Planning for the Unplanned: DB2 9 Disaster Recovery with a NetApp Storage System

SnapMirror Async

Jawahar Lal, Bobby Oommen NetApp Inc.

February 2008 | TR-3654

TABLE OF CONTENTS

1. DISASTER RECOVERY: EXECUTIVE SUMMARY 3

2. INTRODUCTION 3

3. PURPOSE AND SCOPE 4

4. UNDERSTANDING DISASTER RECOVERY LANGUAGE 4

4.1. Business Continuity Plan (BCP) 4
4.2. Disaster Recovery Plan (DRP) 4
4.3. Recovery Time Objective (RTO) 5
4.4. Recovery Point Objective (RPO) 5
4.5. Disaster Tolerance 5
4.6. High Availability (HA) 5
4.7. Archive Logging 5
4.8. Media Recovery 5
4.9. Instance or Crash Recovery 6
4.10. Database-Coordinated Snapshot Copy 6

5. SNAPMIRROR: A QUICK OVERVIEW 6

6. REQUIREMENTS AND ASSUMPTIONS 7

1) General Assumptions 7
2) Environment Assumptions 7
3) Security and Access Issues 8
4) Network Connectivity 8
5) Mount the NetApp storage system's root volume to the database server (Optional) 9
6) Enable rsh access from the database server (Optional) 9
7) Required permissions on the volumes to be used for the database 9
8) Mount and change ownership of the file system on the mountpoint 9

7. NETWORK AND STORAGE INFRASTRUCTURE 10

8. CONFIGURATION DETAILS 11

9. DATABASE SETUP 11

10. CONFIGURE DATABASE REPLICATION USING SNAPMIRROR 11

10.1. Level 0 or baseline Replication 12
10.2. Level 1 replication 13
10.3. Level 2 replication 13

11. DISASTER 14

12. RECOVERY OF THE DATABASE 15

12.1. Breaking the SnapMirror Relationship 15
12.2. Restore the database system and data volumes 15
12.3. Restoring the database logs (online and archive). 15
12.4. Mount the volumes on standby 16
12.5. Initialize the database as mirror 16
12.6. Roll forward the database 16

13. CONCLUSION 16

APPENDIX: SCRIPTS 18

db2inidb.conf 18

1. DISASTER RECOVERY: EXECUTIVE SUMMARY

As the need for uninterrupted availability of enterprise data is growing, more businesses are striving for 24x7 system availability and can't afford any downtime. In this era of continuous information availability, the complete and rapid recovery from a disaster is not just nice to have; it is a necessity.

Whether you're a business, a government agency, a healthcare organization, or an educational institution, you must ensure that you're prepared for when, not if, disaster strikes. A protracted interruption in your organization's ability to access data will disrupt business operations. This can lead to the loss of customers and revenue, a drop in share price, or possible noncompliance fines for failing to protect and/or provide information promptly when required.

A well-planned, rehearsed, and tested disaster recovery solution can save time and money for your organization by offering small recovery windows with no or acceptable data loss. NetApp offers an array of proven, low-cost, and simplified data protection and disaster recovery solutions for your organization's data.

2. INTRODUCTION

As recent world events have proven, disaster can strike anywhere and anytime. The question you need to answer is how well prepared your organization is to recover in the event of a disaster. Though most business houses, even the small ones, are meticulous when it comes to thinking ahead and devising future strategies to take the company forward, a common mistake is ignoring the possibility of a disaster crippling the organization. Critical data loss could well be the fatal prescription for an organization that is otherwise doing well.

Survey after survey shows they're in the majority: Over 50% of companies make no effort whatsoever to prevent avoidable disasters or to put into place strategies for recovering from outage events that can't be avoided. Of those companies that do plan, fewer than 50% actually test the strategy they develop, which is like having no strategy at all.

From time to time different groups, including professional accounting organizations, universities, and the U.S. government, study the results of disasters and downtime on business and employment. The results are never good. Among the most-published results are:

- 93% of companies that suffer a significant data loss are out of business within five years.[1]
- 43% of U.S. businesses never reopen after a disaster, and 29% more close within two years.[2]
- 30% of computer users say they spend the equivalent of one week per year reconstructing lost data.[3]

All these statistics suggest that organizations must plan for disaster recovery up front. The disaster recovery plan is a corporate survival kit in the eventuality of disaster. The continued operations of an organization depend on management's awareness of potential disasters, the ability to develop a disaster recovery plan to minimize disruptions of critical functions, and the capability to recover and restore vital business operations successfully and quickly.

