that state when the backup image was created. To check on the restore process, it is possible to query the restore operation in the following way:

```
symmir -g <device_group> query
```

While it is possible to access the STD volumes once the BCV restore process has been initiated, it is recommended that the restore is fully completed, and the BCV devices split from the STDs before proceeding. Alternatively, where possible, use the `-protect` option for the operation. The goal is to ensure that the BCV state is not changed when access is made to the STD volumes. This ensures the continued validity of the BCV backup image.

After the volumes have been restored appropriately and the volumes subsequently re-mounted to the production host, the SYMIOCTL utility may be used to execute the restore operation. It is assumed that the VDI metadata file relating to the original backup operation will be restored to a location on the production host.

5.5.1.2 TimeFinder/Clone restore

The restore operation is a standard TimeFinder/Clone process of the form:

```
symclone -g <device_group> restore -noprompt
```

The execution results in all changed tracks recorded for the STD devices being rolled back to that state when the backup image was created on the clone devices. To check on the restore process, it is possible to query the restore operation in the following way:

```
symclone -g <device_group> query
```

Unlike TimeFinder/Mirror restore operations, TimeFinder/Clone devices do not allow updates to be propagated to the clone devices, which have been restored. Thus, it is possible to access the STD volumes once the clone restore process has been initiated. The validity of the clone backup image is maintained.

After the restore process has been initiated and the volumes subsequently re-mounted to the production host, the SYMIOCTL utility may be used to execute the restore operation. It is assumed that the VDI metadata file relating to the original backup operation will be restored to a location on the production host.

It will also be necessary to terminate the restored clone process by executing the following command:

```
symclone -g <device_group> terminate -noprompt
```

5.5.1.3 TimeFinder/Snap restore

The restore operation is a standard TimeFinder/Snap process of the form:

```
symsnap -g <device_group> restore -noprompt
```

The execution results in all changed tracks recorded for the STD devices being rolled back to that state when the backup image was created on the snap devices. To check on the restore process, it is possible to query the restore operation in the following way:

```
symsnap -g <device_group> query
```

Unlike TimeFinder/Mirror restore operations, TimeFinder/Snap devices do not allow updates to be propagated to the VDEV devices, which have been restored. Thus, it is possible to access the STD volumes once the Snap restore process has been initiated. The validity of the snap backup image is maintained.

After the restore process has been initiated and the volumes subsequently re-mounted to the production host, the SYMIOCTL utility may be used to execute the restore operation. It is assumed that the VDI metadata file relating to the original backup operation will be restored to a location on the production host.

It will also be necessary to terminate the restored snap session by executing the following command:

```
symsnap -g <device_group> terminate -restored -noprompt
```

5.5.2 SYMIOCTL with NORECOVERY

The SYMIOCTL restore command will be of the form:

```
symioctl -type SQLServer restore snapshot <database name> SAVEFILE <metadata file> -norecovery -noprompt
```

Figure 5-21 provides an example of the execution of the SYMIOCTL command.



Figure 5-21 SYMIOCTL restore production database – NORECOVERY

The resulting state of the database immediately after the SYMIOCTL restore operation is one of a LOADING database. A LOADING database is not available for access directly, and is therefore different to a read-only state as created by a process such as a

STANDBY. The state of the database does allow for subsequent transaction log backup files to be restored into the database instance. This application of incremental log backups allows for the minimization of data loss between the time the original database backup occurred and the time at which the failure occurred which necessitated the restore operation.

The database state can easily be seen by using SQL Server Enterprise Manager as shown in Figure 5-22.

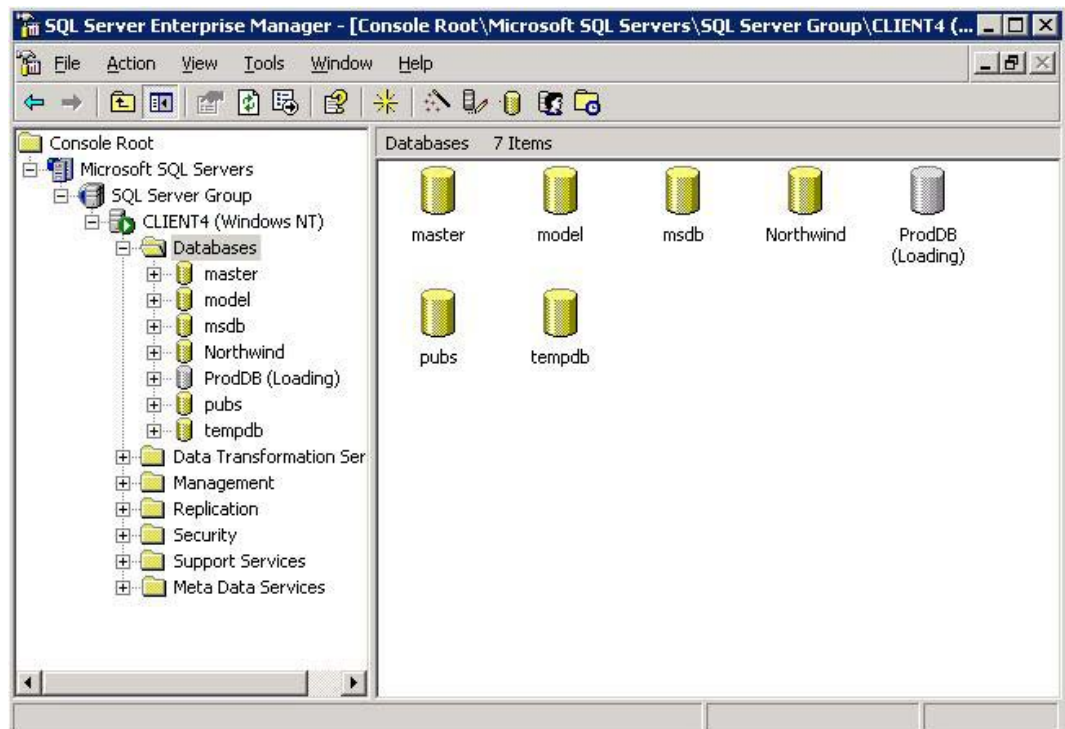


Figure 5-22 SQL Enterprise Manager view of a LOADING database

Transaction log backups need to be applied to this database in sequence, specifically in the order in which they were created. The time required to apply the incremental transaction log backups will relate directly to the amount of change which occurred on the production database during normal operations.

Clearly, a restoration of the production database in this manner will necessitate an outage. While much of time requirement may be mitigated by using disk mirror technology to restore the last full backup image, the incremental transaction logs will still be require to be replayed against the restored database. The time required to restore the database and apply transaction logs is referred to as the Recovery Time Interval. Typically, customers state this as a Recovery Time Objective or that time period that has been dictated by their service-level agreements (SLAs) for which the production database to be unavailable in the event of failure, for restoration purposes.

Application of transaction logs is typically a requirement of any recovery process for a production database. Figure 5-23 on page 5-33 demonstrates the application of an incremental transaction log using SQL Server Query Analyzer.

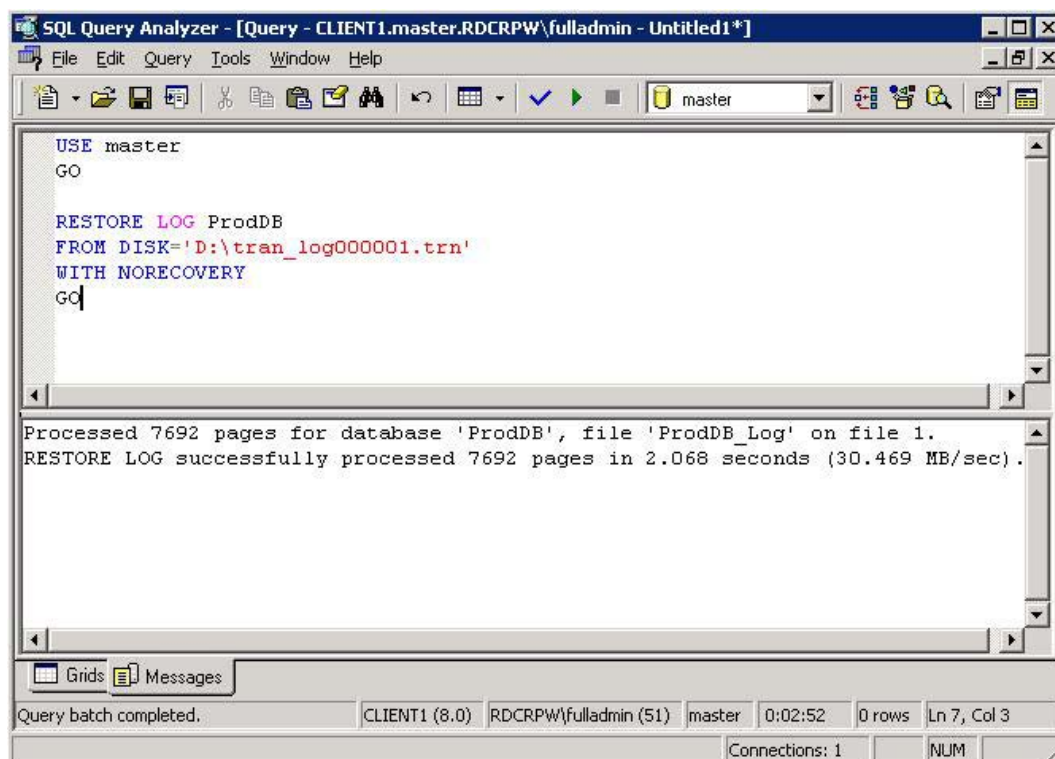


Figure 5-23 Restore of incremental transaction log with NORECOVERY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

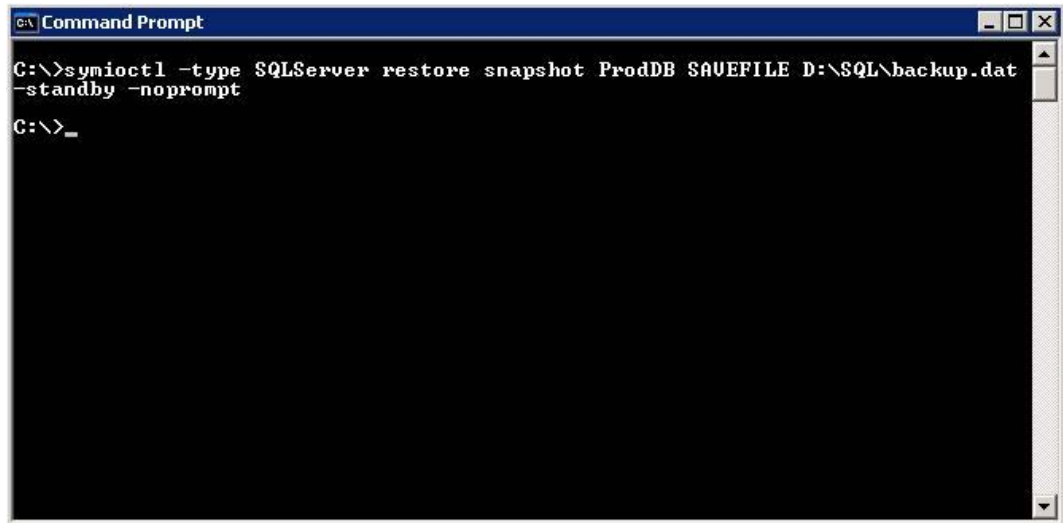
This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

5.5.3 SYMIOCTL with STANDBY

The SYMIOCTL restore command will be of the form:

```
symioctl -type SQLServer restore snapshot <database name> SAVEFILE <metadata file> -standby -noprompt
```

An example of the execution of the SYMIOCTL restore command is shown in Figure 5-24.



```
C:\>symioctl -type SQLServer restore snapshot Proddb SAUEFILE D:\SQL\backup.dat
-standby -noprompt
C:\>_
```

Figure 5-24 SYMIOCTL restore production database – STANDBY

Once completed, the database will be placed in a read-only state. However, during the application of incremental transaction logs it will be necessary to terminate any access from other client connections. Exclusive access to the database is required to be able to apply an incremental transaction log created from the production system.

Since the STANDBY state does leave the target database in a read-only state, this can be seen through SQL Server Enterprise Manager, as shown in Figure 5-25 on page 5-35.

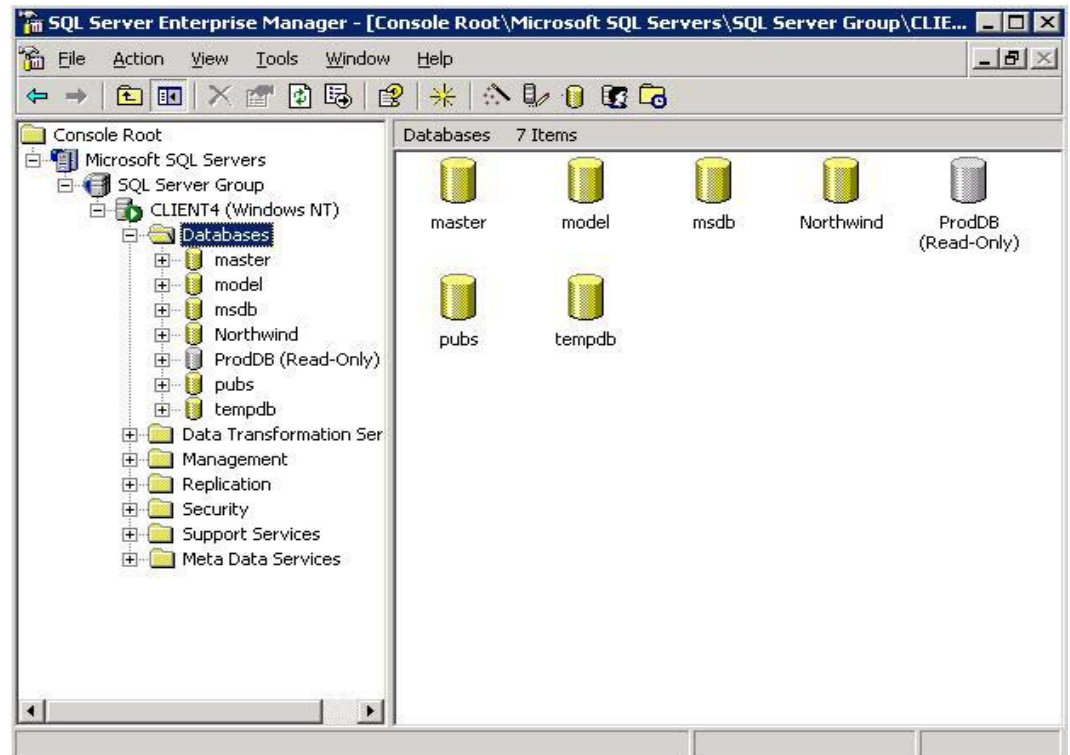


Figure 5-25 SQL Enterprise Manager view of a STANDBY (read-only) database

Similar to the application of incremental transaction logs to a NORECOVERY state, incremental transaction logs may be applied to the STANDBY database. However, to maintain the STANDBY state it is also necessary to specify the STANDBY option on the Transact-SQL RESTORE LOG statement. It is also required that the location for an undo file is provided. The undo file is used to record those pages rolled back for uncommitted transactions, after the application of a transaction log. The roll back operation ensures that a transactionally consistent view is available of the database state. Figure 5-26 on page 5-36 demonstrates the application of an incremental transaction log to a STANDBY database using the relevant Transact SQL commands from within SQL Server Query Analyzer.

Before the application of subsequent incremental transaction logs, the undo file changes are reapplied to the database.

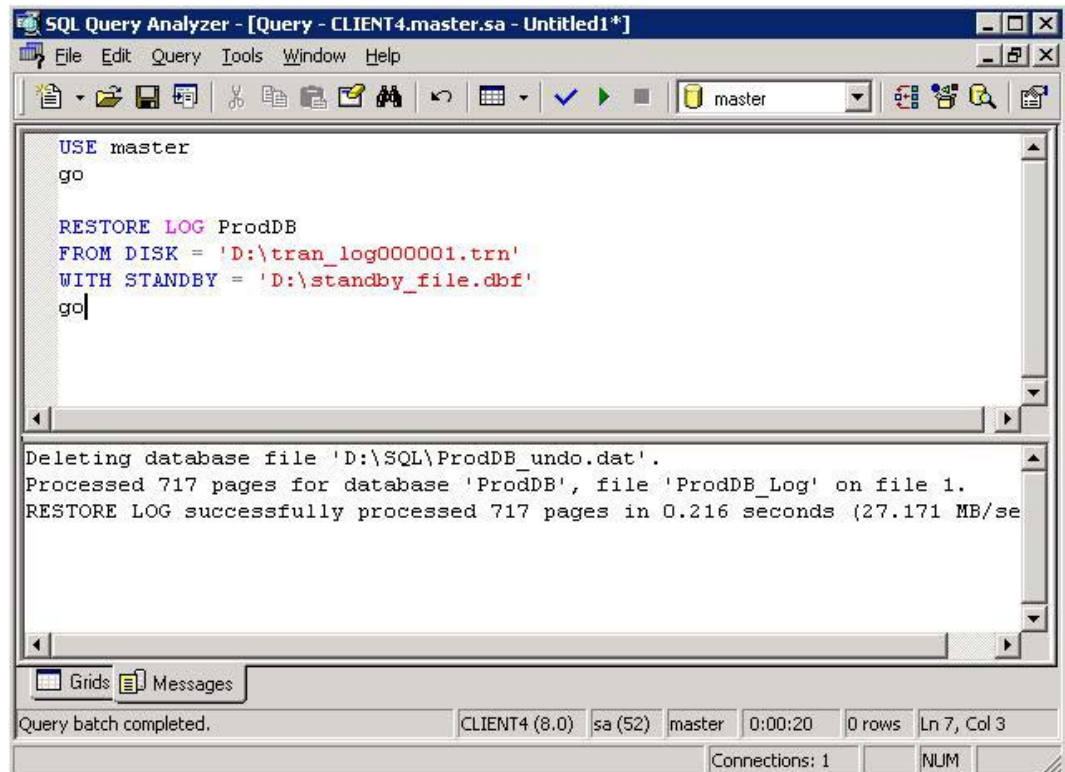


Figure 5-26 Restore of incremental transaction log with STANDBY

The application of incremental transaction logs progress in this manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log, when no further logs are available. Should it be necessary to restore to some point before the end of a given transaction log, refer to section 5.7 for information relating to restoring to timestamps or marked transactions.

5.5.4 SYMIOCTL with RECOVERY

In situations where there are no available incremental transaction log backups available for replay against the full database backup, it may be appropriate to simply restore the last full backup and allow SQL Server to bring the database online. In this case, the end state of the operation will be of an online, fully accessible database. No additional

incremental transaction logs will be able to be applied to this database instance once recovered, as a new chain of transaction logs will be initiated.

As shown in Figure 5-27, the command to restore a database instance back to the production server using SYMIOCTL VDI processing is:

```
symioctl -type SQLServer restore snapshot <database name> SAVEFILE <metadata file> -noprompt
```

The recovery option is implicit when no other forms of recovery are provided.



Figure 5-27 SYMIOCTL restore production database – recovery mode

5.6 Replication Manager VDI restore

Replication Manager provides automated restore procedures through a graphical user interface to simplify the restore process and to allow for the selection of the appropriate replica if there are multiple replicas in existence.

Replication Manager controls the recovery process with respect to the SQL Server instance state for the database, unmount and mount operations, and will also manage the availability of the VDI metadata file on the production system required to complete the restore operation. This process flow is detailed in Figure 5-28 on page 5-38. As a part of the restore request, the administrator is prompted to identify the required state of the database being restored. As with the preceding restore operations, the choices will typically be those of RECOVERY, NORECOVERY, or STANDBY.

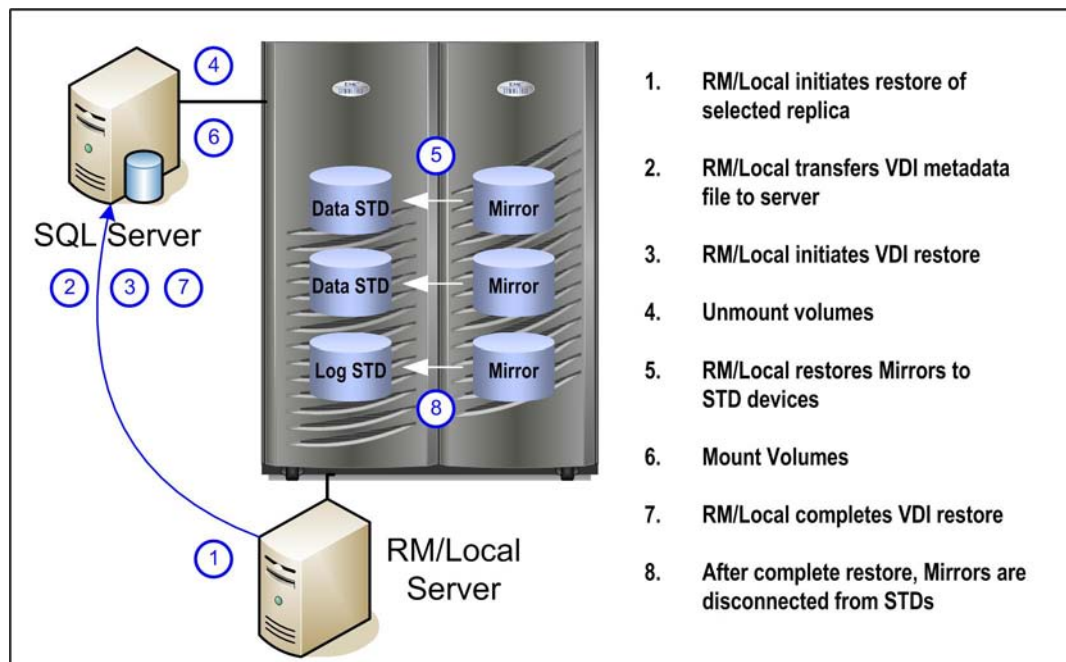


Figure 5-28 Replication Manager/Local restore overview

If Replication Manager was used to create a TimeFinder copy of the database, the restore process must be initiated through the utility. A TimeFinder restore of a Replication Manager copy of a SQL Server database will not be successful.

A more complete discussion on the options available to Replication Manager for SQL Server restoration may be found in the relevant Replication Manager product, user, and administrator guides. Refer to Appendix A for details on locating relevant documentation.

5.7 Applying logs up to timestamps or marked transactions

In those instances where it may be necessary to facilitate recovery to a point in time before some event, it is possible to provide additional options to the RESTORE LOG Transact-SQL statement such that processing of log records within the log backup is terminated at a given location. In general, either timestamps, which are automatically recorded in the transaction log records, or marked transactions can be utilized.

Marked transactions must have been explicitly created on the production database to be used in this manner. These markers placed in the transaction log as a result of executing a marked transaction allow for the roll forward of a full database backup up to the logical point in time when the marked transaction was executed.

Given the aforementioned state of a full database restore procedure being executed in the appropriate NORECOVERY or STANDBY modes described in the previous sections, the appropriate Transact SQL statement may be used to stop processing transaction log records from incremental transaction log backups at the required point.

To restore an incremental transaction log backup and terminate at a point in time recorded in the transaction log records:

```
RESTORE LOG ProdDB  
FROM DISK='D:\SQL\tranlog_00001.trn'  
WITH RECOVERY, STOPAT = 'Dec 10, 2005 10:30 AM'  
go
```

To restore an incremental transaction log backup and terminate at a logical point in time referenced by a marked transaction, such as one created in section 4.3 you may use something similar to the following:

```
RESTORE LOG ProdDB  
FROM DISK='D:\SQL\tranlog_00001.trn'  
WITH RECOVERY, STOPATMARK = 'Before_Update'  
go
```


Chapter 6 Understanding Microsoft SQL Server Disaster Restart and Disaster Recovery

This chapter presents these topics:

6.1	Definitions	6-3
6.2	Design considerations for disaster restart and disaster recovery	6-4
6.3	Tape-based solutions.....	6-8
6.4	Local high-availability solutions	6-9
6.5	Multisite high-availability solutions	6-9
6.6	Remote replication challenges	6-10
6.7	Array-based remote replication	6-13
6.8	Planning for array-based replication.....	6-14
6.9	SQL Server specific issues	6-14
6.10	SRDF/S: Single Symmetrix to single Symmetrix.....	6-15
6.11	SRDF/S and consistency groups	6-17
6.12	SRDF/A	6-22
6.13	SRDF/AR single hop	6-26
6.14	SRDF/AR multi hop	6-28
6.15	Database log-shipping solutions	6-30
6.16	Running database solutions	6-43
6.17	Other transactional systems	6-45

A critical part of managing a database is planning for unexpected loss of data. The loss can occur from a disaster like fire or flood, or it can come from hardware or software failures. It can even be caused by human error or malicious intent. In each instance, the database must be restored to some usable point before application services can be resumed.

The effectiveness of any plan for restart or recovery involves answering the following questions:

- ◆ How much down time is acceptable to the business?
- ◆ How much data loss is acceptable to the business?
- ◆ How complex is the solution?
- ◆ Does the solution accommodate the data architecture?
- ◆ How much does the solution cost?
- ◆ What disasters does the solution protect against?
- ◆ Is there protection against logical corruption?
- ◆ Is there protection against physical corruption?
- ◆ Is the database restartable or recoverable?
- ◆ Can the solution be tested?
- ◆ If failover happens, will failback work?

All restart and recovery plans include a replication component. In its simplest form, the replication process may be as easy as making a tape copy of the database and application. In a more sophisticated form, it could be real-time replication of all changed data to some remote location. Remote replication of data has its own challenges centered on:

- ◆ Distance
- ◆ Propagation delay (latency)
- ◆ Network infrastructure
- ◆ Data loss

This chapter provides an introduction to the spectrum of disaster recovery and disaster restart solutions for SQL Server databases on EMC Symmetrix arrays.

6.1 Definitions

In the next sections, the terms dependent-write consistency, database restart, database recovery, and roll forward recovery are used. A clear definition of these terms is required to understand the context of this section.

6.1.1 Dependent-write consistency

A dependent-write I/O is one that cannot be issued until a related predecessor I/O has completed. Dependent-write consistency is a data state where data integrity is guaranteed by dependent-write I/Os embedded in application logic. Database management systems are good examples of the practice of dependent-write consistency.

Database management systems must devise protection against abnormal termination to successfully recover from one. The most common technique used is to guarantee that a dependent-write cannot be issued until a predecessor write has completed. Typically, the dependent-write is a data or index write while the predecessor write is a write to the log. Because the write to the log must be completed before issuing the dependent-write, the application thread is synchronous to the log write, that is, it waits for that write to complete before continuing. The result of this kind of strategy is a dependent-write consistent database.

6.1.2 Database restart

Database restart is the implicit application of database logs during its normal initialization process to ensure a transactionally consistent data state.

If a database is shut down normally, the process of getting to a point of consistency during restart requires minimal work. If the database abnormally terminates, then the restart process will take longer, depending on the number and size of in-flight transactions at the time of termination. An image of the database created by using EMC Consistency technology while it is running, without conditioning the database, will be in a dependent-write consistent data state, which is similar to that created by a local power failure. This is also known as a DBMS restartable image. The restart of this image transforms it to a transactionally consistent data state by completing committed transactions and rolling back uncommitted transactions during the normal database initialization process.

6.1.3 Database recovery

Database recovery is the process of rebuilding a database from a backup image, and then explicitly applying subsequent logs to roll the data state forward to a designated point of consistency. Database recovery is only possible with databases configured with the appropriate level of database logging, typically using the FULL recovery model.

A recoverable database copy can be taken in one of three ways:

1. With the database shut down and copying the database components using external tools.
2. With the database running using the SQL Server VDI or VSS frameworks.

3. Streamed backup using SQL Server BACKUP processes.

6.1.4 Roll forward recovery

With some databases, it may be possible to take a DBMS restartable image of the database, and apply subsequent incremental transaction log backups, to roll forward the database to a point in time after the image was created. This means that the image created can be used in a backup strategy in combination with transaction log backups. At the time of printing, a DBMS restartable image of a SQL Server database cannot use subsequent logs to roll forward transactions. In most cases, during a disaster, the storage array image at the remote site will be a SQL Server restartable image and cannot have any incremental transaction log backups applied it.

6.2 Design considerations for disaster restart and disaster recovery

Loss of data or loss of application availability has a varying impact from one business type to another. For instance, data loss in the form of transactions for a bank could cost millions, whereas system downtime may not have a major fiscal impact. On the other hand, businesses that are primarily web-based may require 100 percent application availability in order to survive. The two factors, loss of data and loss of uptime, are the business drivers that are baseline requirements for a DR solution. When quantified, these two factors are more formally known as Recovery Point Objective (RPO) and Recovery Time Objective (RTO), respectively.

When evaluating a solution, the RPO and RTO requirements of the business need to be met. In addition, the solution needs to consider operational complexity, cost, and the ability to return the whole business to a point of consistency. Each of these aspects is discussed in the following sections.

6.2.1 Recovery Point Objective (RPO)

The RPO is a point of consistency to which a user wants to recover or restart. It is measured in the amount of time from when the point of consistency was created or captured to the time the disaster occurred. This time equates to the acceptable amount of data loss. Zero data loss (no loss of committed transactions from the time of the disaster) is the ideal goal, but the high cost of implementing such a solution must be weighed against the business impact and cost of a controlled data loss.

Some organizations, like banks, have zero data loss requirements. The database transactions entered at one location must be replicated immediately to another location. This can have an impact on application performance when the two locations are far apart. On the other hand, keeping the two locations close together might not protect against regional disasters like a power outage in the Northeast United States or hurricanes in Florida.

Defining the required RPO is usually a compromise between the needs of the business, the cost of the solution, and the risk of a particular event happening.

6.2.2 Recovery Time Objective (RTO)

The RTO is the maximum amount of time allowed for recovery or restart to a specified point of consistency. This time involves many factors including the time it takes to:

- ◆ Provision power, utilities, and so on
- ◆ Provision servers with the application and database software
- ◆ Configure the network
- ◆ Restore the data at the new site
- ◆ Roll forward the data to a known point of consistency
- ◆ Validate the data

Some delays can be reduced or eliminated by choosing certain DR options like having a hot site where servers are preconfigured and on standby. Also, if storage-based replication is used, the time it takes to restore the data to a usable state is completely eliminated.

As with RPO, each solution for RTO will have a different cost profile. Defining the RTO is usually a compromise between the cost of the solution and the cost to the business when database and applications are unavailable.

6.2.3 Operational complexity

The operational complexity of a DR solution may be the most critical factor in determining the success or failure of a DR activity. The complexity of a DR solution can be considered as three separate phases.

1. Initial set up of the implementation.
2. Maintenance and management of the running solution.
3. Execution of the DR plan in the event of a disaster.

While initial configuration complexity and running complexity can be a demand on people resources, the third phase, execution of the plan, is where automation and simplicity must be the focus. When a disaster is declared, key personnel may not be available in addition to the loss of servers, storage, networks, buildings, etc. If the complexity of the DR solution is such that skilled personnel with an intimate knowledge of all systems involved are required to restore, recover and validate application and database services, the solution has a high probability of failure.

Multiple database environments grow organically over time into complex federated database architectures. In these federated database environments, reducing the complexity of DR is absolutely critical. Validation of transactional consistency within the complex database architecture is time consuming, costly, and requires application

and database familiarity. One reason for the complexity is because of the heterogeneous databases and operating systems involved in these federated environments. Across multiple heterogeneous platforms, it is hard to establish a common clock and therefore hard to determine a business point of consistency across all platforms. This business point of consistency has to be created from intimate knowledge of the transactions and data flows.

6.2.4 Source server activity

DR solutions may or may not require additional processing activity on the source servers. The extent of that activity can impact both response time and throughput of the production application. This effect should be understood and quantified for any given solution to ensure the impact to the business is minimized. The effect for some solutions is continuous while the production application is running; for other solutions, the impact is sporadic, where bursts of write activity are followed by periods of inactivity.

6.2.5 Production impact

Some DR solutions delay the host activity while taking actions to propagate the changed data to another location. This action only affects write activity and although the introduced delay may only be of the order of a few milliseconds, it can impact response time in a high-write environment. Synchronous solutions introduce delay into write transactions at the source site, asynchronous solutions do not.

6.2.6 Target server activity

Some DR solutions require a target server at the remote location to perform DR operations. The server has both software and hardware costs and needs personnel with physical access to it for basic operational functions like power on and power off. Ideally, this server could have some usage like running development or test databases and applications. Some DR solutions require more target server activity and some require none.

6.2.7 Number of copies

DR solutions require replication of data in one form or another. Replication of a database and associated files can be as simple as making a tape backup and shipping the tapes to a DR site or as sophisticated as asynchronous array-based replication. Some solutions require multiple copies of the data to support DR functions. More copies of the data may be required to perform testing of the DR solution in addition to those that support the DR process.

6.2.8 Distance for solution

Disasters, when they occur, have differing ranges of impact. For instance, a fire may take out a building, an earthquake may destroy a city, or a tidal wave may devastate a region. The level of protection for a DR solution should address the probable disasters for a given location. For example when protecting against an earthquake, the DR site should not be in the same locale as the production site. For regional protection, the two sites need to be in two different regions. The distance associated with the DR solution affects the kind of DR solution that can be implemented.

6.2.9 Bandwidth requirements

One of the largest costs for DR is in provisioning bandwidth for the solution. Bandwidth costs are an operational expense; this makes solutions that have reduced bandwidth requirements very attractive to customers. It is important to recognize in advance the bandwidth consumption of a given solution to be able to anticipate the running costs. Incorrect provisioning of bandwidth for DR solutions can adversely affect production performance and can invalidate the overall solution.

6.2.10 Federated consistency

Databases are rarely isolated islands of information with no interaction or integration with other applications or databases. Most commonly, databases are loosely and/or tightly coupled to other databases using triggers, database links, and stored procedures. Some databases provide information downstream for other databases using information distribution middleware; other databases receive feeds and inbound data from message queues and EDI transactions. The result can be a complex interwoven architecture with multiple interrelationships. This is referred to as a federated database architecture.

With a federated database architecture, making a DR copy of a single database without regard to other components invites consistency issues and creates logical data integrity problems. All components in a federated architecture need to be recovered or restarted to the same dependent-write consistent point of time to avoid these problems.

With this in mind, it is possible that point database solutions for DR, like log-shipping, do not provide the required business point of consistency in a federated database architecture. Federated consistency solutions guarantee that all components, databases, applications, middleware, flat files, and so on are recovered or restarted to the same dependent-write consistent point in time.

6.2.11 Testing the solution

Tested, proven, and documented procedures are also required for a DR solution. Many times, the DR test procedures are operationally different from a true disaster set of procedures. Operational procedures need to be clearly documented. In the best-case scenario, companies should periodically execute the actual set of procedures for DR. This could be costly to the business because of the application downtime required to perform such a test, but is necessary to ensure validity of the DR solution.

6.2.12 Cost

The cost of doing DR can be justified by comparing it to the cost of not doing it. What does it cost the business when the database and application systems are unavailable to users? For some companies this is easily measurable, and revenue loss can be calculated per hour of downtime or per hour of data loss.

Whatever the business, the DR cost is going to be an extra expense item and, in many cases, with little in return. The costs include, but are not limited to:

- ◆ Hardware (storage, servers and maintenance)
- ◆ Software licenses and maintenance
- ◆ Facility leasing/purchase
- ◆ Utilities
- ◆ Network infrastructure
- ◆ Personnel

6.3 Tape-based solutions

6.3.1 Tape-based disaster recovery

Traditionally, the most common form of disaster recovery was to make a copy of the database onto tape and using Pickup Truck Access Method (PTAM), take the tapes offsite to a hardened facility. In most cases, the database and application needed to be available to users during the backup process. Taking a backup of a running database created a “fuzzy” image of the database on tape, one that required database recovery after the image had been restored. Recovery usually involved application of logs that were active during the time the backup was in process. These logs had to be archived and kept with the backup image to ensure successful recovery.

The rapid growth of data over the last two decades has meant that this method has become unmanageable. Making a “hot” copy of the database is now the standard—but this method has its own challenges. How can a consistent copy of the database and supporting files be made when they are changing throughout the duration of the backup? What exactly is the content of the tape backup at completion? The reality is that the tape data is a “fuzzy image” of the disk data, and considerable expertise is required to restore the database back to a database point of consistency.

In addition, the challenge of returning the data to a business point of consistency, where a particular database must be recovered to the same point as other databases or applications, is making this solution less viable.

6.3.2 Tape-based disaster restart

Tape-based disaster restart is a recent development in disaster recovery strategies and is used to avoid the “fuzziness” of a backup taken while the database and application are running. A “restart” copy of the system data is created by locally mirroring the disks that contain the production data, and splitting off the mirrors to create a dependent-write consistent point-in-time image on the disks. This image is a DBMS restartable image as previously described. Thus, if this image was restored and the database brought up, the database would perform an implicit recovery to attain transactional consistency. Roll-forward recovery using incremental transaction logs from this database image is not possible for Microsoft SQL Server.

The restartable image on the disks can be backed up to tape and moved offsite to a secondary facility. If this image is created and shipped offsite on a daily basis, the maximum amount of data loss is 24 hours.

The time it takes to restore the database is a factor to consider since reading from tape is typically slow. Consequently, this solution can be effective for customers with relaxed RTOs.

6.4 Local high-availability solutions

Microsoft provides a commonly known high-availability solution known as Windows Cluster Server. This is an implementation of a shared disk environment, though in many ways it is also a shared nothing environment. While storage devices must be accessible to multiple nodes within a given cluster, any individual disk resource cannot be accessible to more than one node at any given time.

Furthermore, Microsoft Cluster Server provides the capability to aggregate resources into resource groups and define dependency. In this way, resource groups may be defined, which logically define an application. In the case of a SQL Server cluster instance, the resource group contains a number of resources such as a network name, an IP address, disk resources and SQL Server processes themselves. With the ability to define dependencies between resources, it is possible to provide structured start and stop processes to ensure reliable operations. For example, the SQL Server service is dependent on both disk resources, and the network name. The network name in turn depends on the IP address. Thus, the higher-level functions ensure that lower-level services are operational before proceeding.

The Microsoft Cluster Server itself is actually a restart environment at its core. In the event that a node running a SQL Server instance fails, for example, the Cluster Service will attempt to move all resources to a peer node to restart the services. During this restart process, SQL Server services will open the database files on the shared disks, detect the failure state, and execute implicit recovery operations against the database to ensure that a transactionally consistent state is obtained. This is an identical process for images created by consistent split processing using EMC Consistency technology.

Since in a single array MSCS environments used a single set of shared disks, disk failures, or other physical or logical corruption of the disk objects, or data maintained within them will affect that Cluster Resource Group irrespective of the node on which it is operating on. There is no protection against this style of logical or physical failure in single array solutions.

6.5 Multisite high-availability solutions

To extend the functionality of single site MSCS configurations and provide additional multisite protection, EMC provides the SRDF/CE for MSCS geographically dispersed clustering product. This solution extends the single array MSCS configuration to allow for multiple sites with array-based replication between the two sites. Certification of the solution is provided by Microsoft in a similar manner to single MSCS configurations. Certified configurations may be found on the Windows Catalog available from the Microsoft website at <http://www.microsoft.com/>.

SRDF/CE for MSCS, provides a level of abstract at the storage level such that MSCS believes it is executing in a standard mode. All typically MSCS functions, procedures, and restrictions remain in place. The benefit provided by SRDF/CE for MSCS is that it can provide single-site failure conditions, and enhance the RTO for applications by automatically managing recovery of resource groups.

It is important to understand SRDF/CE for MSCS operating modes, and specifically the default behavior in the event of a site failure. The default behavior is “No New Onlines,” which indicates that surviving modes may continue to run their current resource groups but will not be able to start new resource groups. Overrides are provided, and are documented in the product guide for SRDF/CE for MSCS.

Geographically dispersed clustering solutions such as that of SRDF/CE for MSCS based on SRDF/S provide zero data loss solutions with extremely small RTO, since most processes are automated.

EMC also provides the AutoStart™ resource management product which provides similar functionality for geographically dispersed configurations. AutoStart is not based on Microsoft Cluster Service, and utilizes its own mechanisms for managing state and ensuring availability of resources. Please refer to the product guide for EMC AutoStart for additional information. Additional details on reference documentation are provided in Appendix A.

6.6 Remote replication challenges

Replicating database information over long distances for the purpose of disaster recovery is challenging. Synchronous replication over distances greater than 200 km may not be feasible because of the negative impact on the performance of writes because of propagation delay; some form of asynchronous replication must be adopted. Considerations in this section apply to all forms of remote replication technology, whether they are array-based, host-based, or managed by the database.

Remote replication solutions usually start with initially copying a full database image to the remote location. This is called instantiation of the database. There are a variety of ways to perform this. After instantiation, only the changes from the source site are replicated to the target site in an effort to keep the target up to date. Some methodologies may not send all of the changes (certain log shipping techniques for instance), by omission rather than design. These methodologies may require periodic re-instantiation of the database at the remote site.

The following considerations apply to remote replication of databases:

- ◆ Propagation delay (latency because of distance)
- ◆ Bandwidth requirements
- ◆ Network infrastructure
- ◆ Method of instantiation
- ◆ Method of re-instantiation

- ◆ Change rate at the source site
- ◆ Locality of reference
- ◆ Expected data loss
- ◆ Failback operations

6.6.1 Propagation delay

Electronic operations execute at the speed of light. The speed of light in a vacuum is 186,000 miles per second. The speed of light through glass (in the case of fiber optic media) is less, approximately 115,000 miles per second. In other words, in an optical network like SONET for instance, it takes 1 millisecond to send a data packet 125 miles or 8 milliseconds for 1000 miles. All remote replication solutions need to be designed with a clear understanding of the propagation delay impact.

6.6.2 Bandwidth requirements

All remote replication solutions have some bandwidth requirements because the changes from the source site must be propagated to the target site. The more changes there are, the greater the bandwidth that is needed. It is the change rate and replication methodology that determine the bandwidth requirement, not necessarily the size of the database.

Data compression can help reduce the quantity of data transmitted and therefore the size of the “pipe” required. Certain network devices, like switches and routers, provide native compression, some by software and some by hardware. GigE directors provide native compression in a DMX to DMX SRDF pairing. The amount of compression achieved is dependent on the type of data that is being compressed. Typical character and numeric database data compresses at about a 2-to-1 ratio. A good way to estimate how the data will compress is to assess how much tape space is required to store the database during a full backup process. Tape drives perform hardware compression on the data before writing it. For instance, if a 300 GB database takes 200 GB of space on tape, the compression ratio is 1.5 to 1.

For most customers, a major consideration in the disaster recovery design is cost. It is important to recognize that some components of the end solution represent a capital expenditure and some an operational expenditure. Bandwidth costs are operational expenses and thus any reduction in this area, even at the cost of some capital expense, is highly desirable.

6.6.3 Network infrastructure

The choice of channel extension equipment, network protocols, switches, and routers ultimately determines the operational characteristics of the solution. EMC has a proprietary BC Design Tool to assist customers in their analysis of the source systems and to determine the required network infrastructure to support a remote replication solution.

6.6.4 Method of instantiation

In all remote replication solutions, a common requirement is for an initial, consistent copy of the complete database to be replicated to the remote site. The initial copy from source to target is called instantiation of the database at the remote site. Following instantiation, only the changes made at the source site are replicated. For large databases, sending only the changes after the initial copy is the only practical and cost-effective solution for remote database replication.

In some solutions, instantiation of the database at the remote site uses a process that is similar to the one that replicates the changes. Some solutions do not even provide for instantiation at the remote site (log shipping for instance). In all cases, it is critical to understand the pros and cons of the complete solution.

6.6.5 Method of re-instantiation

Some methods of remote replication require periodic refreshing of the remote system with a full copy of the database. This is called re-instantiation. Technologies such as log shipping frequently require this since not all activity on the production database may be represented in the log. In these cases, the disaster recovery plan must account for re-instantiation and also for the fact there may be a disaster during the refresh. The business objectives of RPO and RTO must likewise be met under those circumstances.

6.6.6 Change rate at the source site

After instantiation of the database at the remote site, only changes to the database are replicated remotely. There are many methods of replication to the remote site and each has differing operational characteristics. The changes can be replicated using logging technology, hardware and software mirroring for example. Before designing a solution with remote replication, it is important to quantify the average change rate. It is also important to quantify the change rate during periods of burst write activity. These periods might correspond to end-of-month/quarter/year processing, billing, or payroll cycles. The solution needs to be designed to allow for peak write workloads.

6.6.7 Locality of reference

Locality of reference is a factor that needs to be measured to understand if there will be a reduction of bandwidth consumption when any form of asynchronous transmission is used. Locality of reference is a measurement of how much write activity on the source is skewed. For instance, a high locality of reference application may make many updates to a few tables in the database, whereas a low locality of reference application rarely updates the same rows in the same tables during a given period of time.

It is important to understand that while the activity on the tables may have a low locality of reference, the write activity into an index might be clustered when inserted rows have the same or similar index column values, rendering a high locality of reference on the index components.

In some asynchronous replication solutions, updates are “batched” up into periods of time and sent to the remote site to be applied. In a given batch, only the last image of a given row/block is replicated to the remote site. So, for highly skewed application

writes, this results in bandwidth savings. Generally, the greater the time period of batched updates, the greater the savings on bandwidth.

Log shipping technologies do not take into account locality of reference. For example, a row updated a 100 times, is transmitted 100 times to the remote site, whether the solution is synchronous or asynchronous.

6.6.8 Expected data loss

Synchronous DR solutions are zero data loss solutions, that is to say, there is no loss of committed transactions from the time of the disaster. Synchronous solutions may also be impacted by a rolling disaster in which case work completed at the source site after the rolling disaster started may be lost. Rolling disasters are discussed in detail in a later section.

Non-synchronous DR solutions have the potential for data loss. How much data is lost depends on many factors, most of which have been previously defined. For asynchronous replication, where updates are batched and sent to the remote site, the maximum amount of data lost will be two cycles or two batches worth. The two cycles that may be lost include the cycle currently being captured on the source site and the one currently being transmitted to the remote site. With inadequate network bandwidth, data loss could be increased because of the increased transmission time.

6.6.9 Failback operations

If there is the slightest chance that fail over to the DR site may be required, then there is a 100 percent chance that failback to the primary site will also be required, unless the primary site is lost permanently. The DR architecture should be designed in such a way as to make failback simple, efficient, and low-risk. If failback is not planned for, there may be no reasonable or acceptable way to move the processing from the DR site, where the applications may be running on tier 2 servers and tier 2 networks, back to the production site.

Ideally, the DR process should be tested once a quarter, with database and application services fully failed over to the DR site. The integrity of the application and database needs to be verified at the remote site to ensure that all required data was copied successfully. Ideally, production services are brought up at the DR site as the ultimate test. This means that production data would be maintained on the DR site, requiring a failback when the DR test completed. While this is not always possible, it is the ultimate test of a DR solution. It not only validates the DR process, but also trains the staff on managing the DR process should a catastrophic failure ever occur. The downside for this approach is that duplicate sets of servers and storage need to be present to make an effective and meaningful test. This tends to be an expensive proposition.

6.7 Array-based remote replication

Customers can use the capabilities of a Symmetrix storage array to replicate the database from the production location to a secondary location. No host CPU cycles are used for this, leaving the host dedicated to running the production application and database. In

addition, no host I/O is required to facilitate this, the array takes care of all replication and no hosts are required at the target location to manage the target array.

EMC provides multiple solutions for remote replication of databases:

- ◆ SRDF/S: SRDF/Synchronous
- ◆ SRDF/A: SRDF/Asynchronous
- ◆ SRDF/AR: SRDF/Automated Replication

Each of these solutions is discussed in detail in the next sections. In order to use any of the array-based solutions, it is necessary to coordinate the disk layout of the databases with this kind of replication in mind.

6.8 Planning for array-based replication

All Symmetrix solutions replicating data from one array to another are disk based. This allows the Symmetrix to be agnostic to the volume manager, file system, database system, and so on. However, this does not mean that file system and volume manager concerns can be ignored. It is important to understand the relationship of Windows NT File System (NTFS) volumes and LUNs. On Windows, the smallest unit of granularity for storage-based replication is the LUN, a volume set, or disk group, depending on how the disks are set up in disk manager.

In addition, if a database is to be replicated independently of other databases, it should have its own dedicated disks (LUNs). That is, the disks used by a database should not be shared with other applications or databases.

When a set of volumes has been defined for a database for remote replication, care must be taken to ensure that the disks contain everything that is needed to restart the database at the remote site. For SQL Server databases, this must be the complete set of data and transaction log files comprising the database.

Administrators should ensure that the same versions of SQL Server, including service packs and hot fixes, are installed on both the production and target. In instances where application binaries are being replicated with the databases, this will be resolved by the replication process.

6.9 SQL Server specific issues

A number of issues need to be addressed in configurations that utilize remote replication technologies, specifically those that are designed to replicate only a given SQL Server database. One of the most important considerations is that of security, access and authentication information.

SQL Server maintains security and authentication information within the MASTER DB. If the replication solution does not include replicating the MASTER DB and other systems databases, then it is required that some process exists which deals adequately with replicating this information. Should this authentication information not be replicated, then it is possible to attain a state where the copy of the user database is made

available, but users are unable to access the copied database because of authentication failures.

Microsoft provides documentation on a process for managing the export of this account and authentication information in Knowledge Base article 246133 for SQL Server 2000. It is recommended that this process be implemented in configurations where replication of user databases is implemented as a way to provide high availability and system databases are not replicated.

6.10 SRDF/S: Single Symmetrix to single Symmetrix

SRDF/Synchronous, or SRDF/S, is a method of replicating production data changes from locations that are no greater than 200 km apart. Synchronous replication takes writes that are inbound to the source Symmetrix and copies them to the target Symmetrix. The write operation is not acknowledged as complete to the host until both Symmetrix arrays have the data in cache. It is important to realize that while the following examples involve Symmetrix, the fundamentals of synchronous replication described here are true for all synchronous replication solutions. Figure 6-1 depicts the process.

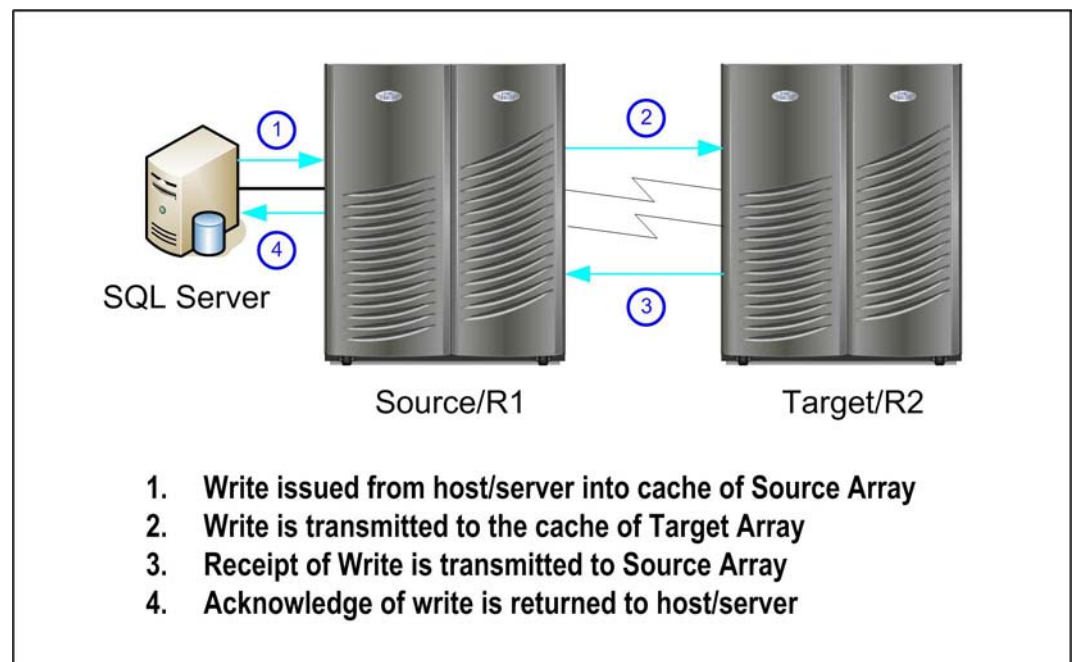


Figure 6-1 SRDF/S replication process

1. A write is received into the source Symmetrix cache. At this time, the host has not received acknowledgement that the write is complete.
2. The source Symmetrix uses SRDF/S to push the write to the target Symmetrix.
3. The target Symmetrix sends an acknowledgement back to the source that the write was received.

4. Ending status of the write is presented to the host.

These four steps cause a delay in the processing of writes as perceived by the database on the source server. The amount of delay depends on the exact configuration of the network, the storage, the write block size, and the distance between the two locations. Note that reads from the source Symmetrix are not affected by the replication.

The following steps outline the process of setting up synchronous replication using Solutions Enabler (SYMCLI) commands.