A disaster recovery plan is a comprehensive statement of consistent actions to be taken before, during, and after a disaster. The plan should be documented and tested to ensure the continuity of operations and availability of critical resources in the event of a disaster. The primary objectives

[1] U.S. Bureau of Labor.

[2] The Hartford & U.S. Small Business Administration, 2002, p. 14.

[3] 3M Corporation.

of the plan are to protect the organization in the event that all or part of its IT services are rendered unusable, minimize the disruption to critical business operations, ensure some level of organizational stability, and provide an orderly recovery after a disaster.

Incidents like 9/11 and Hurricane Katrina give businesses a chance to see their disaster recovery plan in action. While some companies pass with flying colors, the plans of others are exposed as incomplete, unrealistic, and technologically flawed. Those companies with untested or poorly tested plans will eventually discover that they aren't as protected as they thought they were.

There are several approaches to protect data and maintain data availability in the face of hardware, software, or even site failures. Backups provide a way to recover lost data from an archival medium (tape or disk). Redundant hardware technologies also mitigate the damage caused by hardware failures. Data replication or mirroring provides a third mechanism to ensure data availability and minimize downtime. The NetApp SnapMirror® product provides a fast and flexible enterprise solution for data replication over local area, wide area, and Fibre Channel (FC) networks. If a disaster strikes at a source site, the replicated mission-critical data can immediately be made available at a remote site, ensuring uninterrupted operation and data availability. For detailed information on popular disaster recovery solutions offered by NetApp, please visit the data protection solutions page on the NetApp Web site (*www.netapp.com/solutions/data_protection.html*).

3. PURPOSE AND SCOPE

The scope of this document is limited to disaster recovery of a DB2 9 single-instance database. The disaster recovery solutions covered in this document use NetApp SnapMirror technology in “Async” mode.

Using SnapMirror, it is now possible to recover from a disaster at a remote physical location. If critical data is mirrored to a different physical location, a serious disaster no longer means prolonged data unavailability. The mirrored data can be made available to clients across the network until the damage caused by the disaster is repaired. Recovery may include recovery from corruption, natural disaster at the source site, accidental deletion, sabotage, and so on. SnapMirror also allows application server layer information to be replicated to the disaster recovery site. In the event of disaster, once the disaster recovery site is operational, all applications can be switched over to the servers at the disaster recovery site, and all application traffic can be directed to these servers for as long as necessary to recover the source site. Once the source site is recovered, SnapMirror can be used to transfer the data efficiently back to the primary site. After the production site takes over normal application operation again, SnapMirror transfers to the disaster recovery site can resume without requiring a second complete data transfer.

4. UNDERSTANDING DISASTER RECOVERY LANGUAGE

4.1. Business Continuity Plan (BCP)

Business continuity plan (BCP) describes processes and procedures an organization puts in place to ensure that essential business processes continue during and after a disaster. Normally it takes into account the protection of the whole organization, including buildings, IT infrastructure, employees, and all other resources. The main objective of a BCP is to prevent interruption of mission-critical services and to reestablish full functions as swiftly and smoothly as possible.

4.2. Disaster Recovery Plan (DRP)

Disaster recovery plan is a subset of the business continuity plan and focuses solely on the protection and recovery of the mission-critical data and the IT infrastructure. It details step-by-step procedures and processes for the IT staff to follow during and after the disaster. It

describes the recovery priority and objectives (RPO and RTO) for each application.

4.3. Recovery Time Objective (RTO)

The recovery time objective (RTO) indicates the time spent in bringing the application up and resuming the operation after the disaster. It is also known as acceptable downtime after the disaster. The unit of measure for RTO is time, with values ranging from seconds to days or weeks. Lower the application's RTO value, the greater the organization's dependence on that particular application, and consequently the higher the recovery priority after the disaster.

4.4. Recovery Point Objective (RPO)

The recovery point objective (RPO) is the age of files that must be recovered from a last successful backup for resuming the normal operations after a system crash, hardware failure, data corruption, or communications failure. The RPO is expressed backward in time from the instant at which the failure occurred till the last successful recovery point and can be specified in seconds, minutes, hours, or days. RPO is an important consideration in disaster recovery planning.