1. Before the synchronous mode of SRDF can be established, initial instantiation of the database has to take place. In other words, a baseline full copy of all the volumes that are going to participate in the synchronous replication must be executed first. This is usually accomplished using the adaptive copy mode of SRDF. The following command create a group where *<device_group>* is a user specified name:

```
symmdg create <device_group> -type rdf1
```

The type of the device group is dependent on the location of the system being used to define the device group. The *rdf1* type is used if the host is connected locally to the Source/R1 array.

2. Add devices to the group as described in Appendix B.
3. The following command puts the group into adaptive copy mode:

```
symrdf -g <device_group> set mode acp_disk -noprompt
```

4. The following command causes the source Symmetrix to send all the tracks on the source site to the target site using the current mode:

```
symrdf -g <device_group> establish -full -noprompt
```

The adaptive copy mode of SRDF has no impact to host application performance. It transmits tracks to the remote site that have never been sent before or that have changed since the last time the track was sent. It does not preserve write order or dependent-write consistency.

5. When both sides are synchronized, SRDF can then be put into synchronous mode. In the following command, the device group is put into synchronous mode:

```
symrdf -g <device_group> set mode sync -noprompt
```

There is no requirement for host availability at the remote site during the synchronous replication. The target Symmetrix itself manages the in-bound writes and updates the appropriate volumes in the array.

Dependent-write consistency is inherent in a synchronous relationship as the target R2 volumes are at all times equal to the source provided that a single RA group is used. If multiple RA groups are used or if multiple Symmetrix arrays are used on the source site, SRDF/Consistency Groups (SRDF/CG) must be used to guarantee consistency.

6.10.1 How to restart in the event of production site loss

In the event of a disaster where the primary source Symmetrix is lost, it becomes necessary to run database and application services from the DR site. A host at the DR site is required for this. The first requirement is to write enable the R2 devices. If the device group is not yet built on the remote host, it must be created using the R2 devices that were mirrors of the R1 devices on the source Symmetrix. Group Named Services (GNS) can be used to propagate the device group to the remote site if there is a host being utilized there. For more details on GNS see the *Solutions Enabler Symmetrix Base Management CLI Product Guide*.

The following command write enables the R2s in a group:

```
symld -g <device_group> rw_enable -noprompt
```

At this point, the host can issue the necessary commands to access the disks.

Once the data is available to the host, the database can be restarted. The database will perform an implicit recovery when restarted. Transactions that were committed but not completed are rolled forward and completed using the information in the transaction log. Transactions that have updates applied to the database but were not committed are rolled back. The result is a transactionally consistent database.

6.11 SRDF/S and consistency groups

Zero data loss disaster recovery techniques tend to use straightforward database and application restart procedures. These procedures work well if all processing and data mirroring stop at the same instant in time at the production site when a disaster happens. Such is the case when there is a site power failure.

However, in most cases, it is unlikely that all data processing ceases at an instant in time. Computing operations can be measured in nanoseconds and even if a disaster takes only a millisecond to complete, many such computing operations could be completed between the start of a disaster until all data processing ceases. This gives us the notion of a rolling disaster. A rolling disaster is a series of events taken over a period of time that comprise a true disaster. The specific period of time that makes up a rolling disaster could be milliseconds (in the case of an explosion) or minutes in the case of a fire. In both cases, the DR site must be protected against data inconsistency.

6.11.1 Rolling disaster

Protection against a rolling disaster is required when the data for a database resides on more than one Symmetrix array or multiple RA groups. Figure 6-2 depicts a dependent-write I/O sequence where a predecessor log write is happening before a page flush from a database buffer pool. The log device and data device are on different Symmetrix arrays with different replication paths. Figure 6-2 demonstrates how rolling disasters can affect this dependent-write sequence.

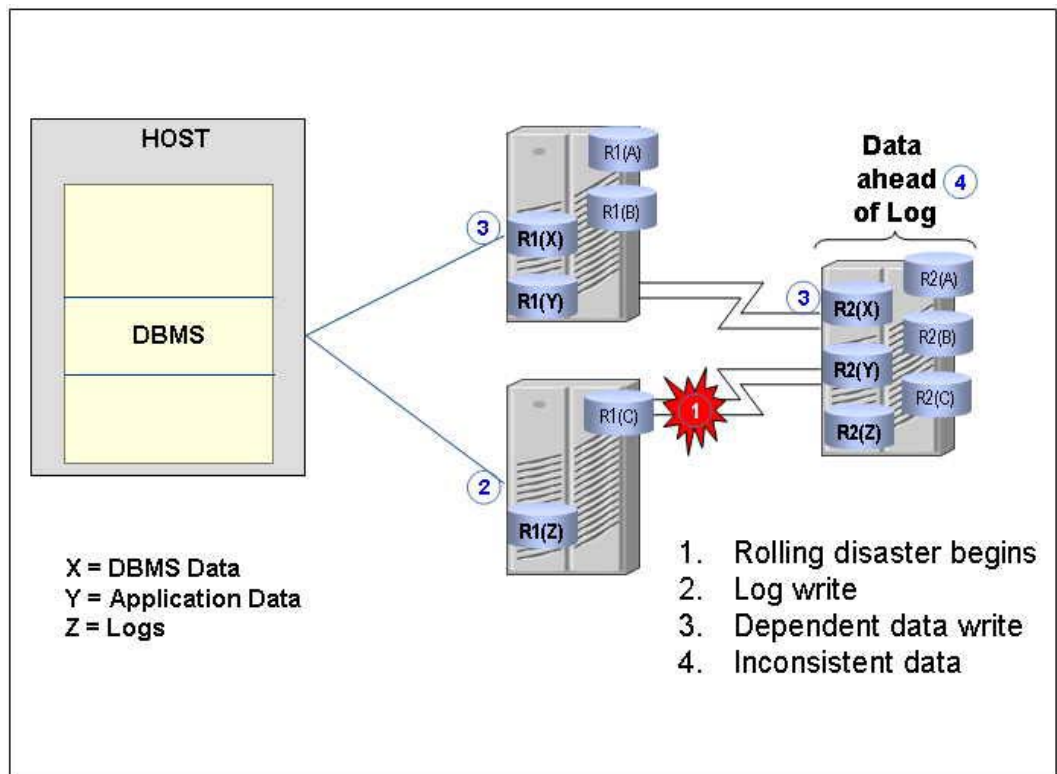


Figure 6-2 Rolling disaster with multiple production Symmetrix arrays

1. This example of a rolling disaster starts with a loss of the synchronous links between the bottom source Symmetrix and the target Symmetrix. This will prevent the remote replication of data on the bottom source Symmetrix.
2. The Symmetrix, which now is no longer replicating, receives a predecessor log write of a dependent-write I/O sequence. The local I/O is completed, however, it is not replicated to the remote Symmetrix, and the tracks are marked as being “owed” to the target Symmetrix. Nothing prevents the predecessor log write from completing to the host completing the acknowledgement process.
3. Now that the predecessor log write has completed, the dependent data write is issued. This write is received on both the source Symmetrix and the target Symmetrix because the rolling disaster has not yet affected those communication links.
4. If the rolling disaster ended in a complete disaster, the condition of the data at the remote site is such that it creates a “data ahead of log” condition, which is an inconsistent state for a database. The severity of the situation is that when the database is restarted, performing an implicit recovery, it may not detect the inconsistencies. A person extremely familiar with the transactions running at the time of the rolling disaster might be able to detect the inconsistencies. Database utilities could also be run to detect some of the inconsistencies.

A rolling disaster can happen in such a manner that data links providing remote mirroring support are disabled in a staggered fashion, while application and database processing continues at the production site. The sustained replication during the time when some Symmetrix units are communicating with their remote partners through their respective links while other Symmetrix units are not (because of link failures) can cause data integrity exposure at the recovery site. Some data integrity problems caused by the rolling disaster cannot be resolved through normal database restart processing and may require a full database recovery using appropriate backups, journals, and logs. A full database recovery elongates overall application restart time at the recovery site.

6.11.2 Protecting against a rolling disaster

SRDF/Consistency Group (SRDF/CG) technology provides protection against rolling disasters. A consistency group is a set of Symmetrix volumes spanning multiple RA groups and/or multiple Symmetrix frames that replicate as a logical group to other Symmetrix arrays using Synchronous SRDF. It is not a requirement to span multiple RA groups and/or Symmetrix frames when using consistency groups. Consistency group technology guarantees that if a single source volume is unable to replicate to its partner for any reason, then all the volumes in the group stop replicating. This ensures that the image of the data on the target Symmetrix is consistent from a dependent-write perspective.

Figure 6-3 depicts a dependent-write I/O sequence where a predecessor log write is happening before a page flush from a database buffer pool. The log device and data device are on different Symmetrix arrays with different replication paths. Figure 6-3 demonstrates how rolling disasters can be prevented using EMC consistency group technology.

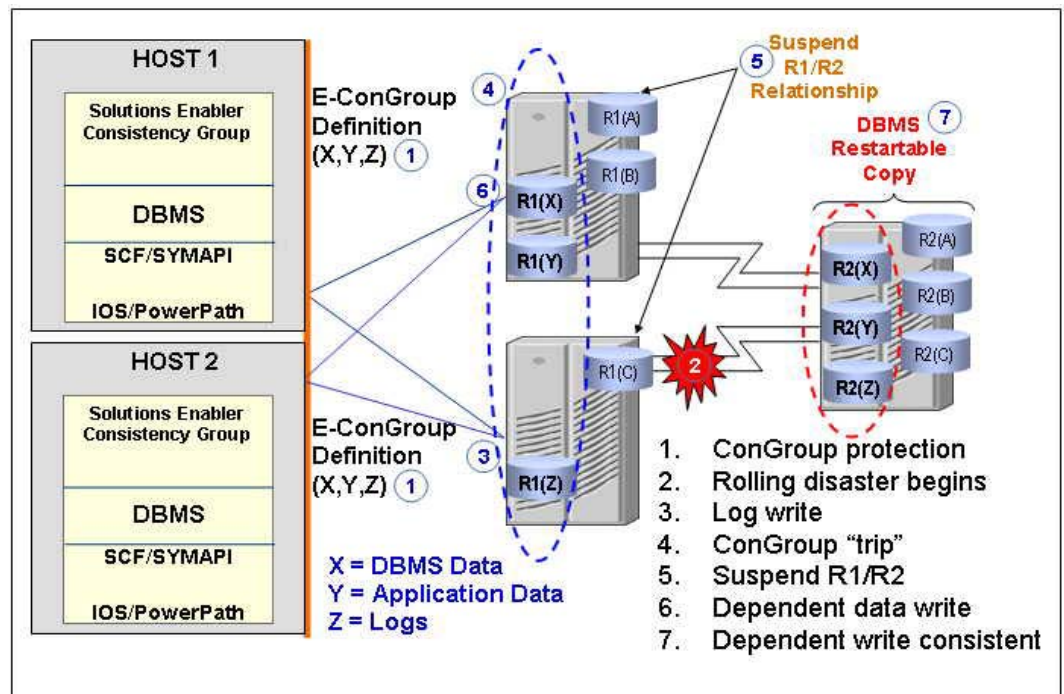


Figure 6-3 SRDF consistency group protection against rolling disaster

1. Consistency group protection is defined containing volumes X, Y, and Z on the source Symmetrix. This consistency group definition must contain all of the devices that need to maintain dependent-write consistency and reside on all participating hosts involved in issuing I/O to these devices. A mix of CKD (mainframe) and FBA (UNIX/Windows) devices can be logically grouped together. In some cases, the entire processing environment may be defined in a consistency group to ensure dependent-write consistency.
2. The rolling disaster previously described begins preventing the replication of changes from volume Z to the remote site.
3. The predecessor log write occurs to volume Z, causing a consistency group (ConGroup) trip.
4. A ConGroup trip will hold the I/O that could not be replicated along with all of the I/O to the logically grouped devices. The I/O is held by PowerPath on the UNIX or Windows hosts, and IOS on the mainframe host. It is held long enough to issue two(2) I/Os per Symmetrix. The first I/O will put the devices in a suspend-pending state.
5. The second I/O performs the suspend of the R1/R2 relationship for the logically grouped devices, which immediately disables all replication to the remote site. This allows other devices outside of the group to continue replicating provided the communication links are available.
6. After the R1/R2 relationship is suspended, all deferred write I/Os are released allowing the predecessor log write to complete to the host. The dependent data write is issued by the DBMS and arrives at X but is not replicated to the R2(X).
7. If a complete failure occurred from this rolling disaster, dependent-write consistency at the remote site is preserved. If a complete disaster did not occur and the failed links were activated again, the consistency group replication could be resumed once synchronous mode is achieved. It is recommended to create a copy of the dependent-write consistent image while the resume takes place. Once the SRDF process reaches synchronization the dependent-write consistent copy is achieved at the remote site.

6.11.3 SRDF/S with multiple source Symmetrix arrays

The implications of spreading a database across multiple Symmetrix frames or across multiple RA groups and replicating in synchronous mode were discussed in previous sections. The challenge in this type of scenario is to protect against a rolling disaster. SRDF consistency groups can be used to avoid data corruption in a rolling disaster situation.

Consider the architecture depicted in Figure 6-4 on page 6-21.

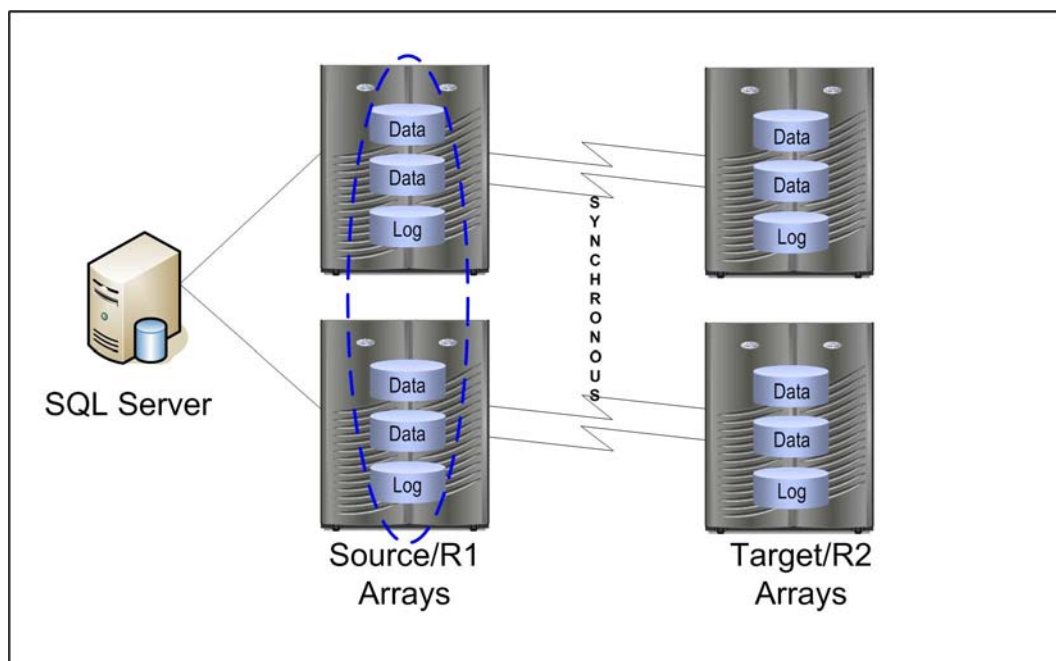


Figure 6-4 SRDF/S with multiple source Symmetrix and ConGroup protection

To protect against a rolling disaster, a consistency group can be created that encompasses all the volumes on all Symmetrix arrays participating in replication, as shown by the blue dotted oval.

The following steps outline the process of setting up synchronous replication with consistency groups using Solutions Enabler (SYMCLI) commands.

1. To create a consistency group for the source side of the synchronous relationship, that is, the R1 side:

```
symcg create <composite_group> -type rdf1 -ppath
```
2. Add devices to the group as described in Appendix B.
3. Before the synchronous mode of SRDF can be established, the initial instantiation of the database has to have taken place. In other words, the baseline full copy of all the volumes that are going to participate in the synchronous replication must be executed first. This is usually accomplished using adaptive copy mode of SRDF.
4. The following command puts the consistency group into adaptive copy mode:

```
symrdf -cg <composite_group> set mode acp_disk -noprompt
```
5. The following command causes the source Symmetrix to send all tracks at the source site to the target site using the current mode:

```
symrdf -cg <composite_group> establish -full -noprompt
```

6. Adaptive copy mode has no host impact. It transmits tracks to the remote site that have never been sent before or that have changed since the last time the track was sent. It does not preserve write order or consistency. When both sides are synchronized, SRDF can be put into synchronous mode. In the following command, the consistency group is put into synchronous mode:

```
symrdf -cg <composite_group> set mode sync -noprompt
```

7. To enable consistency protection, use the following command:

```
symcg -cg <composite_group> enable -noprompt
```

There is no requirement for a host at the remote site during the synchronous replication. The target Symmetrix manages the in-bound writes and updates the appropriate disks in the array

6.12 SRDF/A

SRDF/A, or SRDF/Asynchronous, is a method of replicating production data changes from one Symmetrix to another using delta set technology. Delta sets are the collection of changed blocks grouped together by a time interval that can be configured at the source site. The default time interval is 30 seconds. The delta sets are then transmitted from the source site to the target site in the order they were created. SRDF/A preserves dependent-write consistency of the database at all times at the remote site.

The distance between the source and target Symmetrix is unlimited and there is no host impact. Writes are acknowledged immediately when they hit the cache of the source Symmetrix. SRDF/A is only available on the DMX family of Symmetrix. Figure 6-5 depicts the process.

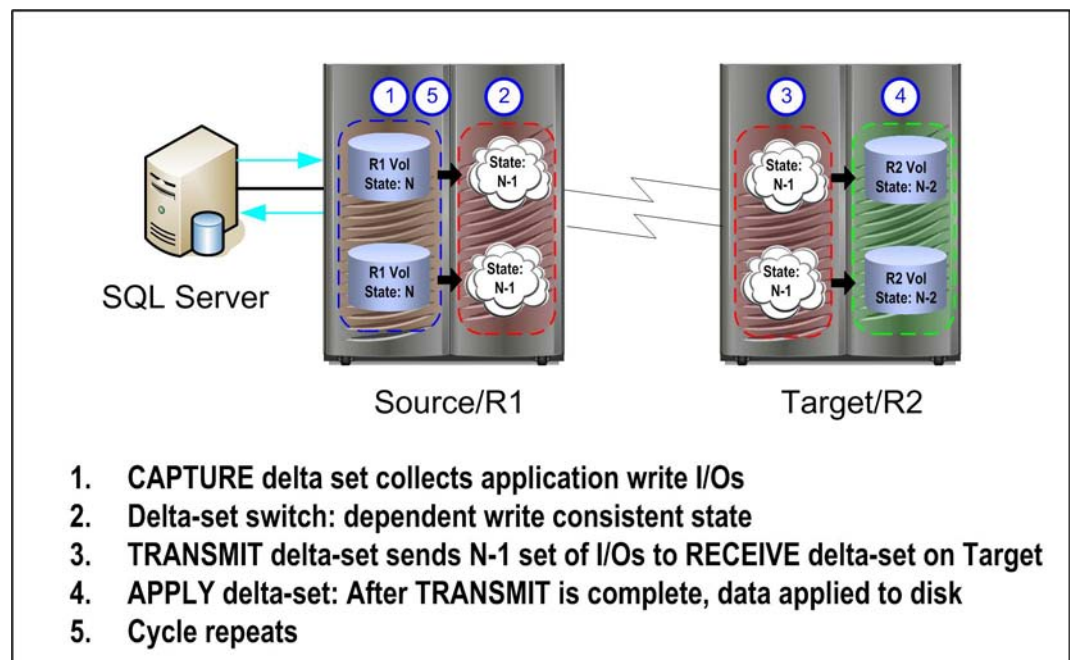


Figure 6-5 SRDF/Asynchronous replication internals

1. Writes are received into the source Symmetrix cache. The host receives immediate acknowledgement that the write is complete. Writes are gathered into the capture delta set for 30 seconds.
2. A delta set switch occurs and the current capture delta set becomes the transmit delta set by changing a pointer in cache. A new empty capture delta set is created.
3. SRDF/A sends the changed blocks that are in the transmit delta set to the remote Symmetrix. The changes collect in the receive delta set at the target site. When the replication of the transmit delta set is complete, another delta set switch occurs and a new empty capture delta set is created with the current capture delta set becoming the new transmit delta set. The receive delta set becomes the apply delta set.
4. The apply delta set marks all the changes in the delta set against the appropriate volumes as invalid tracks and begins destaging the blocks to disk.
5. The cycle repeats continuously.

With sufficient bandwidth for the source database write activity, SRDF/A will transmit all changed data within the default 30 seconds. This means that the maximum time the target data will be behind the source is 60 seconds (two replication cycles). At times of high write activity, it may not be possible to transmit all the changes that occur during a 30 second interval. This means that the target Symmetrix will fall behind the source Symmetrix by more than 60 seconds. Careful design of the SRDF/A infrastructure and a thorough understanding of write activity at the source site are necessary to design a solution that meets the RPO requirements of the business at all times.

Consistency is maintained throughout the replication process on a delta set boundary. The Symmetrix will not apply a partial delta set which would invalidate consistency. Dependent-write consistency is preserved by placing a dependent-write in either the same delta set as the write it depends on or a subsequent delta set.

There is no requirement for a host at the remote site during asynchronous replication. The target Symmetrix manages in-bound writes and updates the appropriate disks in the array.

Different command sets are used to enable SRDF/A depending on whether the SRDF/A group of devices is contained within a single Symmetrix or is spread across multiple Symmetrix arrays.

6.12.1 SRDF/A using single source Symmetrix

Before the asynchronous mode of SRDF can be established, initial instantiation of the database has to take place. In other words, a baseline full copy of all the volumes that are going to participate in the asynchronous replication must be executed first. This is usually accomplished using the adaptive copy mode of SRDF.

The following steps outline the process of setting up asynchronous replication using Solutions Enabler (SYMCLI) commands.

1. To create an SRDF disk group for the source side of the synchronous relationship, that is, the R1 side:

```
symmdg create <device_group> -type rdf1
```

2. Add devices to the group as described in Appendix B.
3. The following command puts the device group into adaptive copy mode:

```
symrdf -g <device_group> set mode acp_disk -noprompt
```

4. The following command causes the source Symmetrix to send all the tracks at the source site to the target site using the current mode:

```
symrdf -g <device_group> establish -full -noprompt
```

5. The adaptive copy mode of SRDF has no impact to host application performance. It transmits tracks to the remote site that have never been sent before or that have changed since the last time the track was sent. It does not preserve write order or consistency. When both sides are synchronized, SRDF can be put into asynchronous mode. In the following command, the device group is put into asynchronous mode:

```
symrdf -g <device_group> set mode async -noprompt
```

There is no requirement for a host at the remote site during the asynchronous replication. The target Symmetrix manages the in-bound writes and updates the appropriate disks in the array.

6.12.2 SRDF/A using multiple source Symmetrix

When a database is spread across multiple Symmetrix arrays and SRDF/A is used for long-distance replication, separate software must be used to manage the coordination of the delta set boundaries between the participating Symmetrix arrays and to stop replication if any of the volumes in the group cannot replicate for any reason. The software must ensure that all delta set boundaries on every participating Symmetrix in the configuration are coordinated to give a dependent-write consistent point-in-time image of the database.

SRDF/A multisession consistency (MSC) provides consistency across multiple RA groups and/or multiple Symmetrix arrays. MSC is available on 5671 microcode and later with Solutions Enabler V6.0 and later. SRDF/A with MSC is supported by an SRDF process daemon that performs cycle-switching and cache recovery operations across all SRDF/A sessions in the group. This ensures that a dependent-write consistent R2 copy of the database exists at the remote site at all times. A composite group must be created using the SRDF consistency protection option (-rdf_consistency) and must be enabled using the symcg enable command before the RDF daemon begins monitoring and managing the MSC consistency group. The RDF process daemon must be running on all hosts that can write to the set of SRDF/A volumes being protected. At the time of an interruption (SRDF link failure, for instance), MSC analyzes the status of all SRDF/A sessions and either commits the last cycle of data to the R2 target or discards it.

The following steps outline the process of setting up synchronous replication with consistency groups using Solutions Enabler (SYMCLI) commands.

1. The replication composite group for the SRDF/A devices can be created using the command:

```
symcg create <composite_group> -rdf_consistency
-type rdf1
```

The *-rdf_consistency* option indicates that the volumes that will be in the group are to be protected by MSC.

2. Add devices to the group as described in Appendix B.
3. Before the asynchronous mode of SRDF can be established, the initial copy of the database has to take place. In other words, the baseline full copy of all the volumes that are going to participate in the asynchronous replication must be executed first. This is usually accomplished using the adaptive copy mode of SRDF. The following command puts the group into adaptive copy mode:

```
symrdf -g <composite_group> set mode acp_disk -noprompt
```

4. The following command causes the source Symmetrix to send all the tracks at the source site to the target site using the current mode:

```
symrdf -g <composite_group> establish -full -noprompt
```

5. The adaptive copy mode of SRDF has no impact on host application performance. It transmits tracks to the remote site that have never been sent before or that have changed since the last time the track was sent. It does not preserve write order or consistency. When both sides are synchronized, SRDF can be put into asynchronous mode. In the following command, the composite group is put into asynchronous mode:

```
symrdf -g <composite_group> set mode async -noprompt
```

6. To enable multisession consistency for the group, execute the following command:

```
symcg -cg <composite_group> enable
```

There is no requirement for a host at the remote site during the asynchronous replication. The target Symmetrix itself manages the in-bound writes and updates the appropriate disks in the array.

6.12.3 Restart processing

In the event of a disaster when the primary source Symmetrix is lost, database and application services must be run from the DR site. A host at the DR site is required for this. If the device or composite group is not defined yet on the remote host, it must first be created using the R2 devices that were mirrors of the R1 devices on the source Symmetrix. The first step that must be done is to write-enable the R2 devices.

R2s on a single Symmetrix:

```
symld -g <device_group> rw_enable -noprompt
```

R2s on multiple Symmetrix:

```
symcgs -cg <composite_group> rw_enable -noprompt
```

At this point, the host can issue the necessary commands to access the disks. This includes the steps required for mounting the volumes appropriately as mount points or drive letters.

Once the data is available to the host, the database can be restarted. The database will perform crash recovery when the database is attached. Transactions that were committed but not completed are rolled forward and completed using the information in the active logs. Transactions that have updates applied to the database but not committed are rolled back. The result is a transactionally consistent database.

6.13 SRDF/AR single hop

SRDF/Automated Replication, or SRDF/AR, is a continuous movement of dependent-write consistent data to a remote site using SRDF adaptive copy mode and TimeFinder consistent split technology. TimeFinder BCVs are used to create a dependent-write consistent point-in-time image of the data to be replicated. The BCVs also have an R1 personality, which means that SRDF in adaptive copy mode can be used to replicate the data from the BCVs to the target site. Since the BCVs are not changing, replication completes in a finite length of time. The length of time for replication depends on the size of the network “pipe” between the two locations, the distance between the two locations, the quantity of changed data tracks, and the locality of reference of the changed tracks. On the remote Symmetrix, another BCV copy of the data is made using data on the R2s. This is necessary because the next SRDF/AR iteration replaces the R2 image in a nonordered fashion, and if a disaster were to occur while the R2s were synchronizing, there would not be a valid copy of the data at the DR site. The BCV copy of the data in the remote Symmetrix is commonly called the “gold” copy of the data. The whole process then repeats.

With SRDF/AR, there is no host impact. Writes are acknowledged immediately when they hit the cache of the source Symmetrix. Figure 6-6 on page 6-27 depicts the process.

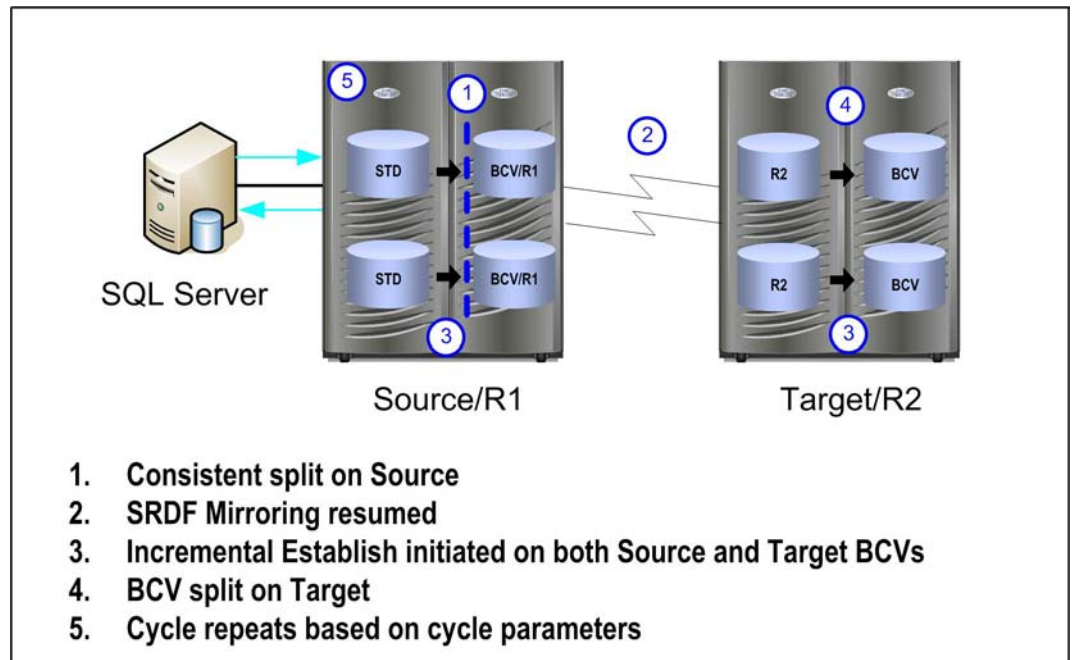


Figure 6-6 SRDF/AR single-hop replication internals

1. Writes are received into the source Symmetrix cache and are acknowledged immediately. The BCVs are already synchronized with the STDs at this point. A consistent split is executed against the STD-BCV pairing to create a point-in-time image of the data on the BCVs.
2. SRDF transmits the data on the BCV/R1s to the R2s in the remote Symmetrix.
3. When the BCV/R1 volumes are synchronized with the R2 volumes, they are reestablished with the standards in the source Symmetrix. This causes the SRDF links to be suspended. At the same time, an incremental establish is performed on the target Symmetrix to create a gold copy on the BCVs in that frame.
4. When the BCVs in the remote Symmetrix are fully synchronized with the R2s, they are split and the configuration is ready to begin another cycle.
5. The cycle repeats based on configuration parameters. The parameters can specify the cycles to begin at specific times, specific intervals or to run back to back.

Note that cycle times for SRDF/AR are usually in the minutes to hours range. The RPO is double the cycle time in a worst-case scenario. This may be a good fit for customers with relaxed RPOs.

The added benefit of having a longer cycle time is that the locality of reference will likely increase. This is because there is a much greater chance of a track being updated more than once in a one-hour interval than in, say, a 30 second interval. The increase in locality of reference shows up as reduced bandwidth requirements for the final solution.

Before SRDF/AR can be started, instantiation of the database has to take place. In other words, a baseline full copy of all the volumes that are going to participate in the

SRDF/AR replication must be executed first. This means a full establish to the BCVs in the source array, a full SRDF establish of the BCV/R1s to the R2s and a full establish of the R2s to the BCVs in the target array is required. There is an option to automate the initial set up of the relationship.

As with other SRDF solutions, SRDF/AR does not require a host at the DR site. The commands to update the R2s and manage the synchronization of the BCVs in the remote site are all managed in-band from the production site.

For additional details, refer to Appendix A for details on locating the SRDF/AR solutions guide.

6.13.1 Restart processing

In the event of a disaster, it is necessary to determine if the most current copy of the data is located on the remote site BCVs or R2s at the remote site. Depending on when in the replication cycle the disaster occurs, the most current version could be on either set of disks. This determination is simple and is described in the SRDF/AR solutions guide.

6.14 SRDF/AR multi-hop

SRDF/Automated Replication multi-hop, or SRDF/AR multi-hop, is an architecture that allows long-distance replication with zero seconds of data loss through use of a bunker Symmetrix. Production data is replicated synchronously to the bunker Symmetrix, which is within 200 km of the production Symmetrix allowing synchronous replication but also far enough away that potential disasters at the primary site may not affect it. Typically, the bunker Symmetrix is placed in a hardened computing facility.

BCVs in the bunker frame are periodically synchronized to the R2s and consistent split in the bunker frame to provide a dependent-write consistent point-in-time image of the data. These bunker BCVs also have an R1 personality, which means that SRDF in adaptive copy mode can be used to replicate the data from the bunker array to the target site. Since the BCVs are not changing, the replication can be completed in a finite length of time. The length of time for the replication depends on the size of the “pipe” between the bunker location and the DR location, the distance between the two locations, the quantity of changed data, and the locality of reference of the changed data. On the remote Symmetrix, another BCV copy of the data is made using the R2s. This is because the next SRDF/AR iteration replaces the R2 image, in a nonordered fashion. If a disaster were to occur while the R2s were synchronizing, there would not be a valid copy of the data at the DR site. The BCV copy of the data in the remote Symmetrix is commonly called the gold copy of the data. The whole process then repeats.

With SRDF/AR multi-hop, there is minimal host impact. Writes are only acknowledged when they hit the cache of the bunker Symmetrix and a positive acknowledgment is returned to the source Symmetrix. Figure 6-7 on page 6-29 depicts the process.

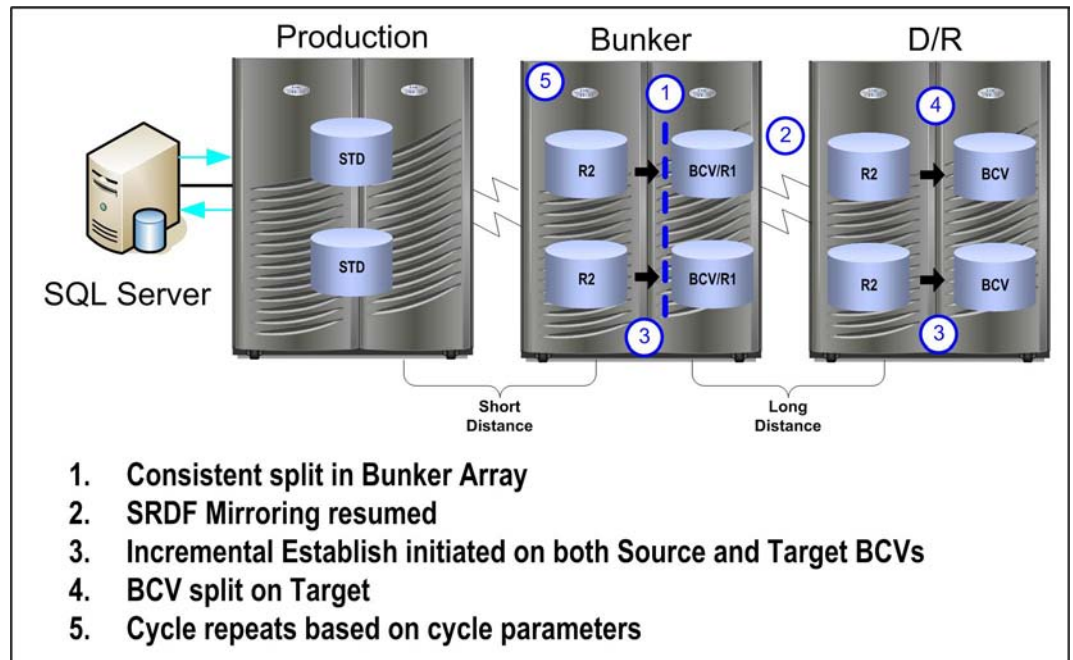


Figure 6-7 SRDF/AR multi-hop replication internals

1. BCVs are synchronized and consistently split against the R2s in the bunker Symmetrix. The write activity is momentarily suspended on the source Symmetrix to get a dependent-write consistent point-in-time image on the R2s in the bunker Symmetrix, which creates a dependent-write consistent point-in-time copy of the data on the BCVs.
2. SRDF transmits the data on the bunker BCV/R1s to the R2s in the DR Symmetrix.
3. When the BCV/R1 volumes are synchronized with the R2 volumes in the target Symmetrix, the bunker BCV/R1s are established again with the R2s in the bunker Symmetrix. This causes the SRDF links to be suspended between the bunker Symmetrix and the DR Symmetrix. At the same time, an incremental establish is performed on the DR Symmetrix to create a gold copy on the BCVs in that frame.
4. When the BCVs in the DR Symmetrix are fully synchronized with the R2s, they are split and the configuration is ready to begin another cycle.
5. The cycle repeats based on configuration parameters. The parameters can specify the cycles to begin at specific times, specific intervals or to run immediately after the previous cycle completes.

It should be noted that even though cycle times for SRDF/AR multi-hop are usually in the minutes-to-hours range, the most current data is always in the bunker Symmetrix. Unless there is a regional disaster that destroys both the primary site and the bunker site, the bunker Symmetrix will transmit all data to the remote DR site. This means zero data loss at the point of the beginning of the rolling disaster or an RPO of 0 seconds. This solution is a good fit for customers with a requirement of zero data loss and long-distance DR.

An added benefit of having a longer cycle time means that the locality of reference will likely increase. This is because there is a much greater chance of a track being updated more than once in a one-hour interval than in say a 30 second interval. The increase in locality of reference shows up as reduced bandwidth requirements for the network segment between the bunker Symmetrix and the DR Symmetrix.

Before SRDF/AR can be initiated, initial instantiation of the database has to take place. In other words, a baseline full copy of all the volumes that are going to participate in the SRDF/AR replication must be executed first. This means a full establish of the R1s in the source location to the R2s in the bunker Symmetrix. The R1s and R2s need to be synchronized continuously. Then a full establish from the R2s to the BCVs in the bunker Symmetrix, a full SRDF establish of the BCV/R1s to the R2s in the DR Symmetrix and a full establish of the R2s to the BCVs in the DR Symmetrix is performed. There is an option to automate this process of instantiation.

For additional details, refer to Appendix A for details on locating the SRDF/AR solutions guide.

6.14.1 Restart processing

In the event of a disaster, it is necessary to determine if the most current copy of the data is on the R2s on the remote site or on the BCV/R1s in the bunker Symmetrix.

Depending on when the disaster occurs, the most current version could be on either set of disks. This determination is simple and is outlined in the SRDF/AR solutions guide.

6.15 Database log-shipping solutions

Log shipping is a strategy that some companies employ for disaster recovery. The process only works for databases which are in full recovery mode, and for which incremental transaction log backups are created.

Microsoft SQL Server does implement a specific form of log shipping using SQL Server agent processes to manage the required steps. In this section, we will discuss the log shipping process itself rather than SQL Server's specific implementation.

The essence of log shipping is that changes to the database at the source site are reflected in the log are propagated to the target site. These incremental transaction log backups are then applied to a database at the target site to maintain a consistent image of the database that can be used for DR purposes. The target database is in either a NORECOVERY or STANDBY state, and will have been restored from a full backup.

6.15.1 Overview of log shipping

Log shipping is commonly used by RDBMS vendors to provide a mechanism that allows for high availability and DR of the database environment. One of the major benefits of the log shipping environment is its ability to mitigate the time and effort required to completely restore the database in the event of total loss of the production system.

Typically, a log shipping environment is created by using a full backup of the production database itself. This is restored to a separate server than the production system, and is

restored using the NORECOVERY or STANDBY form of the RESTORE Transact SQL statement. This results in a completely independent copy of the source database data and transaction files. In this way, this log shipping target system is protected from any physical damage to the data files which may potentially occur on the production system. Since the restored backup on the log shipping target system will become more out of date with respect to the state of the production data as changes occur to the production system. The log shipping environment, as the name suggests, implements a mechanism by which logged changes can be shipped from production system to target log shipping server. Those changes are any data manipulation language (DML) statements, which have been logged within the transaction log. For this reason, the full recovery mode is used for logging, such that all changes are logged and subsequently available for replay on the target server.

In the case of Microsoft SQL Server, the changes are propagated through the creation of incremental backups of the transaction log. Since all changes to the data files are recorded in the transaction log, as a result of SQL Server's implementation of Write Ahead Log (WAL) protocol, using the log records can allow for the same changes to be propagated to a full backup when played back in order. The log shipping environment replays those transaction log backups against the separate restored database and thus places that database in the same state as the production database at the point in time of the end of the last applied transaction log backup.

Incremental transaction log backups created from the production database need to be replayed against the log shipping serve in the order in which they were created. To ensure that this requirement is met, SQL Server enforces that Log Sequence Numbers (LSNs) recorded in the transaction log are played back in sequential order. SQL Server will not allow out of order restores of incremental transaction logs, and thus a reliable mechanism must be utilized to effectively unique transaction log backups.

In the event of production system total failure, once all available transaction logs have been applied, the log shipping database can be brought online and made available for use.

Utilizing the STANDBY mode of operation for the log shipping target server, allows for access in a read-only mode. During periods of time when incremental transaction logs are being applied to the standby database, no user connections are allowed, and all connections must be terminated. Using such a configuration allows for administrators to extract data from the standby instance or for users to utilize the standby instance for reporting requirements. For example, should data have been inadvertently or maliciously deleted from the production system, the log shipping target will typically be at some point behind the production system data state. If the administrator is alerted to the issue in the production system in time, it may be possible to only restore incremental log backups to the log-shipping database until just prior to the time of the logical data corruption. It would then be possible to extract the deleted data from the log shipping target database in STANDBY mode, and insert this data back into the production system.

The creation of incremental transaction logs, their shipment between the production and log shipping target server, and ultimately the application of those incremental logs can be managed manually by an administrator or by some automated processes. The Microsoft SQL Server environment provides a structured, automated process for the

creation, monitoring and ongoing processing of a log shipping environment. Information on the definition, implementation and ongoing management of the SQL Server solution is available in the SQL Server Books Online documentation.

It is possible to allow for EMC disk mirror VDI restore operations to create the target environment from a full database backup which has been processed in the appropriate manner. The target database environment must have been left in either a NORECOVERY or STANDBY mode to be integrated into the SQL Server managed log shipping environment.

To allow SQL Server management tools to utilize this restored target database, when configuring the log shipping environment via the Maintenance Plan subsystem, simply select the “Use existing database (No initialization)” option provided in the Add Destination Database dialog box, as shown in Figure 6-8.

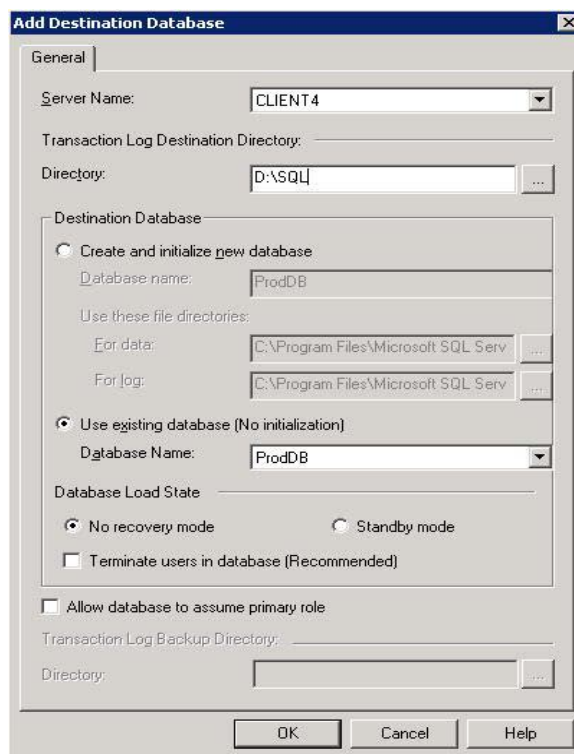


Figure 6-8 Log shipping configuration – Add Destination Database dialog

Also note in Figure 6-8 the Database Load State options presented, which allow configuration of the target database, specifically STANDBY for read only access. The Terminate users in database option relates to the STANDBY mode and ensures that exclusive access can be obtain when apply subsequent transaction logs in a STANDBY mode.

If a disaster occurred at the primary site, the target log shipping server would be able to be brought online and be made available to users, albeit with some loss of data. The amount of data loss would be determined by the processes which manage the creation of the incremental transaction log backups, and their propagation to the remote server.

6.15.2 Log shipping considerations

When considering a log shipping strategy it is important to understand:

- ◆ What log shipping covers
- ◆ What log shipping does not cover
- ◆ Exposure to data loss during periods of change in recovery modes
- ◆ Server requirements
- ◆ How to instantiate and re-instantiate the target database
- ◆ How failback works
- ◆ Federated consistency requirements
- ◆ Amount of data loss in event of site disaster
- ◆ Manageability of the solution
- ◆ Scalability of the solution

6.15.2.1 Log shipping limitations

Log shipping only transfers those changes recorded to the transaction log which are subsequently saved to an incremental transaction log backup. Consequently, operations that do not result in transaction log records are not propagated to the target server. SQL Server will limit the ability to replay logs in certain conditions depending on the version of SQL Server in use and the recovery model selected. Refer to the SQL Server Books Online for the version being used for specific information.

Log shipping is a database centric strategy. It is completely agnostic and does not address changes which occur outside of the database. These changes include, but are not limited to:

- ◆ Application and binary files (service packs)
- ◆ Database configuration changes
- ◆ Database binaries
- ◆ Operating system changes
- ◆ External flat files
- ◆ Addition of new data files on new LUNs

To sustain a working environment at the DR site, procedures must be executed to keep these objects up to date.

6.15.2.2 Server requirements

Log shipping requires a server at the remote DR site to receive and apply the logs to the standby database. It may be possible to offset this cost by using the server for other functions when it is not being used for DR. Database licensing fees for the standby database will also apply.

6.15.2.3 How to instantiate and re-instantiate the target database

Log shipping architectures need to be supported by a method of instantiating the database at the remote site. The method needs to be manageable and timely. For example, shipping 200 tapes from the primary site to the DR site may not be an adequate approach, considering the transfer time and database restore time.

Periodically, it may be necessary to re-instantiate the database at the DR site. The process should be easily managed but also should provide continuous DR protection. That is to say, there must be a contingency plan for a disaster during re-instantiation.

6.15.2.4 How failback works

An important component of any DR solution is designing a failback procedure. If the DR setup is tested with any frequency, this method should be simple and risk free. Log shipping can be implemented in reverse direction and works well when the primary site is still available. In the case of a disaster where the primary site data is lost, the database has to be re-instantiated at the production site.

To facilitate this specific process, within the SQL Server log shipping implementation, there is documentation of required procedures to implement a log shipping role change. Refer to SQL Server Books Online for details on this role change process.

6.15.2.5 Federated consistency requirements

Most databases are not isolated islands of information. They frequently have upstream inputs and downstream outputs, triggers, and stored procedures that reference other databases. There may also be a message queuing system such as Microsoft Message Queue (MSMQ), IBM MQ Series, or TIBCO queues between RDBMS environments or other data stores; externalized data stores for BLOBs, and so on. This entire environment is a federated structure that needs to be recovered to the same point in time to get a transactionally consistent disaster restart point.

Log shipping solutions are single database centric and are not adequate solutions in federated database environments.

6.15.2.6 Data loss expectations

If sufficient bandwidth is provisioned for the solution, the amount of data lost in a disaster is going to be approximately two logs worth of information. In terms of time, it would be approximately twice the length of time it takes to create an incremental transaction log backup. This time will most likely vary during the course of the day because of fluctuations in write activity.

6.15.2.7 Manageability of the solution

The manageability of a DR solution is a key to its success. Log shipping solutions have many components to manage including servers, databases, and external objects as previously noted. Some of the questions that need to be answered to make a clear determination of the manageability of a log shipping solution are:

How much effort does it take to set up log shipping?

How much effort is needed to keep it running on an ongoing basis?

What is the risk if something required at the target site is missing?

If Windows File Shares (CIFS) are used to ship the incremental transaction log backups, what kind of monitoring is needed to guarantee success?

In many of these cases, the specific SQL Server implementation mitigates the operational complexity through the implementation of SQL Agent processes and a replication monitor system, which reports on any failures.

6.15.2.8 Scalability of the solution

The scalability of a solution is directly linked to its complexity. To successfully scale the DR solution, the following questions must be addressed:

- ◆ How much more effort does it take to add more databases?
- ◆ How easy is the solution to manage when the databases grow larger?
- ◆ What happens if the quantity of updates increases dramatically?

6.15.3 Log shipping and the remote database

The remote database of a log shipping implementation is one which has been created from a restore of a full backup of the source database. The remote database will have been restored with either the NORECOVERY or STANDBY options of the RESTORE DATABASE Transact SQL Statement.

Figure 6-9 on page 6-36 depicts the SQL Server log shipping environment.

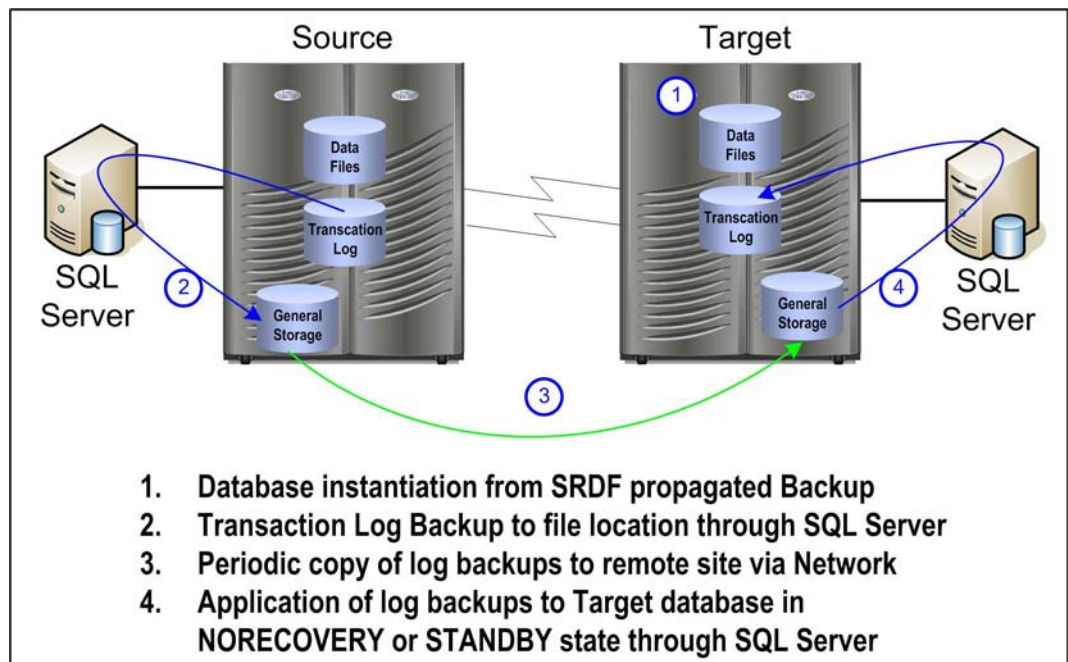


Figure 6-9 Log shipping implementation overview

1. The database must be instantiated at the D/R site, either by tape or by using SRDF, or if it is small enough, by shipping the backup over the network. The database is restored with either the NORECOVERY or STANDBY state.
2. Incremental transaction log backups are created on local storage on the production site (or on a remote file share from the D/R site).
3. If stored locally, the incremental backup logs are copied to the DR site.
4. Periodically, the logs are applied to the remote database

The remote database is continually rolled forward by the application of the incremental transaction log backups being applied to it. If the database is in a STANDBY mode, then all user connections must be terminated during the application of the transaction log backup.

The SQL Server implementation of the log shipping environment manages the creation, transmission and application of incremental transaction log backups. The steps are explained here to detail the log shipping process in general.

NORECOVERY mode logs are applied incrementally, and must be executed with the NORECOVERY keyword, as shown in Figure 6-10 on page 6-37.