4.5. Disaster Tolerance

Greater awareness of the need for disaster recovery is prompting application architects to build disaster readiness into the business systems they design. Disaster tolerance is a term used to signify a system with some ability to withstand major disruption. Several technologies are used to provide disaster tolerance, including hardware redundancy, data replication, server clustering, and remote data centers.

4.6. High Availability (HA)

High availability is an architecture that maximizes the data availability. It is a subcategory of disaster recovery. The ultimate disaster-tolerant system is classed as a high-availability (HA) system. The HA systems are designed to eliminate application downtime by using redundant hardware and networking components and specialized application and operating system software. HA systems can seamlessly route around failures in the computing infrastructure without affecting end-user access to data.

The resilience of this system is often measured in terminology borrowed from the telecommunications industry. For example, a configuration that offers 99.999% availability (also known as five nines) can have only five minutes of planned or unplanned downtime in any given year.

4.7. Archive Logging

Archive logging is a database feature that enables retention of the transaction logs. The retained transaction logs are called archive logs. Using archive and active logs, a database roll-forward recovery is possible to any point in time before the failure occurred, rather than only to the point in time of a full backup. The archived logs can be moved offline and still be used for roll-forward recovery.

4.8. Media Recovery

Media recovery is a user-initiated data recovery. It can be used to recover damaged data files, index files, or transaction logs. The recovery is performed by restoring from the last successful backup followed by roll forward of transaction logs.

4.9. Instance or Crash Recovery

Instance or crash recovery process is a special form of recovery, which happens when a database instance is started for the first time after a crash. The crash recovery uses only online transaction logs, and the goal is to bring the database to a transaction-consistent state, preserving all committed changes up to the point when the instance failed.

4.10. Database-Coordinated Snapshot Copy

A database coordinated Snapshot™ copy is a Snapshot created after bringing the database in write suspend state. It guarantees the database consistency, which means a database recovery is guaranteed from a database coordinated Snapshot copy.

5. SNAPMIRROR: A QUICK OVERVIEW

In order to protect your organization's data from disaster and ensure quick and smooth recovery, your data needs to be replicated to one or more other physical locations. NetApp SnapMirror technology allows data replication between two NetApp storage systems. The NetApp storage system from which data is transferred is referred to as the SnapMirror source, and the NetApp storage system to which the data is transferred is referred to as the SnapMirror destination. The SnapMirror source and destination can be miles apart, provided that both NetApp storage systems can communicate with each other across a network.

NetApp SnapMirror technology supports volume SnapMirror as well as qtree SnapMirror. SnapMirror source volumes and qtrees are writable data objects, but the SnapMirror destination volumes and qtrees are read-only, usually on a separate storage system. In the case of a disaster where the source volumes or qtrees go down, the replicated data at the destination volumes or qtrees can be made available by making the volumes or qtrees writable. The SnapMirror configuration details are maintained in a configuration file called `snapmirror.conf`, which resides on the destination NetApp storage system. This file, along with information entered using the `snapmirror.access` option or the `snapmirror.allow` file, is used to establish relationships between specified source volumes or qtrees and the destination volume or qtree where the mirrored data is kept.

NetApp Data ONTAP® supports SnapMirror Async as well as SnapMirror Sync. In the SnapMirror Async mode, SnapMirror performs incremental, block-based replication as per the frequency defined in `snapmirror.conf` file. Write requests to the SnapMirror source are acknowledged as soon as they are written to its NVRAM and are not delayed until the SnapMirror destination has received and/or processed the request. Performance impact on the SnapMirror source storage system is minimal, as long as the system is configured with sufficient CPU and disk I/O resources. The first and most important step in Async mode involves the creation of a one-time baseline transfer of the entire data set from the SnapMirror source system to the SnapMirror destination system. This is required before incremental updates can be performed. After the baseline transfer is complete, scheduled or manually triggered updates can occur. Each update transfers only the new and changed blocks from the source to the destination storage system. Because asynchronous replication is periodic, SnapMirror is able to consolidate writes and conserve network bandwidth, thereby minimizing the impact on write throughput and write latency.

NetApp Data ONTAP supports SnapMirror Sync to meet the very high availability requirements for certain environments where all data changes written to a production site must be replicated to a remote site synchronously. SnapMirror in Synchronous or Sync mode is a mode of replication that sends updates from the source volumes or qtree to the destination volumes or qtree as they occur, rather than according to a predetermined schedule. This guarantees that data written on the source system is protected on the destination even if the entire source system goes down. With

Synchronous mode, each time a transaction attempts to write data to disk, the data is sent to both the source and destination storage systems in parallel. It is not until both NetApp storage systems have committed the data associated with the write operation to NVRAM that the system acknowledges that the transaction is complete. In other words, the application that initiated the write operation must wait until it receives the acknowledgement from both the source and destination storage systems before it can continue.

SnapMirror Semi-Sync mode is a variation of SnapMirror Sync mode, and it provides a middle ground that keeps the source and destination file systems more closely synchronized than the Async mode, but with less impact on application performance. Configuration of Semi-Sync mode is nearly identical to the configuration of Sync mode, with the exception being the addition of an option that specifies how many writes, seconds, or ops can be outstanding (unacknowledged by the destination system) before the source system delays acknowledging write operations from clients. Internally, Semi-Sync mode works identically to Sync mode in most cases. The only difference lies in how quickly client writes are acknowledged; the replication methods used are the same.

Note: The Sync and Semi-Sync modes are supported only with volume SnapMirror.

For more details on SnapMirror deployment and implementation, please refer to the technical reports "SnapMirror Deployment and Implementation Guide" and "Synchronous SnapMirror Design and Implementation Guide" on NOW™ (NetApp on the Web) at *http://now.netapp.com*.

6. REQUIREMENTS AND ASSUMPTIONS

1) General Assumptions

In order to take maximum advantage of the procedures described in this document, it is assumed that readers of this document are familiar with the following:

- Commands and operations of Data ONTAP and NetApp storage system
- Administration and operation of DB2 9 server instance and database
- Linux® system administration commands
- NetApp SnapMirror technology
- Disaster recovery concept

The NetApp storage systems used to produce this document are loaded with Data ONTAP 7G and are licensed for NFS, FCP, iSCSI, and SnapMirror products.

We used Linux RHEL 5 server as the database server. It is assumed that the Linux hosts used for the production database have DB2 9 server software installed and a single database is used per instance.

In order to produce this document we used NFS protocol. If you are using a SAN environment, then make sure that database hosts have the following products installed and configured:

- Appropriate HBA
- SanSurfer Utility (installed if HBA used is from Qlogic)
- NetApp Host Attach/Support Kit

Please check the Compatibility and Configuration Guide for NetApp FCP and iSCSI Products to find out about supported HBAs.

2) Environment Assumptions

This document covers disaster recovery solutions offered by NetApp for DB2 9 server using SnapMirror technology. The scripts and process steps contained in this document may require significant modifications to run under your version of UNIX®. The sample scripts in this document assume the following:

- The NetApp storage system used for the SnapMirror source is 'ntapsrc'.
- The NetApp storage system used as the SnapMirror destination is 'ntapdst'.
- The aggregate on the NetApp storage systems used for database storage is 'dbaggr'.
- The flexible volume used to store database binaries is 'dbsys'.
- The flexible volume used to store database data is 'dbdata'.
- The flexible volume used to store database transaction logs is 'dblogs'.
- The flexible volume used to store database archive logs is 'dbarchlogs'.
- The flexible volumes dbdata, dblogs, and dbarchlogs reside in aggregate dbaggr.
- At the database host, the mountpoints used are /mnt/dbsys, /mnt/dbdata, /mnt/dblogs, and /mnt/dbarchlogs.
- DB2 Home resides on a UNIX database host.

We also assumed that the database host used for accessing the database at the disaster recovery site has a similar setup to the database host on the primary site and has all required privileges to access the volumes on the SnapMirror destination NetApp storage system.

3) Security and Access Issues

You need to make sure that the FlexVol® volumes to be used for the database's binaries, data, transaction logs, and archive logs have their security style set appropriately. If the database host used is a UNIX host, then the security style must be set to 'UNIX'. The security style can be changed by executing the following command on the NetApp storage system:

qtree security <volume name> unix

For example, to change the security style for a volume named dbdata, you would execute the following command on the NetApp storage system:

```
qtree security /vol/dbdata unix
```

Note: Parameters shown in angle brackets (< >) are required; parameters or options shown in square brackets ([ ]) are optional; options shown in curly brackets ({}) are mutually exclusive, and one value must be selected; a comma followed by ellipses (…) indicates that the preceding parameter can be repeated multiple times.