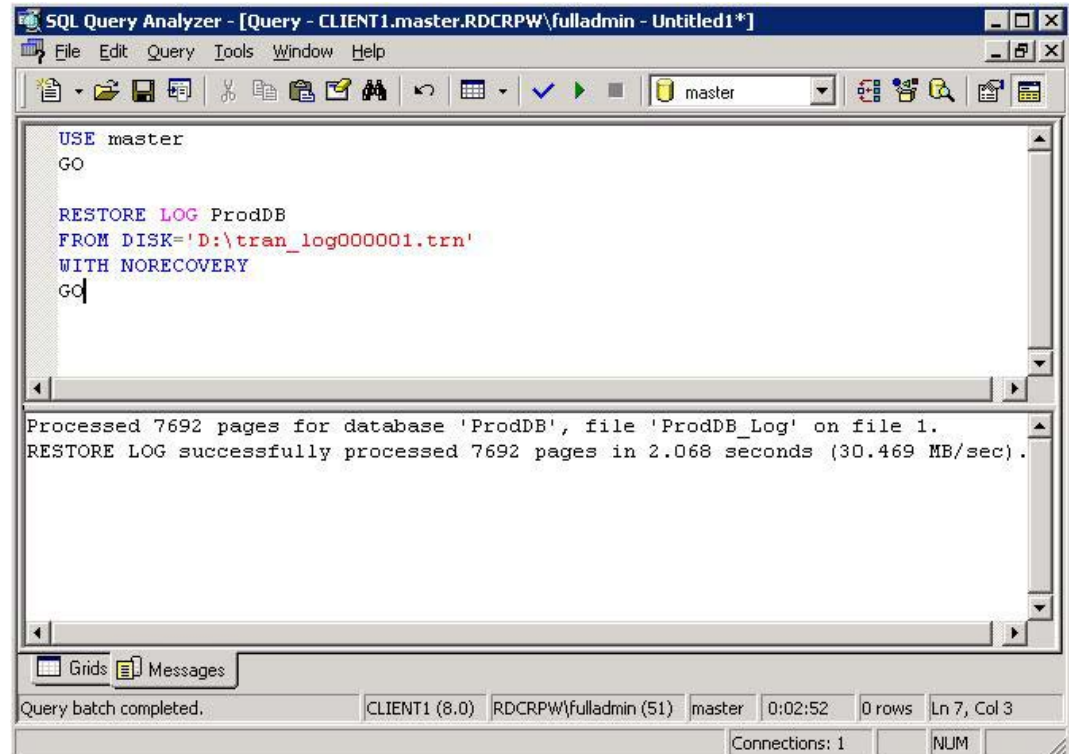


Figure 6-10 Restore of incremental transaction log with NORECOVERY

In STANDBY mode logs are applied incrementally, and must be executed with the STANDBY keyword and the specification of the standby file location, as shown in Figure 6-11 on page 6-38. SQL Server used the standby file to copy uncommitted pages from the data files and roll those data pages back to a consistent state—typically that state represented by only viewing committed updates.

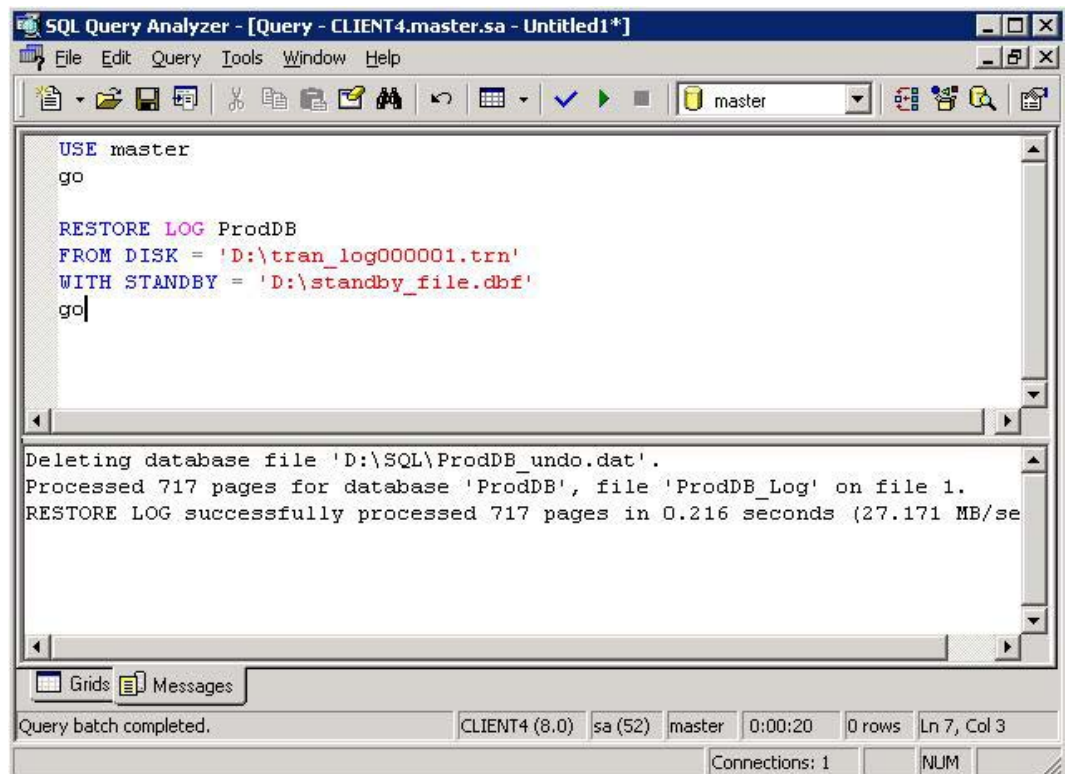


Figure 6-11 Restore of incremental transaction log with STANDBY

In the case of the STANDBY mode, before the application of subsequent incremental transaction logs, all user connections must be terminated, and the data pages stored in the standby file reapplied to their respective data files. This represents that state which existed as of the end of the application of the previous transaction log backup, and thus allows for the continuation of the application of a subsequent log.

The application of incremental transaction logs progress in the applicable manner until the last available transaction log is processed. In the case of the last incremental transaction log restore operation, the RECOVERY keyword should be used. This indicates to the SQL Server instance that final recovery processing should be executed against the database. This will roll back uncommitted transactions in the database and return it to an online state.

It is also possible to execute the following Transact-SQL against the database state, should there be no further transaction logs:

```
RESTORE LOG <database>
WITH RECOVERY
GO
```

This causes recovery processing to be executed without the need to specify a valid transaction log when no further logs are available.

6.15.4 Shipping logs with SRDF

Rather than using Windows network shares to ship the logs from the source to the target site, SRDF can be used as the transfer mechanism.

SRDF/Synchronous can be used to ship incremental transaction log backups when the Symmetrix arrays are less than 200 km apart. Synchronous mode will ensure that when a transaction log backup has been completed it is guaranteed to be available on the remote site for replay, thus mitigating some of the data loss exposure created by network transfers

In addition, federated DR requirements can be satisfied by synchronously replicating data external to the database.

The advantages to using SRDF to ship the logs follow:

- ◆ **Guaranteed delivery** — SRDF brings deterministic protocols into play, an important advantage over FTP. This means that SRDF guarantees that what is sent is what is received. If packets are lost on the network, SRDF retransmits the packets as necessary.
- ◆ **Restartability** — SRDF also brings restartability to the log shipping functionality. Should a network outage or interruption cause data transfer to stop, SRDF can restart from where it left off after the network is returned to normal operations.

While replicating the incremental log backups to the remote site, the receiving volumes (R2s) cannot be used by the host, as they are read only. If the business requirements are such that a standby database should be continually applying logs, periodically BCVs can be used to synchronize against the R2s and split. Then the BCVs can be accessed by a host at the remote site and the incremental transaction log backups retrieved and applied to the standby database

6.15.5 SQL Server Database Mirroring

Database Mirroring is a new implementation of a high availability solution for SQL Server 2005. Database Mirroring, as with SQL Server log shipping, is implemented with two database instances which are referred to as the *principal* (the production instance) and the *mirror* (the remote). It is also possible to configure a third SQL Server instance as a *witness* to facilitate quorum processing. The ability to create a quorum with the witness provides automatic failover capabilities.

Unlike log shipping, which uses a coarse granularity based on the incremental transaction log intervals, and where each backup can thus include thousands of discrete transactions, Database Mirroring works at the individual log record level and can be configured to run in a synchronous or asynchronous mode. The ability to run in synchronous mode, and thus implement a zero data loss solution is based both on the rate of update and the distance (latency) between the principal and mirror.

In its simplest form, the Database Mirroring environment can be implemented as shown in Figure 6-12 on page 6-40.

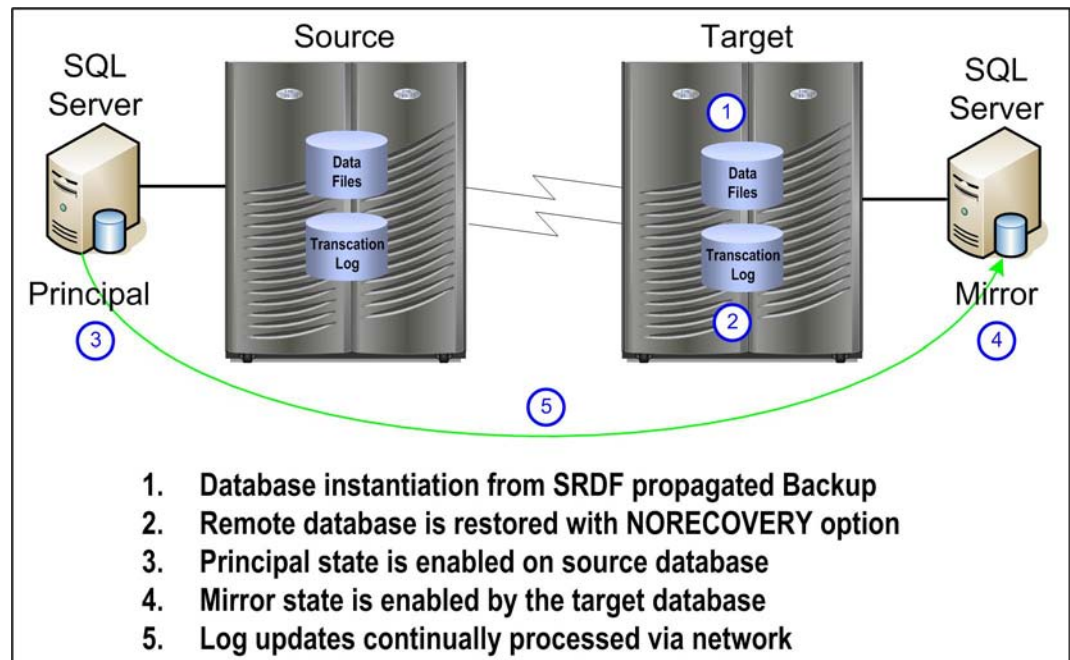


Figure 6-12 SQL Server Database Mirroring with SRDF/S

1. The database must be instantiated at the DR site, either by using some form of restore from media or by utilizing SRDF. If small enough, the database backup may be transferred via the network
2. The remote database instance is restored in a NORECOVERY mode.
3. Utilizing either the SQL Server Management Studio or the appropriate Transact-SQL statements, mirroring is enabled on the production database.
4. The remote database is also entered into a mirroring mode by utilizing the SQL Server Management Studio GUI or by using the appropriate Transact-SQL statements.
5. SQL Server Database Mirroring processing then enables the transmission of each transaction log write submitted to the local transaction log file on the production instance to be also transmitted via network connectivity to the remote mirror.

As with typical log shipping environments, the target (remote) database may require re-instantiation periodically. SRDF can be used in combination with disk mirror backup operations to facilitate this process when required. In addition, if SRDF was used in the initial creation of the remote database backup, subsequent re-instantiations will become incremental operations. This may allow for more efficient processing depending on the amount of data change in the intervening time period.

6.15.5.1 SQL Server Database Mirroring modes of operation

SQL Server 2005 Database Mirroring provides two effective forms of replication between the principal and mirror. One of SYNC or ASYNC must be selected as the operational mode. Figure 6-13 details the process flow used by the Database Mirroring implementation.

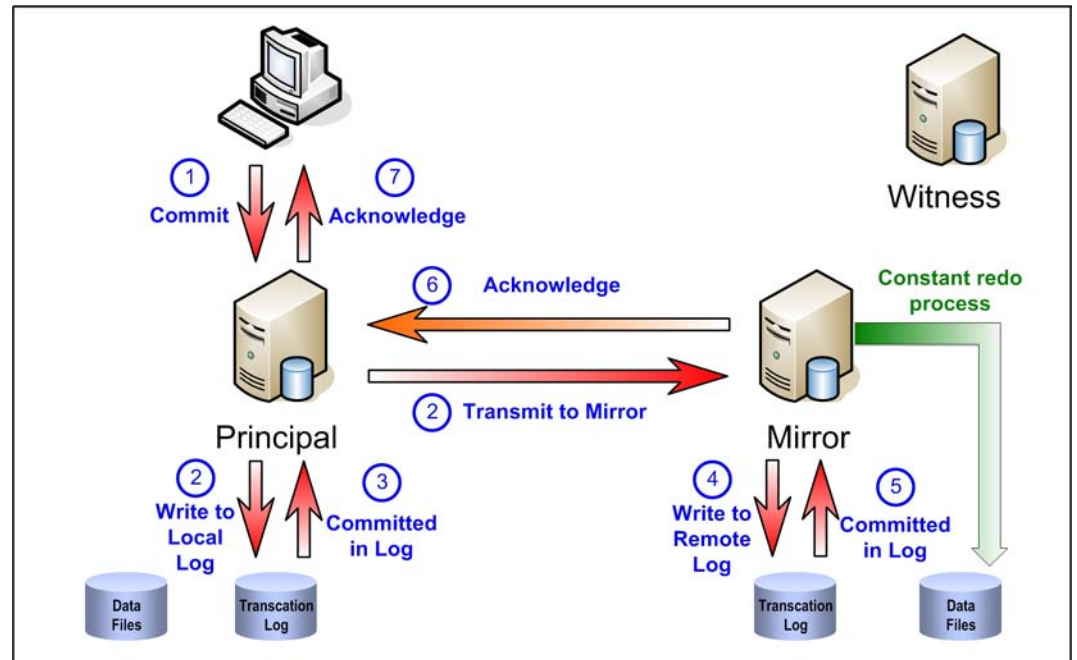


Figure 6-13 SQL Server Database Mirroring flow overview – SYNC mode

1. An operation occurs, which necessitates a log write. In this example, a commit of a transaction is executed by a user process.
2. Principal submits both a local log write operation and log write operation to mirror server.
3. Local log write I/O operation completes, and I/O is acknowledged to server
4. If in SYNC mode, then I/O completion status is not returned to calling process until log write is also executed against mirror's log device
5. Mirror log write is acknowledged to mirror server
6. Acknowledgement of successful log write operation sent back to principal
7. Principal SQL Server instance reports success to calling process.

Should the ASYNC form of operation be employed, then steps 2 through 6 are not in the critical path for log write operations. As a result, only the local log write operation is required before acknowledgement is returned to the calling process.

For environments where there is high latency on network traffic, or where the distance creates latency, SYNC mode can provide a serious impact to production operations.

The use of the ASYNC form of database mirroring attempts to mitigate performance issues, by removing the dependency on the log write of the mirror instance. It is possible that over longer latency network links, a back log of writes may occur, leading to greater data loss.

6.15.5.2 SQL Server Database Mirroring client connectivity

The SQL Client connectivity stack has been enhanced to ensure a high-availability solution in the event of principal server failure. New functionality was added to the SQL client to allow for the designation of both the principal and mirror instances. In this way, should a failover occur, the client application could automatically be redirected to the alternate server. Clearly, this functionality assumes that automatic failover operations have been implemented.

6.15.5.3 Initiation of the mirror database

The mirror database state is derived from a restore operation from a valid backup of the source production database. EMC disk mirror backup technology may be utilized to create the required state of a NORECOVERY database. Once created, standard SQL Server processes may be utilized to configure, manage, and monitor the ongoing operations.

6.15.5.4 SQL Server Database Mirroring high availability mode

The Database Mirroring solution for SQL Server 2005 provides the ability to utilize an additional server with a SQL Server 2005 instance to participate in the environment to implement additional automatic failover capabilities. This third server is referred to as the “witness.”

The witness server allows for the management of service availability by forming quorum between the three servers. At a minimum, quorum is formed by at least two of the available servers. In the event that either the principal or the mirror fail, the witness will form quorum with the remaining SQL Server instance, and in the case of forming quorum with the principal, retain ongoing service; or in the case of forming quorum with the mirror, promote the mirror to provide service as the new principal.

Additionally, protect is provided for “split-brain scenarios,” where connectivity is lost between the hosts, rather than a physical server failure. In this case, a server which is unable to form quorum with at least its partner server, or the witness will not continue (or start) service for the database. This does mean that if connectivity is lost between all three servers, then service to the production database will be lost because the principal is unable to form quorum, thus resulting in it taking the database offline. Also, the mirror is unable to form quorum with the witness, and thus is not able to promote itself to principal status.

6.15.5.5 SQL Database Mirroring failover operations

Given the nature of the replication cycle of transaction information between the principal and the mirror, it is an extremely fast operation to translate the mirror to principal state. There is no requirement to apply additional logs, since all log records will have already been recorded. The only internal recovery processes required by the database in this

state is to rollback uncommitted transactions, which is a relatively efficient process. Enhanced by the new recovery features available in SQL Server 2005, the database instance can be made instantly available to users, while the rollback operations continue. Transactional consistency for users is always marinated. Once all rollback operations complete, the instance becomes the new principal. The capabilities also exist for the old principal to assume the state of the mirror. Therefore, the flow of information is reverses, and provides a re-creation of the level of availability and protection.

Network connectivity, the latency and amount of data flow will be the primary considerations for deploying a Database Mirroring solution.

6.16 Running database solutions

Running database solutions attempt to use DR resources actively. Instead of having the database and server sitting idly waiting for a disaster to occur, the idea of having the database running and serving a useful purpose at the DR site is an attractive one. Most hardware, software replication technologies that utilize backup/restore or recovery processes typically require exclusive access to the database, not allowing users to access the target database.

The solutions presented in this section perform replication at the database transaction level or application layer and therefore allow user access even when the database is being updated by the replication process. They typically lend themselves to single database solutions, but provide the mechanisms to provision “scale-out solutions,” where multiple discrete instances provide service.

6.16.1 SQL Server transactional replication

SQL Server provides several transactional replication implementations. These replication implementations are implemented on “articles,” which are the basic unit of replication. An article may be a table, a partitioned part of a table, or other valid SQL Server objects allowed for replication between SQL Server instances.

In general, a “publication” is created for the article on the source SQL Server instance (or publisher). A “distributor” is an intermediate SQL Server instance which provides a distribution database which stores information relating to the publications. “Subscribers” are the target SQL Server instances which subscribe to the various Publications, and they may communicate with the distributor and the publisher to obtain required information regarding the publication, or in the instance of updateable or merge replications to submit transactional changes.

Replication is configurable through SQL Server Management user interfaces, and definitive information on implementation and functionality can be found in the SQL Server Books Online documentation.

6.16.1.1 Snapshot Replication

Snapshot Replication, as mentioned, forms basis from which all other forms of SQL Server Transactional replication are derived. Once the definition of the article(s) to be published is determined, SQL Server replication processes will obtain both schema

regarding the article and the data for the defined article. To provide a consistent state, a lock is obtained across the defined data for the duration of the creation of the snapshot.

The schema is obtained to ensure that the required target tables and columns may be created correctly prior to importing the data. The data is effectively exported from the source database in BulkCopy (bcp) format for SQL Server only environments. It is possible to configure an Oracle database as the destination for SQL Server replication, should an Oracle environment be configured, the data is exported as text for later import.

Once schema and data have been exported, these are copied and stored in the distributor database. As subscribers are introduced into the environment, they will be implemented with this snapshot data. As the snapshot data is static, over time it will differ from the current data within the production environment. To this end, SQL Server provides updateable snapshots. The implementation and details of updateable snapshots is beyond the scope of this document and is covered in the SQL Server Books Online documentation.

6.16.1.2 Transactional Replication

During the implementation of SQL Server replication, it is possible to define the replication to be a form of Transaction Replication. Again, this is based on the Snapshot Replication, but it further extends the implementation to allow for incremental updates to be propagated to the subscribers on a defined schedule. Updates flow from the publisher to subscribers via the distributor, and updates are not expected from the subscribers. Again, an updateable form of this implementation is available, but is not discussed here.

After the creation of the Snapshot, SQL Server replication agents extract INSERT, UPDATE and DELETE transactions from the transaction log of the publisher. These transactions are stored in the distributor and on a cyclical basis copied by the subscribers and applied. In this way, the state of the data within the snapshot is maintained in a more up to date manner.

New subscribers will be able to use the last snapshot created and all subsequent transactional updates to implement a replica of the source publication without interacting with the production system in any manner—this is the value of the distributor system.

6.16.1.3 Merge Replication

The last form of replication discussed here is that of Merge Replication. Again, this form of replication is based on the Snapshot Replication to define the initial subscriber state.

Unlike the other forms of transactional replication discussed, updates are expected and allowed on both the publisher and subscribers. All systems interact and have conflict managers implemented to ensure consistent outcomes of conflicting updates to the same data. It is typically most suitable to have well partitioned data in this form of replication to mitigate update conflicts.

Merge Replication may be implemented across all forms of SQL Server, including Mobile editions. Therefore, this form of replication lends itself very well to mobile

workers who need to collect and update information while being disconnected from the production database.

6.17 Other transactional systems

Another form of propagating data between systems exists in the form of Messaging Queuing systems. Microsoft provides the Microsoft Message Queuing (MSMQ) environment as a mechanism of passing information between source and target applications. Many other forms of Message Queuing systems are also provided by third-party companies such as TIBCO and IBM. These systems provide a method of persisting information in an appropriate store, and can subsequently be used as a mechanism to provide updates across multiple platforms. In this way, these systems may be able to maintain a common (although perhaps not synchronous, depending on configuration) state of multiple database environments. These solutions are typically customized for specific implementations.

Additionally, other forms of transactional systems such as Distributed Transaction Coordinators may be used to ensure consistent states across multiple database environments using such mechanisms as two-phase commit processing. Microsoft provides such a mechanism as its MSDTC product. Again, typically a customized implementation, solutions built on two-phase commit operations may be implemented in such a way as to ensure that multiple databases are synchronized. It should be noted that implementing two-phase commit operations across long latency links will adversely affect the performance of the source system.

When considering the usage of such Message Queuing and Distributed Transaction Coordinators, it is important to include them as a central part of any D/R solution. Data which may persist within the queuing system, or records of commit outcomes maintained in a transactional coordinator will typically be required for restart/recovery purposes in the remote site. These implementations are a key indicator of a federated environment, and are often much better suited to restart operations than recovery operations in a site failure situation.

Chapter 7 Microsoft SQL Server Database Layouts on EMC Symmetrix

This chapter presents these topics:

7.1	The performance stack.....	7-2
7.2	SQL Server layout recommendations.....	7-5
7.3	Symmetrix performance guidelines.....	7-7
7.4	RAID considerations.....	7-19
7.5	Host versus array-based striping.....	7-24
7.6	Data placement considerations.....	7-26
7.7	Other layout considerations.....	7-31
7.8	Database-specific settings for SQL Server.....	7-32

Monitoring and managing database performance should be a continuous process in all SQL Server environments. Establishing baselines and then collecting database performance statistics for comparison against those baselines are important to monitor as are performance trends. This chapter discusses the performance stack and how database performance should be managed in general. Subsequent sections, discuss Symmetrix DMX specific layout and configuration issues to help ensure the database meets the required performance levels

7.1 The performance stack

Performance tuning involves the identification and elimination of bottlenecks in the various resources that make up the system. Resources include the application, the code (SQL) that drives the application, the database, the host, and the storage. Tuning performance involves analyzing each of these individual components that make up an application, identifying bottlenecks or potential optimizations that can be made to improve performance, implementing changes that eliminate the bottlenecks or improve performance, and verifying that the change has improved overall performance. This is an iterative process and is performed until the benefits to be gained by continued tuning are outweighed by the effort required to tune the system.

Figure 7-1 on page 7-3 shows the various “layers” that need to be examined as a part of any performance analysis. The potential benefits achieved by analyzing and tuning a particular layer of the performance stack are not equal, however. In general, tuning the upper layers of the performance stack, that is, the application and SQL statements, provide a much better return on investment than tuning the lower layers, such as the host or storage layers. For example, implementing a new index on a heavily used table that changes logical access from a full table scan to index lookup with individual row selection can vastly improve database performance if the statement is run many times (thousands or millions) a day.

When tuning a SQL Server database application, developers, DBAs, system administrators, and storage administrators need to work together to monitor and manage the process. Efforts should begin at the top of the stack and address application and SQL statement tuning before moving down into the database and host-based tuning parameters. After all of these have been addressed, storage-related tuning efforts should then be performed.

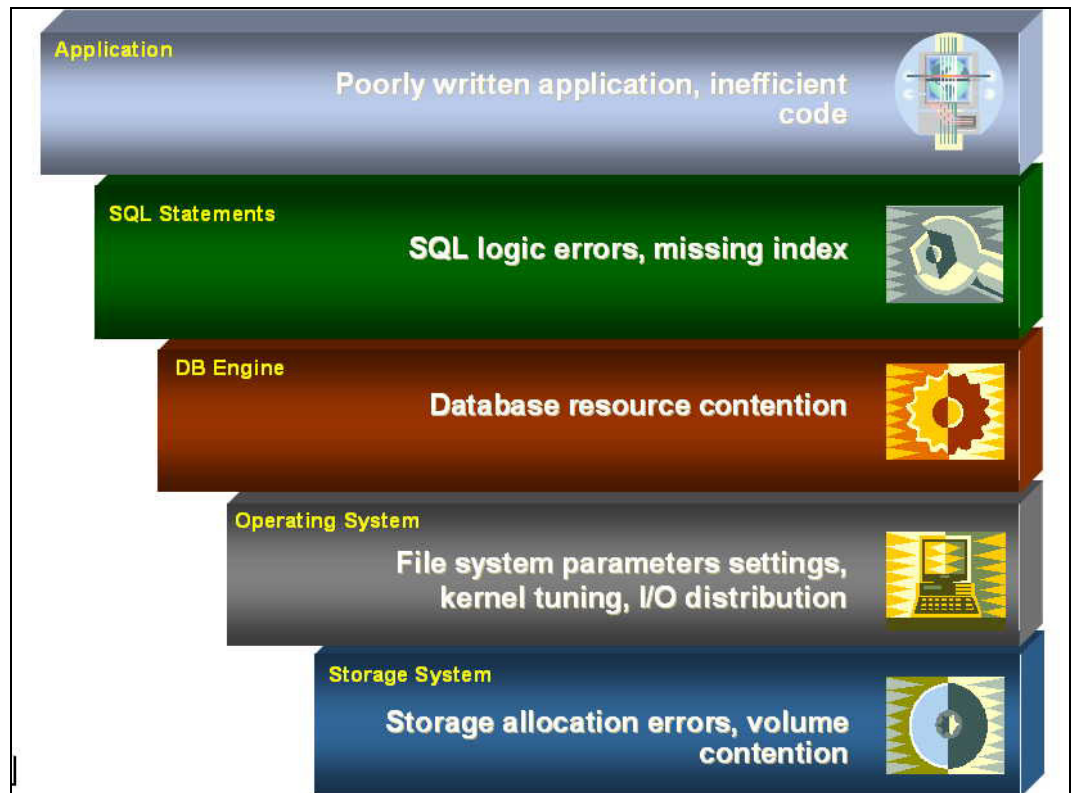


Figure 7-1 The performance stack

7.1.1 Optimizing I/O

The primary goal at all levels of the performance stack is to optimize I/O. Optimization at each level ensures that all system resources are used appropriately to service that workload, which is valid and required.

In theory, an ideal database environment is one in which most I/Os are satisfied from memory rather than going to disk to retrieve the required data. In practice, however, this is not realistic; careful consideration of the disk I/O subsystem is necessary. Poorly performing transactions, which result in major table scan operations, can adversely affect least recently used (LRU) caching mechanisms utilized by both RDBMS buffer managers, and storage array controllers. As a result, viable pages may be discarded to satisfy the need to table scan operation, only to result in a re-read of the data so that changes may be made. Furthermore, the storage array controller may also have discarded cache buffers that previously contained relevant data. It is this style of activity whose negative effects can significantly affect overall optimal operation of the system. These activities that adversely affect optimization mechanisms, such as LRU chains, create massive demands on storage subsystems, and present limited value as a standard operating policy. These types of demanding workloads may also make it extremely difficult to meet the performance guidelines provided in the next section.

Optimizing performance of a SQL Server database on an EMC Symmetrix DMX involves a detailed evaluation of the I/O requirements of the proposed application or environment. A thorough understanding of the performance characteristics and best

practices of the Symmetrix, including the underlying storage components (disks and directors, for instance) is also needed. Additionally, knowledge of complementary software products such as EMC SRDF, EMC TimeFinder, EMC Symmetrix Optimizer, and backup software, along with how utilizing these products will affect the database, is important for maximizing performance.

Ensuring optimal configuration for a given SQL Server database requires a holistic approach to application, host, and storage configuration planning. Configuration considerations for host- and application-specific parameters are beyond the scope of this document.

7.1.2 Storage system layout considerations

What is the best way to configure SQL Server environments on EMC Symmetrix DMX storage? Customers frequently ask this question. However, before recommendations can be given, details of the configuration and requirements for the database, host(s), and storage environment need to be understood. The principal goal for optimizing any layout on the Symmetrix DMX is to maximize the spread of I/O across the components of the array, reducing or eliminating any potential bottlenecks in the system. The next sections examine the tradeoffs between optimizing storage performance and manageability for SQL Server. They also discuss recommendations for laying out SQL Server databases on EMC Symmetrix DMX arrays.

At the heart of all recommendations are some measurable performance characteristics, which are typically represented as providing optimal response for the SQL Server database engine. These recommendations are in the form of response times for the various parts of a SQL Server database and are listed in Table 7-1.

Table 7-1 SQL Server data and log response guidelines

Data File Read and Write Operations	Recommendation
Response < 10 ms	Very Good
Response < 20 ms	Acceptable
Response > 20 ms	Need for investigation and improvement
Log Write Operations	
Response < 5 ms	Very Good
Response 5 to 10 ms	Acceptable
Response 10 to 15 ms	Needs improvement
Response 15 to 20 ms	Investigate and improve
Response > 20 ms	Immediate action should be initiated

7.2 SQL Server layout recommendations

Traditional best practices for database layouts focus on avoiding incidents of contention for storage-related resources. Eliminating contention involves understanding how the database manages the data flow process and ensures that concurrent or near-concurrent storage resource requests are separated on to different physical spindles. Many of these recommendations still have value in a Symmetrix DMX environment. Before examining other storage-based optimizations, a brief digression to discuss these recommendations is made.

7.2.1 File system partition alignment

It is EMC's common recommendation when utilizing LUNs presented from storage arrays to ensure that partitions created on this LUNs are aligned to 64 KB boundaries during the partition creation phase.

This partition alignment and the resulting volume alignment should not be confused with the Windows Allocation Unit size specified when formatting the volume. There are two different processes.

A number of EMC best practice documents exist on this matter and are available on Powerlink. These documents discuss the issue, and describe processes to implement corrective measures.

7.2.2 General principles for layout

There are some general recommendations that can be made for any kind of database and are thus worth mentioning here:

- ◆ Transaction logs on separate hypers and spindles. This minimizes contention for the logs as new writes come in from the database and any old transaction log

information is streamed out during incremental transaction log backups. It also isolates the sequential write and random read activity for these members from other volumes with differing access characteristics.

- ◆ Isolate TEMPDB data and log files from other user databases and optionally allocate multiple data files for TEMPDB data.
- ◆ Utilize multiple files within a filegroup to distribute I/O load. SQL Server will allow for certain parallel operations when tables have been created on multiple files. Full table scans would represent one such parallel operation.
- ◆ Implement only a single file on any given LUN. In general, this provides for the best performance configuration and may not always be possible. Windows Server and HBA configurations create device I/O queues based on LUNs. Ensuring that queue structures are scaled out, and that workloads are optimized for a LUN, will in turn result in performance scaling.

While these recommendations may be considered to be best practices, in certain circumstances they may not be possible to implement. In such cases, it is possible to colocate these files in shared locations. However, this will require constant monitoring and management to ensure that the overall performance of the environment is not suffering as a result of these competing workloads.

If it is planned to use array replication technologies such as TimeFinder and SRDF, it is prudent to organize the database data files in such a way to facilitate recovery. Since array replication techniques copy volumes at the physical disk level (as seen by the host), all data files for a given user database should be created on a set of disks dedicated to the database and should not be shared with other applications and databases. Therefore, co-locating data and log files, or other files on LUNs (viewed as physical disks by Windows servers) will affect functionality. Specifically, an attempt to restore a database located on a LUN may result in the inadvertent and erroneous restoration of other unrelated data.

This discussion is with respect to a LUN device. Windows Disk Management will allow for multiple volumes to be created on a given LUN. Restoration of disk mirror devices will restore the LUN, and therefore any and all volumes which exist on those LUNs.

It is recommended in a SQL Server environment that separate LUNs only contain volumes that contain database files from the same database, or databases which will be backed up and restored together.

The key point of any recommendation for optimizing the layout is that it is critical to understand both the type (sequential or random), size (long or short), and quantity (low, medium or high) of I/O against the various data files and other elements (logs, tempdb, and so on) of the database. Without a clear understanding of the data elements and the access patterns expected against them, serious contention issues on the back-end directors or physical spindles may arise that can negatively impact SQL Server performance. Knowledge of the application, and both data elements and access patterns is critical to ensuring high performance in the database environment.

7.2.3 SQL Server layout – replication considerations

If it is planned to use array replication technologies such as TimeFinder and SRDF, it is recommended to organize the database in such a way as to facilitate optimal performance. While changes are being made to optimize transmission of write operations over SRDF links, there are still locking mechanisms in place that may need to be addressed. In configurations such as SRDF/S, write ordering is preserved by locking the hypervolume while the update operation is transmitted to the remote array. Such processing can adversely affect the performance of I/O operations against these devices. One way to mitigate this affect is to allow for more parallel transmission of such updates.

Since it is the hypervolume and its locking that can become the bottleneck in these environments, adding more hypervolumes to support the I/O workload is one way to distribute the workload across multiple devices, and allow for parallelization across the resources.

Clearly, this becomes a tradeoff of size and performance. Moreover, larger hypervolumes will result in additional storage allocation for the database location. If this is not a suitable outcome, then it may be appropriate to create large numbers of smaller hypervolumes so the optimization for parallelism is attained and storage allocation is managed. This may need to be well planned for disk mirror backup operations to ensure that BCV devices are also sized appropriately.

7.3 Symmetrix performance guidelines

Optimizing performance for SQL Server in an EMC Symmetrix environment is very similar to optimizing performance for all applications on the storage array. In order to maximize performance, a clear understanding of the I/O requirements of the applications accessing storage is required. The overall goal when laying out an application on disk devices in the back end of the Symmetrix DMX is to reduce or eliminate bottlenecks in the storage system by spreading out the I/O across all of the array's resources. Inside a Symmetrix DMX array, there are a number of areas to consider:

- ◆ Front-end connections into the Symmetrix DMX — This includes the number of connections from the host to the Symmetrix DMX that are required, and whether front-end Fibre Channel ports will be directly connected or a SAN will be deployed to share ports between hosts.
- ◆ Memory cache in the Symmetrix DMX — All host I/Os pass through cache on the Symmetrix DMX. I/O can be adversely affected if insufficient cache is configured in the Symmetrix DMX for the environment. Also, writes to individual hypervolumes or to the array as a whole may be throttled when a threshold, known as the “write pending limit” (discussed next), is reached.
- ◆ Back-end considerations — There are two sources of possible contention in the back end of the Symmetrix – the back-end directors and the physical spindles. Proper layout of the data on the disks is needed to ensure satisfactory performance.

7.3.1 Front-end connectivity

Optimizing front-end connectivity requires an understanding of the number and size of I/Os (both reads and writes) that will be sent between the hosts and the Symmetrix DMX. There are limitations to the amount of I/O that each front-end director port, each front-end director processor, and each front-end director board can handle. Additionally, SAN fan-out counts, that is, the number of hosts that can be attached through a Fibre Channel switch to a single front-end port, need to be carefully managed.

A key concern when optimizing front-end performance is determining which of the following I/O characteristics is more important in the customer's environment:

- ◆ Input/output operations per second (IOPS)
- ◆ Throughput (MB/s)
- ◆ A combination of IOPS and throughput

In OLTP database applications, where I/Os are typically small and random, IOPS is the more important factor. In DSS applications, where transactions in general require large sequential table or index scans, throughput is the more critical factor. In some databases, a combination of OLTP and DSS like I/Os are required. Optimizing performance in each type of environment requires tuning the host I/O size.

Figure 7-2 on page 7-9 depicts the relationships between the page size of a random read request from the host, and both IOPS and the throughput needed to fulfill that request from the Symmetrix DMX. SQL Server utilizes an 8 KB page size, and it can be seen that the Symmetrix DMX provides high IOPS at this size.

Currently, each Fibre Channel port on the Symmetrix DMX is theoretically capable of 200 MB/s of throughput. In practice, however, the throughput available per port is significantly less and depends on the I/O size and on the shared utilization of the port and processor on the director. Increasing the size of the I/O from the host perspective decreases the number of IOPS that can be performed, but increases the overall throughput (MB/s) of the port. Limiting total throughput to a fraction of the theoretical maximum (100 to 120 MB/s is probably a good rule of thumb) will ensure that enough bandwidth is available for connectivity between the Symmetrix DMX and the host.

It is always recommended in SQL Server environments to provide connectivity to at least two or potentially more front-end fibre connectors to not only provide scale out by distributing workload across multiple adaptors, but also to provide high availability when used in conjunction with EMC PowerPath.

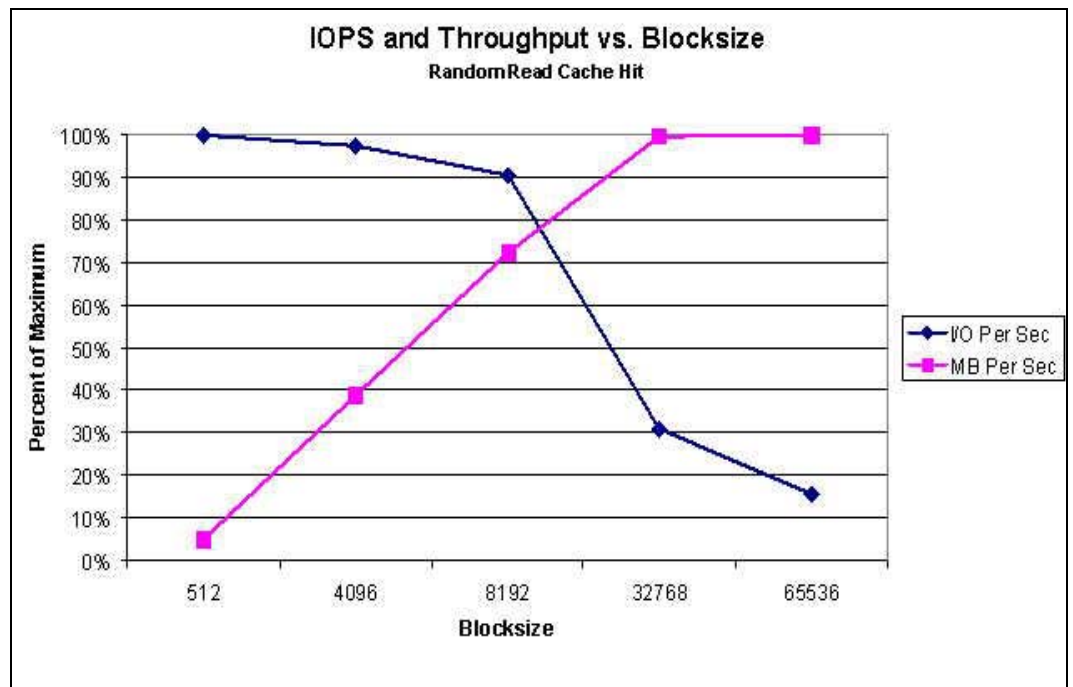


Figure 7-2 Relationship between I/O size, operations per second, and throughput

7.3.2 Symmetrix cache

The Symmetrix cache plays a key role in improving I/O performance in the storage subsystem. The cache improves performance by allowing write acknowledgements to be returned to a host when data is received in solid state cache, rather than being fully destaged to the physical disk drives. Additionally, reads benefit from cache when sequential requests from the host allow follow-on reads to be prestaged in cache. The following sections briefly describe how the Symmetrix cache is used for writes and reads, and then discuss performance considerations for it.

7.3.2.1 Write operations and the Symmetrix cache

All write operations on a Symmetrix array are serviced by cache. When a write is received by the front-end director, a cache slot must be found to service the write operation. Since cache slots are a representation of the underlying hypervolume, if a prior read or write operation caused the required data to already be loaded into cache, the existing cache slot may be used to store the write I/O. The cache slot is marked write pending if it is not already in this state.

If a cache slot representing the storage area is not currently allocated, a call is made to locate a free cache slot from the global pool, for the write. The write operation is moved to the cache slot and the slot is then marked write pending. At some later point, Enginuity™ will destage the write to physical disk. The decision of when to destage is based on overall system load, physical disk activity; read operations to the physical disk, and availability of cache.

Cache is used to service the write operation to optimize the performance of the host system. Because write operations to cache are significantly faster than physical writes to disk media, the write is reported as complete to the host operating system much more efficiently. Battery backup and priority destage functions within the Symmetrix ensure that no data loss occurs in the event of system power failure.

If the write operation to a given disk is delayed because of higher priority operations (read activity is one such operation), the write pending slot remains in cache for longer periods of time. Cache slots are allocated as needed to a volume for this purpose. Enginuity calculates thresholds for allocations to limit the saturation of cache by a single hypervolume. These limits are referred to as write pending limits.

Cache allocations are based on a per-hypervolume basis. As write pending thresholds are reached, additional allocations may occur, as well as re-prioritization of write activity. As a result, write operations to the physical disks may increase in priority to ensure that excessive cache allocations do not occur. This is discussed next in more detail.

In the manner described, the cache enables buffering of write I/Os and allows for a steady stream of write activity to service the destaging of write operations from a host. In a “bursty” write environment, this serves to even out the write activity. Should the write activity constantly exceed the low write priority to the physical disk, Enginuity will raise the priority of write operations to attempt to meet the write demand. Ultimately, if the write I/O load from the host exceeds the physical disk ability to write, the volume maximum write pending limit may be reached. In this condition, new cache slots will only be allocated for writes to a particular volume once a currently allocated slot is freed by destaging it to disk. This condition, if reached, may severely impact write operations to a single hypervolume.

7.3.2.2 Read operations and the Symmetrix cache

As discussed in the previous section, read operations typically have an elevated priority for service from the physical disks. As user processes in general need to wait for an I/O operation to complete before continuing. This is generally a good practice for storage arrays, especially those able to satisfy write operations from cache.

When a read request is received from a host system, Enginuity checks to see if a corresponding cache slot representing the storage area exists. If so, a further check is made to determine if the required data to service the read has been loaded into the cache slot. If so, the read request may be serviced immediately—this is considered a read hit.

If the cache slot has been allocated, but the data is not available, this is considered a short read miss, and a request is made to load the data from disk to the available cache slot. The read request must wait for a transfer from disk.

If a cache slot does not already exist for this storage location, but free slots are available, the read operation must wait for a cache slot to be allocated and for the transfer from disk—this is referred to as a long read miss.

Although cache slots themselves are 32 KB, a cache slot may contain only the requested data. That is, if a read request is made for an 8 KB block, then only that 8 KB block will

be transferred into the cache slot, as opposed to reading the entire 32 KB track from disk. The smallest read request unit is 4 KB.

In the DMX-3, the cache slot size increases from 32 KB to 64 KB. Sectors also increase from 4 KB to 8 KB

7.3.2.3 Symmetrix cache performance considerations

An important performance consideration is to ensure that an appropriate amount of cache is installed in the Symmetrix DMX. All I/O requests from hosts attached to the array are serviced from the Symmetrix DMX cache. Symmetrix DMX cache can be thought of as an extension to database buffering mechanisms. As such, many database application environments can benefit from additional Symmetrix DMX cache. With newly purchased arrays, appropriately sizing the cache is performed by the salesteam, based upon the number and size of physical spindles, configuration (including number and type of volumes), replication requirements (SRDF for example), and customer requirements.

Cache usage can be monitored through a number of Symmetrix DMX monitoring tools. Primary among these is ControlCenter Performance Manager (formerly known as Workload Analyzer). Performance Manager contains a number of views that analyze Symmetrix DMX cache utilization at both the hypervolume and overall system level. Views provide detailed information on specific component utilizations, including disks, directors (front end and back end), and cache utilization.

Symmetrix cache plays a key role in host I/O read and write performance. Read performance can be improved through prefetching by the Symmetrix DMX if the reads are sequential in nature. Enginuity algorithms detect sequential read activity and prestage reads from disk in cache before the data is requested. Write performance is greatly enhanced because all writes are acknowledged back to the host when they reach Symmetrix DMX cache rather than when they are written to disk. While reads from a specific hypervolume can use as much cache as is required to satisfy host requests assuming free cache slots are available, the DMX limits the number of writes that can be written to a single volume (that is, the write pending limit previously discussed). Understanding the Enginuity write pending limits is important when planning for optimal performance.

As previously discussed, the write pending limit is used to prevent high write rates to a single hypervolume from consuming all of the storage array cache for its use, at the expense of performance for reads or writes to other volumes. The write pending limit for each hypervolume is determined at system startup and depends on the number and type of volumes configured and the amount of cache available. The limit is not dependent on the actual size of each volume. The more cache available, the more write requests that can be serviced in cache by each individual volume. While some sharing of unused cache may take place (although this is not guaranteed), an upper limit of three times the initial write pending limit assigned to a volume is the maximum amount of cache any hypervolume can acquire for changed tracks. If the maximum write pending limit is reached, destaging to disk must take place before new writes can come in. This forced destaging to disk before a new write can be received into cache limits writes to that particular volume to physical disk write speeds. Forced destage of writes can significantly reduce performance to a hypervolume if the write pending limit is reached.

If performance problems with a particular volume are identified, an initial step in determining the source of the problem should include verification of the number of writes and the write pending limit for that volume.

In addition to limits imposed at the hypervolume level, there are additional write pending limits imposed at the system level. Two key cache utilization points for the DMX are reached when 40 percent and 80 percent of the cache is used for pending writes. Under normal operating conditions, satisfying read requests from a host has greater priority than satisfying write requests. However, when pending writes consume 40 percent of cache, the Symmetrix DMX then prioritizes reads and writes equally. This re-prioritization can have a profound effect on database performance. The degradation is even more pronounced if cache utilization for writes reaches 80 percent. At that point, the DMX begins a forced destage of writes to disk with discernable performance degradation of both writes and reads. If this threshold is reached, it is a clear indicator that both the cache and the total I/O on the array need to be reexamined.

Write pending limits are also established for Symmetrix metavolumes. Metavolumes are created by combining two or more individual hypervolumes into a single logical device that is then presented to a host as a single logical unit (LUN). Metavolumes can be created as concatenated or striped metavolumes. Striped metavolumes use a stripe size of 960 KB. Concatenated metavolumes write data to the first hyper in the metavolume (metahead) and fill it before beginning to write to the next member of the meta. Write pending limits for a metavolume are calculated on a member-by-member (hypervolume) basis.

Determining the write pending limit and current number of writes pending per hypervolume can be done simply using SYMCLI commands:

The following SYMCLI command returns the write pending limit for hypervolumes in a Symmetrix:

```
symcfg -sid <sid> -v list | findstr Pending
```

```
Max # of system write pending slots:162901
```

```
Max # of DA write pending slots:81450
```

```
Max # of device write pending slots:4719
```

Depending on cache availability, the maximum number of write pending slots an individual hypervolume can use is up to three times the maximum number of device write pending slots previously listed, that is, $3 * 4,719 = 14,157$ write pending tracks.

The number of write pending slots utilized by a host's devices can be found using the SYMCLI command symstat as shown next.

```
symstat -i 30
```

	DEVICE	KB/sec	KB/sec	% Hits	%Seq	Num WP	
13:09:52		READ	WRITE	READ	WRITE	RD WRT	READ Tracks

```

13:09:52 035A (Not Visible ) 0  0  0  0 N/A N/A N/A  2
      0430 (Not Visible ) 0  0  0  0 100  0 100 2679
      0431 (Not Visible ) 0  0  0  0 100  0 100 2527
      0432 (Not Visible ) 0  0  0  0 82 28  0 2444
      0434 (Not Visible ) 0  0  0  0  0 100  0 14157
      0435 (Not Visible ) 0  0  0  0  0 100  0 14157
      043A (Not Visible ) 0  0  0  0 N/A N/A N/A  49
      043C (Not Visible ) 0  0  0  0 N/A N/A N/A  54
      043E (Not Visible ) 0  0  0  0 N/A N/A N/A  15
      043F (Not Visible ) 0  0  0  0 N/A N/A N/A  10
      0440 (Not Visible ) 0  0  0  0 N/A N/A N/A  807
      0441 (Not Visible ) 0  0  0  0 N/A N/A N/A  328
      0442 (Not Visible ) 13  1 66  0  0 100  0  17
      0443 (Not Visible ) 0  0  0  0 100  0 100 1597
      0444 (Not Visible ) 0  0  0  0 N/A N/A N/A  4

```

From this, we can see that the devices 435 and 434 have reached the device write pending limit of 14,157. Further analysis on the cause of the excessive writes and methods of alleviating this performance bottleneck against these devices should be made.

Alternatively, Performance Manager may be used to determine the device write pending limit, and whether device limits are being reached. Figure 7-3 on page 7-14 is a Performance Manager view displaying both the device write pending limits and device write pending counts for a given device; this example shows Symmetrix device 055. For the Symmetrix in this example, the write pending slots per device was 9,776 and thus the max write pending limit was 29,328 slots (3 * 9776). In general, a distinct flat line in such graphs indicates that a limit has been reached.



Figure 7-3 Performance Manager graph of write pending for single hypervolume

The functionality of using metavolumes when we compare the same workload run against a 4 member striped metavolume. In Figure 7-4 on page 7-15, the hypervolumes comprising the one metavolume were used for the same workload as generated in Figure 7-3. It can be seen that each of the volumes serviced a proportion of the workload. Because the location of the file being created and the stripe depth used, one of the volumes incurred more write activity. However, even in this case, it did not exceed the lowest of the write pending thresholds, let alone reach the maximal threshold limit.



Figure 7-4 Performance Manager graph of write pending for four member striped metavolume

In the same way, the throughput and overall performance of the workload was substantially improved. Figure 7-5 on page 7-16 shows a comparison of certain metrics in this configuration. It should be obvious that this is not truly a fair comparison since we are comparing a single hypervolume against 4 hypervolumes within the metavolume. However, it does show that multiple disks are better able to satisfy an intense workload, which can clearly exceed the capability of a single device. It also serves to demonstrate the management and dynamic allocation of cache resources for volumes.

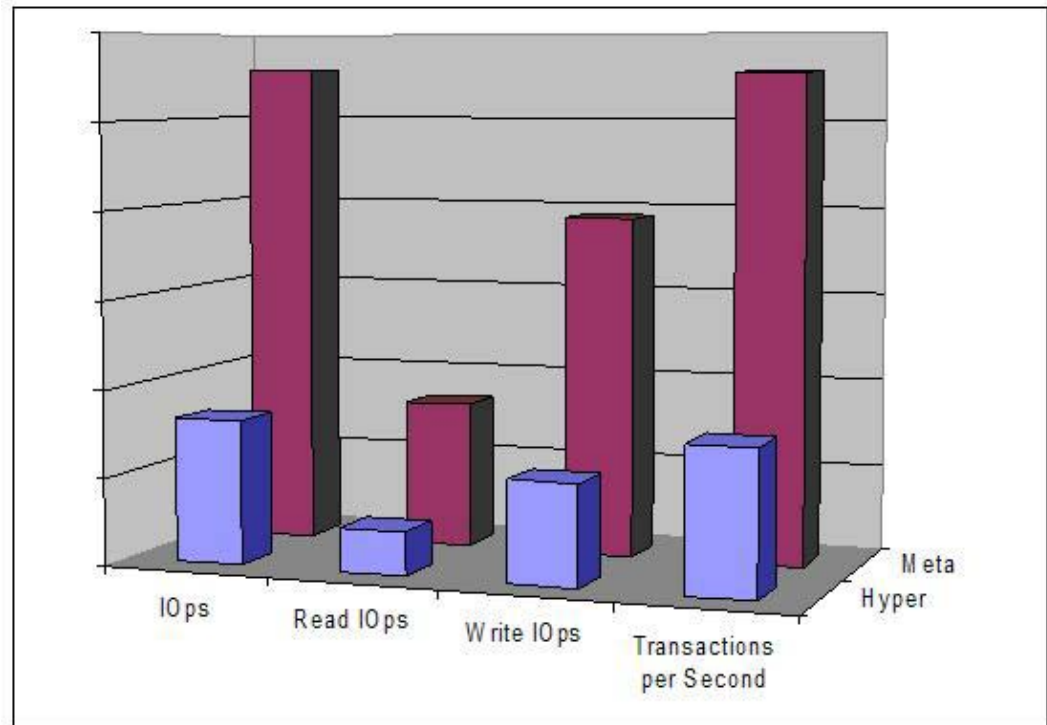


Figure 7-5 Comparison of write workload single hyper and striped metavolume

The number of cache boards can also have a minor effect on performance. When comparing Symmetrix DMX arrays that have the same amount of cache, increasing the number of boards (for example, 4 cache boards with 16 GB each as opposed to 2 cache boards with 32 GB each) has a small positive effect on the performance in DSS applications. This is because of the increased number of paths between front-end directors and cache, and has the affect of improving overall throughput. However, configuring additional boards is only helpful in high-throughput environments such as DSS applications. For OLTP workloads, where IOPS are more critical, additional cache directors provide no added performance benefits. This is because the number of IOPS per port or director depends is limited by the processing power of CPUs on each board.

In the DMX-3, write pending limits for individual volumes is modified. Instead of allowing writes up to 3x the initial write pending limit, up to ~1/20 of the cache can be utilized by any individual hypervolume.

7.3.3 Back-end considerations

Back-end considerations are typically the most important part of optimizing performance on the Symmetrix DMX. Advances in disk technologies have not kept up with performance increases in other parts of the storage array such as director and bandwidth (that is, Direct Matrix™ versus Bus) performance. Disk access speeds have increased by a factor of three to seven in the last decade while other components have easily increased one to three orders of magnitude. As such, most performance bottlenecks in the Symmetrix DMX are attributable to physical spindle limitations.

An important consideration for back-end performance is the number of physical spindles available to handle the anticipated I/O load. Each disk is capable of a limited number of operations. Algorithms in the Symmetrix DMX Enginuity operating environment optimize I/Os to the disks. Although this helps to reduce the number of reads and writes to disk, access to disk, particularly for random reads, is still a requirement to attempt to optimize I/O operations and counts for individual spindles. If an insufficient number of physical disks are available to handle the anticipated I/O workload, performance will suffer.

It is critical to determine the number of spindles required for a SQL Server database implementation based upon I/O performance requirements, and not solely on the physical space considerations.

In order to reduce or eliminate back-end performance issues on the Symmetrix DMX, take care to spread access to the disks across as many back-end directors and physical spindles as possible. EMC has long recommended for data placement of application data to “go wide before going deep”. This means that performance is improved by spreading data across the back-end directors and disks, rather than allocating specific applications to specific physical spindles. Significant attention should be given to balancing the I/O on the physical spindles. Understanding the I/O characteristics of each data file and separating high application I/O volumes on separate physical disks will minimize contention and improve performance. Implementing Symmetrix Optimizer may also help to reduce I/O contention between hypervolumes on a physical spindle. Symmetrix Optimizer identifies I/O contention on individual hypervolumes and nondisruptively moves one of the hypers to a new location on another disk. Symmetrix Optimizer is an invaluable tool in helping to reduce contention on physical spindles should workload requirements change in an environment.

Placement of data on the disks is another performance consideration. Because of the rotational properties of disk platters, tracks on the outer parts of the disk perform better than inner tracks. While the Symmetrix DMX Enginuity algorithms smooth much of this variation, small performance increases can be achieved by placing high I/O objects on the outer parts of the disk. Of more importance, however, is minimizing the seek times associated with the disk head moving between hypervolumes on a spindle. Physically locating higher I/O devices together on the disks can significantly improve performance. Disk head movement across the platters (seek time) is a large source of latency in I/O performance. By placing higher I/O devices contiguously, disk head movement may be reduced, increasing I/O performance of that physical spindle.

It is also important to understand that the Symmetrix array is designed to provide shared access to resources. As a result, items such as physical disk spindles are subdivided into hypervolumes. Multiple hypervolumes exist on any given disk spindle. Each hypervolume may either be presented to a host connected to the Symmetrix, or combined as a part of a metavolume, which is subsequently presented to a host. The key point here is that spindles are not specifically designated to a given host, but rather they can be shared amongst different hosts. The overall load and performance of any individual spindle will result from the cumulative load represented by all the workloads targeted to all hypervolumes on that spindle. Expectations should be set appropriately for the anticipated performance of these resources. Administrators should understand that their particular system may not be the only system generating workload to the

spindles, but they may be seeing the performance impact from the other hosts workloads.

7.3.4 Additional layout considerations

There are a number of additional factors that determine the best layout for a given hardware and software configuration. It is important to evaluate each of these factors to create an optimal layout for a SQL Server database.

7.3.4.1 Host bus adapters

A host bus adapter (HBA) is a circuit board and/or integrated circuit adapter that provides I/O processing and physical connectivity between a server and a storage device. The connection may route through Fibre Channel switches if Fibre Channel FC-SW is used. Because the HBA relieves the host microprocessor of both data storage and retrieval tasks, it can improve the server's performance time. An HBA and its associated disk subsystems are sometimes referred to as a disk channel.

HBAs can be a bottleneck if an insufficient number of them are provisioned for a given throughput environment. When configuring SQL Server systems estimate the throughput required and provision sufficient HBAs accordingly.

It is recommended that there be a minimum of two HBAs located in any production system. This will at a minimum provide a load sharing capability across these resources, and also provide a level of fault protection when used in combination with products such as PowerPath.

7.3.4.2 Host addressing limitations

HBAs also have limitations on how many LUNs can be addressed on a given channel. Windows HBAs in general support up to 255 logical unit devices (LUNs).

These factors must be weighed when designing the database storage infrastructure. The final architecture will always be a compromise between what is ideal and what is economically feasible within the constraints of the implementation environment.

It is recommended to utilize Symmetrix based striped metavolume LUNs to provide storage allocations for SQL Server database file locations in OLTP environments. Striped metavolumes provide a method of balancing back-end workloads because of the stripe used across member devices. This methodology attempts to ensure that a single volume does not become overly saturated with I/O activity.

7.3.5 Configuration recommendations

The key recommendations for configuring the Symmetrix DMX for optimal performance include the following:

- ◆ Understand the I/O — It is critical to understand the characteristics of the database I/O, including the number, type (read or write) size, location (that is, data files, logs), and sequentially of the I/Os. Empirical data or estimates are needed to assist in planning.

- ◆ Multiple host bus adapters — Implementing multiple Fibre HBAs zoned to separate front-end adapters on the Symmetrix can provide an efficient balanced path to the array. When used in conjunction with PowerPath, they provide a level of fault protection.
- ◆ Physical spindles — The number of disk drives in the DMX should first be determined by calculating the number of I/Os required, rather than solely based on the physical space needs. The key is to ensure that the front end needs of the applications can be satisfied by the flow of data from the back end.
- ◆ Spread out the I/O — Both reads and writes should be spread across the physical resources (front-end and back-end ports, physical spindles, hypervolumes) of the DMX. This helps to prevent bottlenecks such as hitting port or spindle I/O limits, or reaching write pending limits on a hypervolume from developing.
- ◆ Bandwidth — A key consideration when configuring connectivity between a host and the Symmetrix DMX is the expected bandwidth required to support database activity. This requires an understanding of the size and number of I/Os between the host and the Symmetrix. Connectivity considerations for both the number of HBAs and Symmetrix front-end ports are required

7.4 RAID considerations

Integrated cached storage arrays provide multiple ways to protect and manage database data. The options chosen at the array level can affect the operation of the running database. The next sections provide a brief description of RAID configurations provided by the Symmetrix DMX.

7.4.1 Types of RAID

The following defines RAID configurations that are available on the Symmetrix DMX:

- ◆ Unprotected — This configuration is not typically used in a Symmetrix DMX environment for production volumes. BCVs and occasionally R2 devices (used as target devices for SRDF) can be configured as unprotected volumes.
- ◆ RAID 1 — These are mirrored devices and are the most common RAID type in a Symmetrix DMX. Mirrored devices require writes to both physical spindles. However, intelligent algorithms in the Enginuity operating environment can use both copies of the data to satisfy read requests that are not already in the cache of the Symmetrix DMX. RAID 1 offers optimal availability and performance but at an increased cost over other RAID protection options.
- ◆ RAID 5 — A relatively recent addition to Symmetrix data protection (Enginuity 5670+), RAID 5 stripes parity information across all volumes in the RAID group. RAID 5 offers good performance and availability at a decreased cost. Data is striped using a stripe width of four tracks (128 KB). RAID 5 is configured either as RAID 5 (3+1) (75 percent usable) or RAID 5 (7+1) (87.5 percent usable) configurations. Figure 7-6 on page 7-20 shows the configuration for 3+1 RAID 5. Figure 7-21 on page 7-21 shows how a random write in a RAID 5 environment is performed.

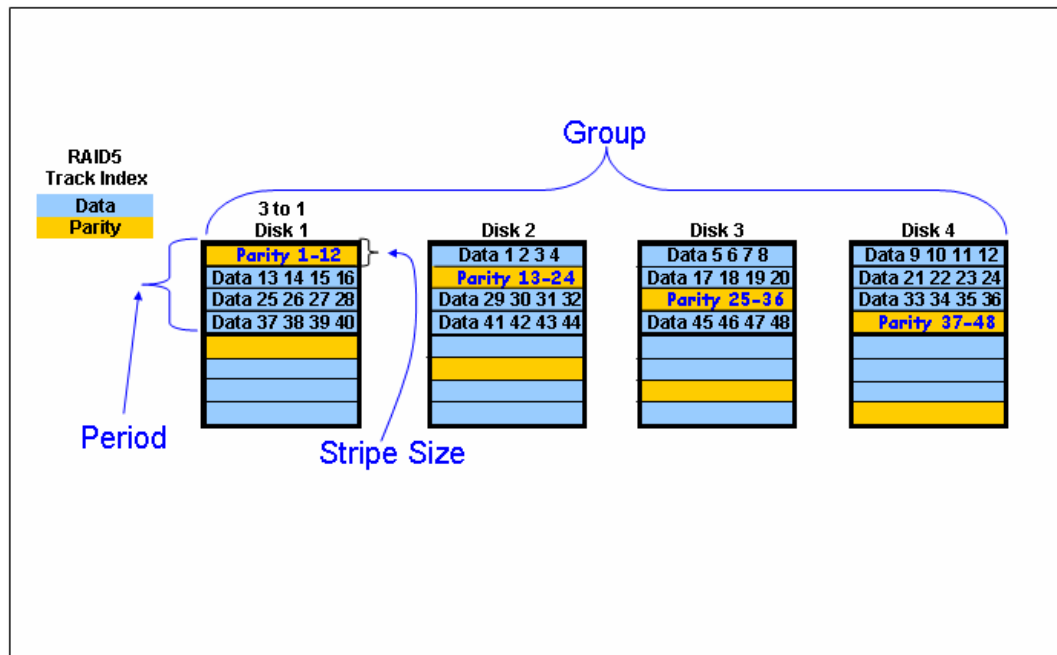


Figure 7-6 RAID 5 (3+1) layout detail

- ◆ RAID-S — Proprietary EMC RAID configuration with parity information on a single hypervolume. RAID-S functionality was optimized and renamed as parity RAID in the Symmetrix DMX.
- ◆ Parity RAID — Before the availability of RAID 5, parity RAID was implemented in storage environments that required a lower cost and did not have high-performance requirements. Parity RAID utilizes a proprietary RAID protection scheme with parity information being written on a single hypervolume. Parity RAID is configured in 3+1 (75 percent usable) and 7+1 (87.5 percent usable) configurations. Parity RAID is not recommended in current DMX configurations, where RAID 5 should be used instead.
- ◆ RAID 1/0 — These are striped and mirrored devices. This configuration is only used in mainframe environments. However, RAID 1/0 can also be configured by creating striped metavolumes, as described next.

7.4.1.1 RAID 5 write penalty

RAID 5 is generally understood to suffer from a performance penalty for write operations. The penalty is imposed as a result of recalculation of the parity information, which forms the basis of the protection scheme.

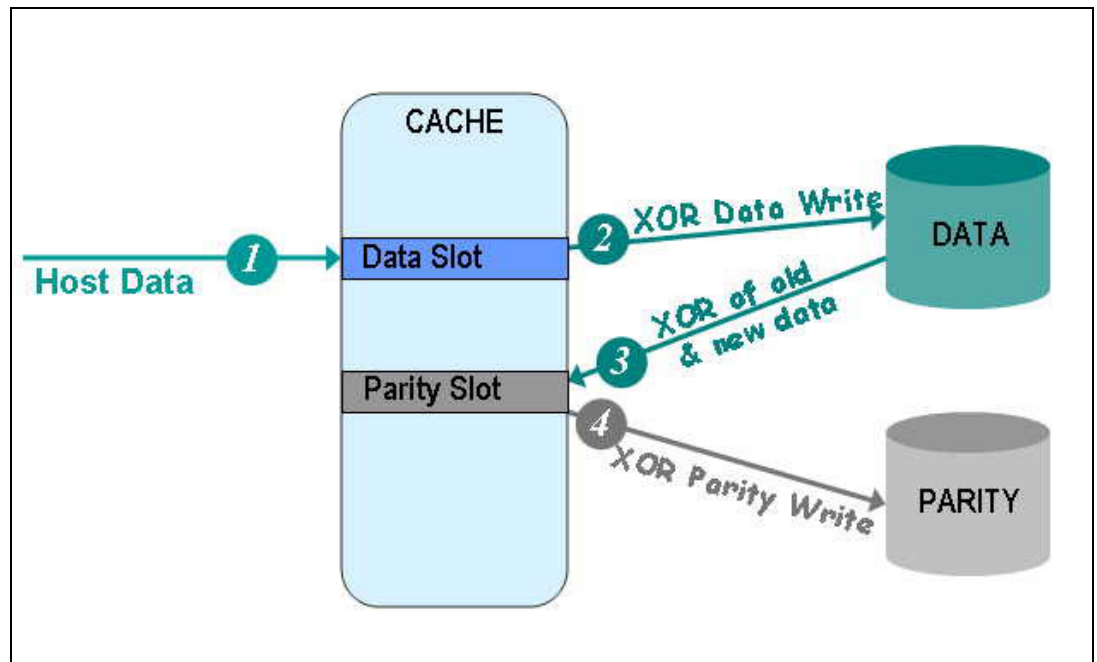


Figure 7-7 Anatomy of a RAID 5 write

The following describes the process of a random write operation to a RAID 5 volume:

1. A random write is received from the host and is placed into a data slot in cache to be destaged to disk.
2. The write is destaged from cache to the physical spindle. When received, parity information is calculated in cache on the drive by reading the old data and using an exclusive OR calculation with the new data.
3. The new parity information is written back to Symmetrix cache.
4. The new parity information is written to the appropriate parity location on another physical spindle.

It is also important to note some of the optimizations implemented within Enginuity for large sequential batch update (write) operations. As previously discussed, when random write operations are processed, there may be a requirement to generate a background read operation to be able to read and regenerate new parity information. With subsequent optimizations and when large sequential writes are being generated by the host application, Enginuity is able to calculate parity information based on the data being written. It can then write the new parity information at the same time as the data is destaged to disk. In this way, the write penalty is removed. The size of the write operation must be at least complete RAID 5 stripe, since each stripe is 128 KB, in a RAID 5 (3+1) environment, the write must be (3 x 128 KB) or 384 KB. For a RAID 5 (7+1) the write must be (7 x 128 KB) or 896 KB. Thus, large sequential write operations which may be typical of large batch updates may benefit from this optimization.

Determining the appropriate level of RAID to configure in an environment depends on the availability and performance requirements of the applications that will utilize the Symmetrix DMX. Combinations of RAID types are configurable in the Symmetrix DMX with some exceptions. For example, storage may be configured as a combination of RAID 1 and 3+1 RAID 5 devices. Combinations of 3+1 and 7+1 parity RAID or RAID 5 are currently not allowed in the same Symmetrix DMX. Likewise, mixing any types of RAID 5 and parity RAID in the same frame is not allowed.

Until recently, RAID 1 was the predominant choice for RAID protection in Symmetrix storage environments. RAID 1 provides maximum availability and enhanced performance over other available RAID protections. In addition, performance optimizations such as Symmetrix Optimizer, which reduces contention on the physical spindles by nondisruptively migrating hypervolumes, and dynamic mirror service policy, which improves read performance by optimizing reads from both mirrors, were only available with mirrored volumes, not with parity RAID devices. While mirrored storage is still the recommended choice for RAID configurations in the Symmetrix DMX, the relatively recent addition of RAID 5 storage protection provides customers with a reliable, economical alternative for their production storage needs.

RAID 5 storage protection became available with the 5670+ release of the Engenuity operating environment. RAID 5 storage protection provides economic advantages over using RAID 1, while at the same time providing high availability and performance. RAID 5 implements the standard data striping and rotating parity across all members of the RAID group (either 3+1 or 7+1). Additionally, Symmetrix Optimizer functionality is available with RAID 5 to reduce spindle contention. RAID 5 provides customers with a flexible data protection option for dealing with varying workloads and service-level requirements. With the advent of RAID 5 protection, using parity RAID in Symmetrix DMX systems is not recommended.

7.4.2 RAID recommendations

While EMC recommends RAID 1 to be the primary choice in RAID configuration for reasons of reliability and availability, SQL Server databases can be deployed on RAID 5-protected disks for many database environments, excluding those requiring support for high I/O demands and performance intensive applications. Databases used for test, development, QA, or reporting are likely candidates for using RAID 5 protected volumes.

Another potential candidate for deployment on RAID 5 storage are DSS applications. In many DSS environments, read performance greatly outweighs the need for rapid writes. This is because data warehouses typically perform loads off-hours or infrequently (once a week or month); read performance in the form of database user queries is significantly more important. Since there is no RAID penalty for RAID 5 read performance, only write performance, these types of applications are generally good candidates for RAID 5 storage deployments. Conversely, production OLTP applications typically require small random writes to the database, and as such, are generally more suited to RAID 1 storage.

An important consideration when deploying RAID 5 is disk failures. When disks containing RAID 5 members fail, two primary issues arise—performance and data availability. Performance will be affected when the RAID group operates in the degraded mode because the missing data must be reconstructed using parity and data

information from other members in the RAID group. Performance will also be affected when the disk rebuild process is initiated after the failed drive is replaced or a hot spare is disk is activated. Potential data loss is the other important consideration when using RAID 5. Multiple drive failures that cause the loss of multiple members of a single RAID group will result in loss of data. While the probability of such an event is insignificant, the potential in RAID 5 (7+1) environment is much higher than that for RAID 1. As such, the probability of data loss because of the loss of multiple members of RAID 5 group should be carefully weighed against the benefits of using RAID 5.

The bottom line in choosing a RAID type is ensuring that the configuration meets the needs of the customer's environment. Considerations include read and write performance, balancing the I/O across the spindles and the back end of the Symmetrix, tolerance for reduced application performance when a drive fails, and the consequences of losing data in the event of multiple disk failures. It should also be noted that this is not an all-or-nothing approach. It is feasible to implement a combination of RAID 5 and RAID 1 devices to support a given database, selecting the optimal configuration for each type of workload is the key consideration.

In general, EMC recommends RAID 1 for all types of customer data including SQL Server databases. However, RAID 5 configurations may be beneficial for many applications and should be considered.

7.4.3 Symmetrix metavolumes

Individual Symmetrix hypervolumes of the same RAID type (RAID 1, RAID 5) may be combined together to form a virtualized device called a Symmetrix metavolume. Metavolumes are created for a number of reasons including:

- ◆ A desire to create devices that are greater than the largest hypervolume available (in 5670 and 5671 Enginuity operating environments, this is currently just less than 31 GB per hypervolume).
- ◆ To reduce the number of volumes presented down a front-end director or to an HBA. A metavolume presented to an HBA only counts as a single LUN even though the device may comprise a large number of individual hypers.
- ◆ To increase performance of a LUN by spreading I/O across more physical spindles.

There are two types of metavolumes—concatenated or striped. With concatenated metavolumes, the individual hypers are combined to form a single volume, such that data is written to the first hypervolume sequentially before moving to the next. Writes to the metavolume start with the metahead and proceed on that physical until full, and then move on to the next hypervolume. Striped metavolumes, on the other hand, write data across all members of the device. The stripe size is set at two cylinders or 960 KB.

In nearly all cases, striped metavolumes are recommended over concatenated volumes in SQL Server database environments. The exception to this general rule occurs in specific DSS environments where metavolumes may obscure the sequential nature of the I/Os from the Enginuity prefetching algorithms.

7.5 Host versus array-based striping

Another hotly disputed issue when configuring a storage environment for maximum performance is whether to use host-based or array-based striping in SQL Server environments. Striping of data across the physical disks is critically important to database performance because it allows the I/O to be distributed across multiple spindles. Although disk drive size and speeds have increased dramatically in recent years, spindle technologies have not kept pace with host CPU and memory improvements. Performance bottlenecks in the disk subsystem can develop if the data storage requirements and configuration are not closely watched. In general, the discussion concerns tradeoffs between performance and manageability of the storage components. The following sections present in more depth the tradeoffs of using host-based and array-based striping.

7.5.1 Host-based striping

Host-based striping is configured through the Logical Volume Manager utilized on most open systems hosts. The base Windows operating system provides support for Logical Volume Management of this type when utilizing dynamic disks.

Two important things to consider when creating host-based striping are the number of disks to configure in a stripe set, and an appropriate stripe size. While no definitive answer can be given that optimizes these settings for any given configuration, the following are general guidelines to use when creating host-based stripes:

- ◆ Ensure that the stripe size used is a power of two multiple of the track size configured on the Symmetrix DMX (that is, a multiple of 32 KB for DMX-2 and 64 KB for DMX-3), the database, and host I/Os. Alignment of database blocks, Symmetrix tracks, host I/O size, and the stripe size can have considerable impact on database performance. Typical stripe sizes are 64 KB to 256 KB, although it can be as high as 512 KB or even 1 MB.
- ◆ Multiples of four physical devices for the stripe width are generally recommended, although this may be increased to eight or 16 as required for LUN presentation or SAN configuration restrictions as needed. Take care with RAID 5 metavolumes to ensure that members do not end up on the same physical spindles (a phenomenon known as vertical striping) because this may adversely affect performance. In general RAID 5 metavolumes are not recommended.
- ◆ When configuring in an SRDF environment, smaller stripe sizes (32 KB for example), particularly for the active logs, are recommended. This is to enhance performance in synchronous SRDF environments because of the limit of having only one outstanding I/O per hypervolume on the link.
- ◆ Data alignment (that is, along block boundaries) can play a significant role in performance, particularly in Windows environments. Refer to operating system-specific documentation to learn how to align data blocks from the host along Symmetrix DMX track boundaries.

7.5.2 Symmetrix-based striping (metavolumes)

An alternative to using host-based striping is to stripe at the Symmetrix DMX level. Striping in the Symmetrix is accomplished through the use of striped metavolumes, as discussed in the previous section. Individual hypervolumes are selected and striped together, forming a single LUN that is presented through the front-end director to the host. At the Symmetrix level, all writes to this single LUN are striped. Currently, the only stripe size available for a metavolume is 960 KB. It is possible to create metavolumes with up to 255 hypervolumes, although in practice metavolumes are usually created with four to sixteen devices.

7.5.3 Striping recommendation

Determining the appropriate striping method depends on a number of factors. In general, striping is a tradeoff between manageability and performance. With host-based striping, CPU cycles are used to manage the stripes—Symmetrix metavolumes require no host cycles to stripe the data. This small performance decrease in host-based striping is offset, however, by the fact that each device in a striped volume group maintains an I/O queue, thereby increasing performance over a Symmetrix metavolume, which only has a single I/O queue on the host. EMC has performed tests that have shown that striping at the host level provides somewhat better performance than comparable Symmetrix based striping, and is generally recommended if performance is paramount. Host-based striping may also be recommended with environments utilizing synchronous SRDF, since stripe sizes in the host can be tuned to smaller increments than is currently available with Symmetrix metavolumes, thereby increasing performance. This host-based striping may be most applicable for write intensive files such as the transaction log.

Management considerations however, generally favor Symmetrix-based metavolumes over host-based stripes. In many environments, customers have achieved high-performance back-end layouts on the Symmetrix by allocating all of the storage as four-way striped metavolumes. This has the advantage that any volume selected for host data is always striped, with reduced chances for contention on any given physical spindle. Additional storage requirements for any host volume group, since additional storage is configured as a metavolume, are also striped. Management of added storage to an existing volume group utilizing host-based striping may be significantly more difficult, requiring in some cases a full backup, reconfiguration of the volume group, and restore of the data in order to successfully expand the stripe.

An alternative in Windows environments gaining popularity recently is the combined use of both host-based and array-based striping. Known as “double striping” or a “plaid,” this configuration utilizes striped metavolumes in the Symmetrix array, which are then presented to a volume group and striped at the host level. This has many advantages in database environments where read access is small and highly random in nature. Since I/O patterns are pseudo random, access to data is spread across a large quantity of physical spindles, thereby decreasing the probability of contention on any given disk. Double striping, in some cases, can interfere with data prefetching at the

Symmetrix DMX level when large, sequential data reads are predominant—this configuration may not be appropriate for DSS workloads.

Another method of double striping the data is through the use of Symmetrix metavolumes and RAID 5. A RAID 5 hypervolume stripes data across either four or eight physical disks using a stripe size of 128 KB. Striped metavolumes stripe data across two or more hypers using a stripe size of 960 KB. When using striped metavolumes in conjunction with RAID 5 devices, ensure that members do not end up on the same physical spindles because this will adversely affect performance. In some cases, double striping using this method may also affect prefetching for long, sequential reads.

While host-based, array-based, or double striping in a storage environment have positive and negative factors, the important thing is to ensure that some form of striping is used for the storage layout. The appropriate layer for disk striping can have a significant impact on the overall performance and manageability of the database system. Deciding which form of striping to use depends on the specific nature and requirements of the database environment in which it is configured.

With the advent of RAID 5 data protection in the Symmetrix DMX, an additional option of triple striping data using RAID 5, host-based striping, and metavolumes combined is now available. However, triple striping increases data layout complexity, and in testing has shown no performance benefits over other forms of striping. In fact, it has actually been shown to be detrimental to performance. As a result, is not recommended in any Symmetrix DMX configuration.

7.5.3.1 Host-based striping and limitations

When considering the usage of host-based striping in a Windows environment, it is important to understand the limitations that this may impose. Windows Server is provided with a built-in Logical Volume Manager, which supports the usage of dynamic disks. It is only when using dynamic disks that host-based striping mechanisms may be implemented.

Using the base dynamic disk format in the base Windows Logical Volume Management is prohibited in environments such a Windows Cluster configurations. And for these environments a third-party volume manager is required.

Additionally, certain EMC products and utilities and software products do not interoperate with the base Windows dynamic volume implementation. It is important to check the requirements of the proposed solution to ensure that the ultimate configuration provides support for the various software products. For EMC products, consult the relevant product release notes and guide for detailed information.

7.6 Data placement considerations

Placement of the data on the physical spindles can potentially have a significant impact on SQL Server database performance. Placement factors that affect database performance include:

- ◆ Hypervolume selection for specific database files on the physical spindles themselves.
- ◆ The spread of database files across the spindles to minimize contention.
- ◆ The placement of high I/O devices contiguously on the spindles to minimize head movement (seek time).
- ◆ The spread of files across the spindles and back-end directors to reduce component bottlenecks.

Each of these factors is discussed next.

7.6.1 Disk performance considerations

As shown in Figure 7-8 on page 7-28, there are five main considerations for individual spindle performance.

- ◆ **Actuator Positioning (seek time)** — This is the time it takes the actuating mechanism to move the heads from their present position to a new position. This delay averages a few milliseconds in length and depends on the type of drive. For example, a 15k drive has an average seek time of approximately 3.5 ms for reads and 4 ms for writes. The full disk seek of 7.4 ms for reads and 7.9 ms for writes.

Disk drive characteristics can be found at appropriate disk vendor websites such as <http://www.seagate.com>.

- ◆ **Rotational speed** — This is because of the need for the platter to rotate underneath the head to correctly position the data that needs to be accessed. Rotational speeds for spindles in the Symmetrix DMX range from 7,200 rpm to 15,000 rpm. The average rotational delay is the time it takes for one half of a revolution of the disk. In the case of a 15k drive, this would be about 2 milliseconds.
- ◆ **Interface speed** — This is a measure of the transfer rate from the drive into the Symmetrix cache. It is important to ensure that the transfer rate between the drive and cache is greater than the drive's rate to deliver data. Delay caused by this is typically a very small value, on the order of a fraction of a millisecond.
- ◆ **Areal density** — This is a measure of the number of bits of data that fits on a given surface area on the disk. The greater the density, the more data per second can be read from the disk as it passes under the disk head.
- ◆ **Cache capacity and algorithms** — Newer disk drives have improved read and write algorithms, as well as cache, to improve the transfer of data in and out of the drive and to make parity calculations for RAID 5.

Delay caused by the movement of the disk head across the platter surface is called seek time. The time associated with a data track rotating to the required location under the disk head is referred to as rotational delay. The cache capacity on the drive, disk algorithms, interface speed, and the areal density (or zoned bit recording) combines to produce a disk transfer time. Therefore, the time it takes to complete an I/O (or disk

latency) consists of these three elements—the seek time, the rotational delay, and the transfer time.

Data transfer times are typically on the order of fractions of a millisecond. Therefore, rotational delays and delays because repositioning the actuator heads are the primary sources of latency on a physical spindle. Additionally, rotational speeds of disk drives have increased from top speeds of 7,200 rpm up to 15,000 rpm, but still average on the order of a few milliseconds. The seek time continues to be the largest source of latency in disk assemblies when using the entire disk.

Transfer delays are lengthened in the inner parts of the drive—more data can be read per second on the outer parts of the drive than by data located on the inner regions. Therefore, performance is significantly improved on the outer parts of the disk. In many cases, performance improvements of more than 50 percent can sometimes be realized on the outer cylinders of a physical spindle. This performance differential typically leads customers to place high I/O objects on the outer portions of the drive.

While placing high I/O objects such as active logs on the outer edges of the spindles has merit, performance differences across the drives inside the Symmetrix DMX are significantly smaller than the stand-alone disk characteristics would attest. Enginuity operating environment algorithms, particularly the algorithms that optimize ordering of I/O as the disk heads scan across the disk, greatly reduce differences in hypervolume performance across the drive. Although this smoothing of disk latency may actually increase the delay of a particular I/O, overall performance characteristics of I/Os to hypervolumes across the face of the spindle will be more uniform.

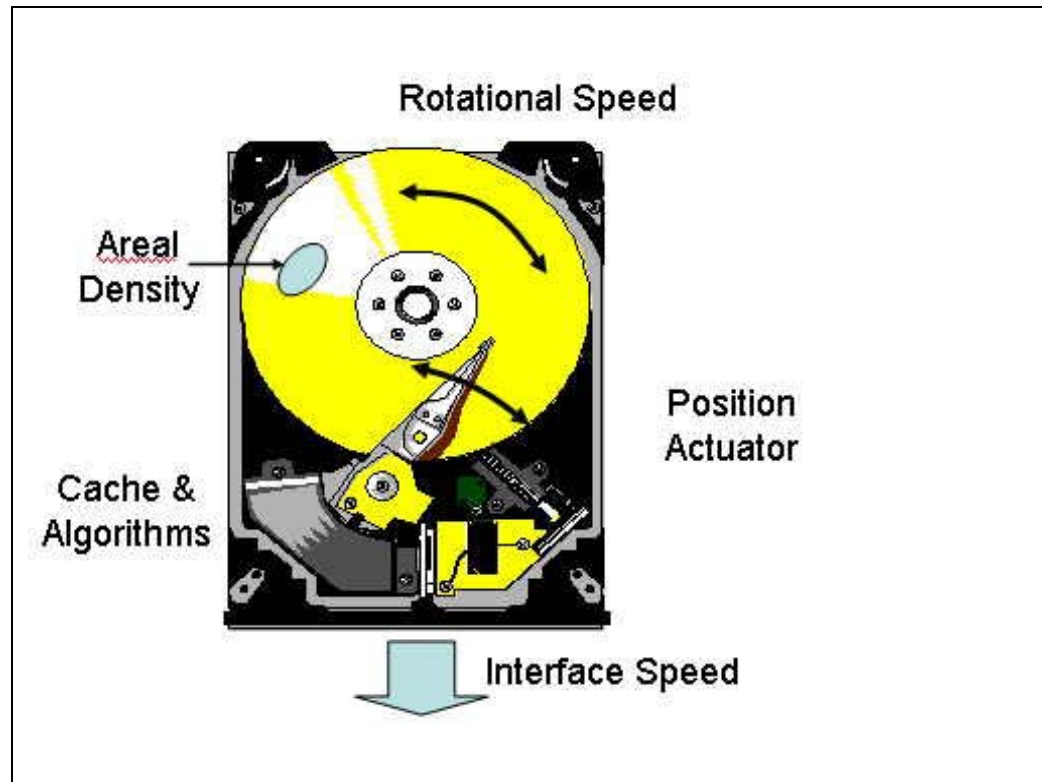


Figure 7-8 Disk performance factors

7.6.2 Hypervolume contention

Disk drives are capable of receiving only a limited number of read or write I/Os before performance degradation occurs. While disk improvements and cache, both on the physical drives and in disk arrays, have improved disk read and write performance, the physical devices can still become a critical bottleneck in SQL Server database environments. Eliminating contention on the physical spindles is a key factor in ensuring optimal SQL Server performance on Symmetrix DMX arrays.

Contention can occur on a physical spindle when I/O (read or write) to one or more hypervolumes exceeds the I/O capacity of the disk. While contention on a physical spindle is undesirable, this type of contention can be rectified by migrating high I/O data onto other devices with lower utilization. This can be accomplished using a number of methods, depending on the type of contention that is found. For example, when two or more hypervolumes on the same physical spindle have excessive I/O, contention may be eliminated by migrating one of the hypervolumes to another, lower utilized physical spindle. This could be done through processes such as LVM mirroring at the host level or by using tools such as EMC Symmetrix Optimizer to nondisruptively migrate data from impacted devices. One method of reducing hypervolume contention is careful layout of the data across the physical spindles on the back-end of the Symmetrix. Other methods of reducing contention are to use striping, either at the host level or inside the Symmetrix.

Hypervolume contention can be found in a number of ways. SQL Server specific data collection and analysis tools such as the database SNAP feature, as well as host server tools can identify areas of reduced I/O performance in the database. Additionally, EMC tools such as Performance Manager can help to identify performance bottlenecks in the Symmetrix DMX array. Establishing baselines of the system and proactive monitoring are essential in helping to maintain an efficient, high-performance database.

Commonly, tuning database performance on the Symmetrix system is performed post-implementation. This is unfortunate because with a small amount of upfront effort and detailed planning, significant I/O contention issues could be minimized or eliminated in a new implementation. While detailed I/O patterns of a database environment are not always well known, particularly in the case of a new system implementation, careful layout consideration of a database on the back end of the Symmetrix can save time and future effort in trying to identify and eliminate I/O contention on the disk drives.

7.6.3 Maximizing data spread across back-end devices

A long-standing data layout recommendation at EMC has been “go wide before going deep”. What this means is that data placement on the Symmetrix DMX should be spread across the back-end directors and physical spindles before locating data on the same physical drives. By spreading the I/O across the back end of the Symmetrix, I/O bottlenecks in any one array component can be minimized or eliminated.

Considering recent improvements in the Symmetrix DMX component technologies such as CPU performance on the directors and the Direct Matrix architecture, the most common bottleneck in new implementations is with contention on the physical spindles and the back-end directors. To reduce these contention issues, a detailed examination of the I/O requirements for each application that will utilize the Symmetrix storage should

be made. From this analysis, a detailed layout that balances the anticipated I/O requirements across both back-end directors and physical spindles should be made.

Before data is laid out on the back end of the DMX, it is helpful to understand the I/O requirements for each of the file systems or volumes that are being laid out. Many methods for optimizing layout on the back-end directors and spindles are available. One time-consuming method involves creating a map of the hypervolumes on physical storage, including hypervolume presentation by director and physical spindle, based upon information available in EMC ControlCenter. This involves documenting the environment using a tool such as Excel, with each hypervolume marked on its physical spindle and disk director. Using this map of the back end and volume information for the database elements, preferably categorized by I/O requirement (high/medium/low, or by anticipated reads and writes), the physical data elements and I/Os can be evenly spread across the directors and physical spindles.

This type of layout can be extremely complex and time-consuming. Additional complexity is added when RAID 5 hypers are added to the configuration. Since each hypervolume is really placed on either four or eight physical volumes in RAID 5 environments, trying to uniquely map out each data file or database element is beyond what most customers feel provides value. In these cases, one alternative is to rank each of the database elements or volumes in terms of anticipated I/O. Once ranked, each element may be assigned a hypervolume in order on the back end. Since BIN file creation tools almost always spread contiguous hypervolume numbers across different elements of the back end, this method of assigning the ranked database elements usually provides a reasonable spread of I/O across the spindles and back-end directors in the Symmetrix DMX. Combined with Symmetrix Optimizer, this method of spreading the I/O is normally effective in maximizing the spread of I/O across DMX components.

7.6.4 Minimizing disk head movement

Perhaps the key performance consideration controllable by a customer when laying out a database on the Symmetrix DMX is minimizing head movement on the physical spindles. Head movement is minimized by positioning high I/O hypervolumes contiguously on the physical spindles. Disk latency caused by interface or rotational speeds cannot be controlled by layout considerations. The only disk drive performance considerations that can be controlled are the placement of data onto specific, higher performing areas of the drive (discussed in a previous section), and the reduction of actuator movement by trying to place high I/O objects in adjacent hypervolumes on the physical spindles.

One method, described in the previous section, describes how volumes can be ranked by anticipated I/O requirements. Utilizing a documented map of the back-end spindles, high I/O objects can be placed on the physical spindles, grouping the highest I/O objects together. Recommendations differ as to whether placing the highest I/O objects together on the outer parts of the spindle (that is, the highest performing parts of a physical spindle) or in the center of a spindle are optimal. Since there is no real consensus and any definitive answer to this question is likely to be “it depends,” the historical recommendation of putting high I/O objects together on the outer part of the spindle is still a reasonable suggestion. Placing these high I/O objects together on the outer parts of the spindle should help to reduce disk actuator movement when doing reads and

writes to each hypervolume on the spindle, thereby improving a controllable parameter in any data layout exercise.

7.7 Other layout considerations

Besides the layout considerations described in previous sections, a few additional factors may be important to DBAs or storage administrators who want to optimize database performance and general maintenance operations. Some additional configuration factors to consider include:

- ◆ Implementing SRDF/S for the database.
- ◆ Creating database clones using TimeFinder/Mirror or TimeFinder/Clone.
- ◆ Database clones using TimeFinder/Snap.

These additional layout considerations are discussed in the next sections.

7.7.1 Database layout considerations with SRDF/S

There are two primary concerns that must be considered when SRDF/S is implemented. The first is the inherent latency added for each write to the database. Latency occurs because each write must be first written to both the local and remote Symmetrix caches before the write can be acknowledged to the host. This latency must always be considered as a part of any SRDF/S implementation. Because the speed of light cannot be circumvented, there is little that can be done to mitigate this latency.

The secondary consideration is more amenable to DBA mitigation. Each hypervolume configured in the Symmetrix is only allowed to send a single update I/O across the SRDF link. Performance degradation results when multiple I/Os are written to a single hypervolume, since subsequent writes must wait for predecessors to complete. Striping at the host level can be particularly helpful in these situations. Utilizing a smaller stripe size (32 KB to 128 KB) ensures that larger writes will be spread across multiple hypervolumes, reducing the chances for SRDF to serialize writes across the link.

7.7.2 Database cloning, TimeFinder, and sharing spindles

Database cloning is useful when DBAs wish to create backup or other business continuance images of a database. A common question when laying out a database is whether BCVs should share the same physical spindles as the production volumes or whether the BCVs should be isolated on separate physical disks. There are pros and cons to each of the solutions; the optimal solution generally depends on the anticipated workload.

The primary benefit of spreading BCVs across all physical spindles is performance. By spreading I/Os across more spindles, there is a reduced chance of developing bottlenecks on the physical disks. Workloads that utilize BCVs, such as backups and reporting databases, may generate high I/O rates. Spreading this workload across more physical spindles may significantly improve performance in these environments.

The main drawbacks to spreading BCVs across all spindles in the Symmetrix are that synchronization may cause spindle contention during resynchronization, and that BCV workloads may negatively impact production database performance. When re-synchronizing the BCVs, data is read from the production hypervisors and copied into cache. From there, it is destaged to the BCVs. When the physical disks share production and BCVs, the synchronization rates can be greatly reduced because of increased seek times due to the conflict between reading from one part of the disk and writing to another. The other drawback to sharing physical disks is the increased workload on the spindles that may impact performance on the production volumes. Sharing the spindles increases the chance that contention may arise, decreasing database performance.

Determining the appropriate location for BCVs, sharing the same physical spindles or isolated on their own disks, depends on customer preference and workload. In general, it is recommended that the BCVs share the same physical spindles. However, in cases where the BCV synchronization and utilization may negatively impact applications (such as, databases that run 24-by-7 with high I/O requirements), it may be beneficial for the BCVs to be isolated on their own physical disks.

7.7.3 Database clones using TimeFinder/Snap

TimeFinder/Snap provides many of the benefits of full volume replication techniques such as TimeFinder/Mirror or TimeFinder/Clone, but at greatly reduced costs. However, two performance considerations must be taken into account when using TimeFinder/Snap to make database clones for backups or other business continuous functions. These include Copy on First Write (COFW) and Copy on Access (COA) penalties. The first affects the production volume performance while the second affects access to the snap volumes.

The first of these penalties, COFW, comes about because of the need for data to be copied from the production hypervisors to the save area as writes come into them. TimeFinder/Snap uses virtual devices

7.8 Database-specific settings for SQL Server

Microsoft SQL Server is a dynamically tuning database environment. Most interaction with the disk storage system is entirely managed automatically by SQL Server and thus no tuning options are provided.

Issues, which may affect the overall performance of a SQL Server database environment when used on a Symmetrix array, are discussed next.

7.8.1 Remote replication performance considerations

As previously discussed, Synchronous replication of storage on which SQL Server data and transaction log files reside can introduce increased latency. It may also be important to monitor the setting of the Recovery Time Interval.

The Recovery Time Interval controls the anticipated amount of time it takes to resolve transactional state for the database in the event of production server failure. Specifically, this setting will affect any Recovery Time Objective. It does not affect the

Recovery Point Objective, since the use of SRDF/S and the SQL Server Write Ahead Log will guarantee a zero data loss solution (RPO = zero).

With smaller Recovery Time Intervals, the checkpoint mechanisms of the SQL Server database will become more aggressive, generating additional write activity to the data files. In higher latency SRDF/S environments this may have a negative effect on overall performance.

Also note that increasing the Recovery Time Interval may result in a dirty buffer pool size increase, since dirty pages will be allowed to persist in memory for longer periods of time. However, lazy writer operations will continue, and thus on an ongoing basis, dirty pages will be flushed to disk, although not as aggressively as by the checkpoint mechanism.

The Recovery Time Interval may be configured by the `sp_configure` Transact-SQL stored procedure. More information for changing this setting may be found in the SQL Server Books Online documentation.

7.8.2 TEMPDB storage and replication

The TEMPDB system database is utilized by SQL Server to provide semi-persistent storage for certain operations. The amount of activity to the TEMPDB data and log files may be significant in certain environments, depending on the style of Transact-SQL statements being executed and certain SQL Server functionality such as the new isolation levels provide by SQL Server 2005.

SQL Server recreates the TEMPDB storage when a new instance of the SQL Server environment is initiated. Thus, there is no viable persistent data stored within TEMPDB. Consequently, there is no value in replicating the storage space allocated for TEMPDB data and log files over SRDF links. It is important, however, to have similarly allocated TEMPDB configurations on the target storage, such that in the event of site failover, performance is not adversely affected due to insufficient allocation of TEMPDB space.

In geographically dispersed clustering solutions such as SRDF/CE for MSCS, the replication of TEMPDB space is unavoidable because all storage space allocation must be shared.

Appendix A Related Documents

This appendix presents these topics:

A.1	Related documents.....	A-2
-----	------------------------	-----

A.1 Related documents

The following is a list of related documents that may assist readers with more detailed information on topics described in this solutions guide. Many of these documents may be found the EMC Powerlink site at: <http://Powerlink.EMC.com>).

For Microsoft SQL Server information, refer to the Microsoft websites, including <http://www.microsoft.com/sql/>, or more directly to the documentation page for the various versions at: <http://msdn.microsoft.com/sql/sqlref/docs/default.aspx>

Microsoft SQL Server Books On Line documentation provides extensive coverage of features and functions and may be installed through the SQL Server installation process, independently of the SQL Server database engine. Updated versions of the Books On Line documentation are available for free download from the Microsoft SQL Server website: <http://www.microsoft.com/sql>

SYMCLI

Solutions Enabler Release Notes (by release)

Solutions Enabler Support Matrix (by release)

Solutions Enabler Symmetrix Device Masking CLI Product Guide (by release)

Solutions Enabler Symmetrix Base Management CLI Product Guide (by release)

Solutions Enabler Symmetrix CLI Command Reference (by release)

Solutions Enabler Symmetrix Configuration Change CLI Product Guide (by release)

Solutions Enabler Symmetrix SRM CLI Product Guide (by release)

Solutions Enabler Symmetrix Double Checksum CLI Product Guide (by release)

Solutions Enabler Installation Guide (by release)

Solutions Enabler Symmetrix CLI Quick Reference (by release)

TimeFinder

Solutions Enabler Symmetrix TimeFinder Family CLI Product Guide (by release)

TimeFinder/Integration Modules Product Guide (by release)

SRDF

Solutions Enabler Symmetrix SRDF Family CLI Product Guide (by release)

Symmetrix Remote Data Facility (SRDF) Product Guide

Symmetrix Automated Replication UNIX and Windows

Replication Manager

Replication Manager Product Guide

Replication Manager Support Matrix

AutoStart

AutoStart Data Source for SRDF Administrator Guide

AutoStart Module for SQL Server 2000 Administrators Guide

Appendix B References

This appendix presents these topics:

B.1	Sample SYMCLI group creation commands.....	B-2
-----	--	-----

B.1 Sample SYMCLI group creation commands

The following shows how Symmetrix device groups and composite groups are created for the TimeFinder family of products including TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap.

This example shows you how to build and populate a device group and a composite group for TimeFinder/Mirror usage.

Device group:

1. To create the device group execute the command:

```
symmdg create dbgroup -type regular
```

2. The standard devices need to be added to the group. The database containers reside on five Symmetrix devices. The device numbers for these are 0CF, 0F9, 0FA, 0FB, and 101.

```
symld -g dbgroup add dev 0CF
```

```
symld -g dbgroup add dev 0F9
```

```
symld -g dbgroup add dev 0FA
```

```
symld -g dbgroup add dev 0FB
```

```
symld -g dbgroup add dev 101
```

3. Associate the BCV devices to the group. The number of BCV devices should be the same as the number of standard devices. They should also be the same size. The device serial numbers of the BCVs used in the example are 00C, 00D, 063, 064, and 065:

```
symbcv -g dbgroup associate dev 00C
```

```
symbcv -g dbgroup associate dev 00D
```

```
symbcv -g dbgroup associate dev 063
```

```
symbcv -g dbgroup associate dev 064
```

```
symbcv -g dbgroup associate dev 065
```

Composite group:

1. To create the composite group execute the command:

```
symcpg create dbgroup -type regular
```

2. The standard devices need to be added to the composite group. The database containers reside on five Symmetrix devices on two different Symmetrix arrays. The device numbers for these are 0CF, 0F9 on Symmetrix with the last three digits

of 123, and device numbers 0FA, 0FB, and 101 on the Symmetrix with the last three digits of 456.

3. Associate the BCV devices to the composite group. The number of BCV devices should be the same as the number of standard devices. They should also be the same size. The device serial numbers of the BCVs used in the example are 00C, 00D, 063, 064, and 065:

```
symbcv -cg dbggroup associate dev 00C -sid 123
```

```
symbcv -cg dbggroup associate dev 00D -sid 123
```

```
symbcv -cg dbggroup associate dev 063 -sid 456
```

```
symbcv -cg dbggroup associate dev 064 -sid 456
```

```
symbcv -cg dbggroup associate dev 065 -sid 456
```

This example shows how to build and populate a device group and a composite group for TimeFinder/Clone usage.

Device group:

1. To create the device group dbggroup execute the command:

```
symdg create dbggroup -type regular
```

2. The standard devices need to be added to the group. The database containers reside on five Symmetrix devices. The device numbers for these are 0CF, 0F9, 0FA, 0FB, and 101:

```
symld -g dbggroup add dev 0CF
```

```
symld -g dbggroup add dev 0F9
```

```
symld -g dbggroup add dev 0FA
```

```
symld -g dbggroup add dev 0FB
```

```
symld -g dbggroup add dev 101
```

3. The target clone devices need to be added to the group. The targets for the clones can be standard devices or BCV devices. In this example, BCV devices are being used. The number of BCV devices should be the same as the number of standard devices. They should also be the same size or larger than the paired standard device. The device serial numbers of the BCVs used in the example are 00C, 00D, 063, 064, and 065:

```
symbcv -g dbggroup associate dev 00C
```

```
symbcv -g dbggroup associate dev 00D
```

```
symbcv -g dbggroup associate dev 063
```

```
symbcv -g dbgroup associate dev 064
```

```
symbcv -g dbgroup associate dev 065
```

Composite Group:

1. To create the composite group dbgroup execute the command:

```
symcg create dbgroup -type regular
```

2. The standard devices need to be added to the group. The database containers reside on five Symmetrix devices on two different Symmetrix arrays. The device numbers for these are 0CF, 0F9 on Symmetrix with the last 3 digits of 123, and device numbers 0FA, 0FB, and 101 on the Symmetrix with the last 3 digits of 456:

```
symcg -g dbgroup add dev 0CF -sid 123
```

```
symcg -g dbgroup add dev 0F9 -sid 123
```

```
symcg -g dbgroup add dev 0FA -sid 456
```

```
symcg -g dbgroup add dev 0FB -sid 456
```

```
symcg -g dbgroup add dev 101 -sid 456
```

3. Add the target for the clones to the device group. In this example, BCV devices are added to the composite group to simplify the later symclone commands. The number of BCV devices should be the same as the number of standard devices. They should also be the same size. The device serial numbers of the BCVs used in the example are 00C, 00D, 063, 064, and 065.

```
symbcv -cg dbgroup associate dev 00C -sid 123
```

```
symbcv -cg dbgroup associate dev 00D -sid 123
```

```
symbcv -cg dbgroup associate dev 063 -sid 456
```

```
symbcv -cg dbgroup associate dev 064 -sid 456
```

```
symbcv -cg dbgroup associate dev 065 -sid 456
```

This example shows how to build and populate a device group and a composite group for TimeFinder/Snap usage:

Device group:

1. To create the device group dbgroup execute the command:

```
symdg create dbgroup -type regular
```

2. Add the standard devices to the group. The database containers reside on five Symmetrix devices. The device numbers for these are 0CF, 0F9, 0FA, 0FB, and 101:

```
symld -g dbggroup add dev 0CF
```

```
symld -g dbggroup add dev 0F9
```

```
symld -g dbggroup add dev 0FA
```

```
symld -g dbggroup add dev 0FB
```

```
symld -g dbggroup add dev 101
```

3. Then the virtual devices or VDEVs need to be added to the group. The number of VDEVs should be the same as the number of standard devices. They should also be the same size. The device serial numbers of the VDEVs used in the example are 291, 292, 394, 395, and 396:

```
symld -g dbggroup add dev 291 -vdev
```

```
symld -g dbggroup add dev 292 -vdev
```

```
symld -g dbggroup add dev 394 -vdev
```

```
symld -g dbggroup add dev 395 -vdev
```

```
symld -g dbggroup add dev 396 -vdev
```

Composite group:

1. To create the composite group dbggroup execute the command:

```
symcg create dbggroup -type regular
```

2. The standard devices need to be added to the composite group. The database containers reside on five Symmetrix devices on two different Symmetrix arrays. The device numbers for these are 0CF, 0F9 on Symmetrix with the last 3 digits of 123, and device numbers 0FA, 0FB and 101 on the Symmetrix with the last 3 digits of 456:

```
symcg -g dbggroup add dev 0CF -sid 123
```

```
symcg -g dbggroup add dev 0F9 -sid 123
```

```
symcg -g dbggroup add dev 0FA -sid 456
```

```
symcg -g dbggroup add dev 0FB -sid 456
```

```
symcg -g dbggroup add dev 101 -sid 456
```

3. Then the virtual devices or VDEVs need to be added to the composite group. The number of VDEVs should be the same as the number of standard devices. They

should also be the same size. The device serial numbers of the VDEVs used in the example are 291, 292, 394, 395, and 396:

```
symld -cg dbggroup add dev 291 -sid 123 -vdev
```

```
symld -cg dbggroup add dev 292 -sid 123 -vdev
```

```
symld -cg dbggroup add dev 394 -sid 456 -vdev
```

```
symld -cg dbggroup add dev 395 -sid 456 -vdev
```

```
symld -cg dbggroup add dev 396 -sid 456 -vdev
```