4) Network Connectivity

You also need to make sure that the database hosts and NetApp storage systems can communicate with each other through the network. This is done by making appropriate entries to the /etc/hosts files on the NetApp storage system as well as on the database host systems.

- Add the following line to the database host's /etc/hosts file if it doesn't already exist:

  <NetApp storage system IP> <NetApp storage system name>

  For example, to add an entry to the /etc/hosts file on the database server for the NetApp storage system named ntapsrc that has IP address 10.32.70.134, you would add the following line:

  10.32.70.134 ntapsrc

- Add the following line to the /etc/hosts file on a NetApp storage system if it doesn't already exist:

  <database server IP> <database server name>

For example, to add an entry to the /etc/hosts file on the NetApp storage system for a database server named dbhost1 that has IP address 172.32.70.43, you would add the following line:

172.32.70.43 dbhost1

5) Mount the NetApp storage system's root volume to the database server (Optional)

In order to mount the NetApp storage system's root volume to the database host and make necessary changes to some configuration files, the user root on the database sever must have access to the root volume /vol/vol0. For example, to grant access on a root volume named /vol/vol0 on a NetApp storage system to the user root on the database host system (dbhost1), you would execute the following command on the storage systems:

```
exportfs –p rw=dbhost1,root=dbhost1,anon=0 /vol/vol0
exportfs –a
```

6) Enable rsh access from the database server (Optional)

If you intend to use 'rsh' commands from your database host, then add the IP address and the user name of the database host to the /etc/hosts.equiv file on the NetApp storage system. The entry should look similar to the following:

<IP> <username>

For example to add the hostname 'hostsrc' and user 'db2inst1' to the hosts.equiv file your entry should look similar to the following:

10.61.162.170 db2inst1

7) Required permissions on the volumes to be used for the database

Before you create a database on the FlexVol volumes on the NetApp storage system, you need to mount them to a database host system. In order to mount FlexVol volumes, the user root on the database host must have access to them. Execute the following commands on the NetApp storage systems to grant access on FlexVol volumes:

exportfs -p rw=<db host name>,root=<db host name>,anon=0 <volume name>

For example, to grant access on the FlexVol volume named dbdata to the user root on the database server named dbhost1, you would execute the following command:

```
exportfs -p rw=dbserver,root=dbserver,anon=0 /vol/dbdata
exportfs –a
```

Grant permission on all FlexVol volumes to be used for the database using the above command.

8) Mount and change ownership of the file system on the mountpoint

The FlexVol volumes to be used for the database need to be mounted on the database host system by creating an entry in the /etc/fstab file as

<filer name>:/vol/<volname> <mount point> rw,nointr,rsize=32768, wsize=32768,bg,vers=<nfs version>,sec=sys,tcp 0 0

mount <mount point>

For example to create a mountpoint for the volume dbdata on source storage with mountpoint as /db2inst1/dbdata on database server you would add the following to the /etc/fstab and mount it:

```
ntapsrc:/vol/dbdata /db2inst1/dbdata  rw,nointr,rsize=32768,
wsize=32768,bg,vers=nfs3,sec=sys,tcp 0 0
mount /db2inst1/dbdata
```

To install DB2 and create a database successfully, you need to make sure that the file system on the mountpoints is owned by the instance owner on the database server. Change the ownership of the mounted volumes to the user `db2inst1` by executing the following command at the database server:

`chown –R db2inst1:db2iadm1 <mount point>`

For example, to change the ownership of a file system on the mountpoint named `/db2inst1/dbdata`, you would execute the following command on the database server:

```
chown –R db2inst1:db2iadm1 /db2inst1/dbdata
```

NOTE: The above mount command example is for a Linux host. For other operating systems, please refer to the operating system manual or "Integrating DB2 9 with NetApp Storage" technical paper.[4]

7. NETWORK AND STORAGE INFRASTRUCTURE

Figure 1 illustrates a high-level network architecture diagram for the database disaster recovery scenario using NetApp storage systems and SnapMirror. The same architecture was used to produce this technical document.

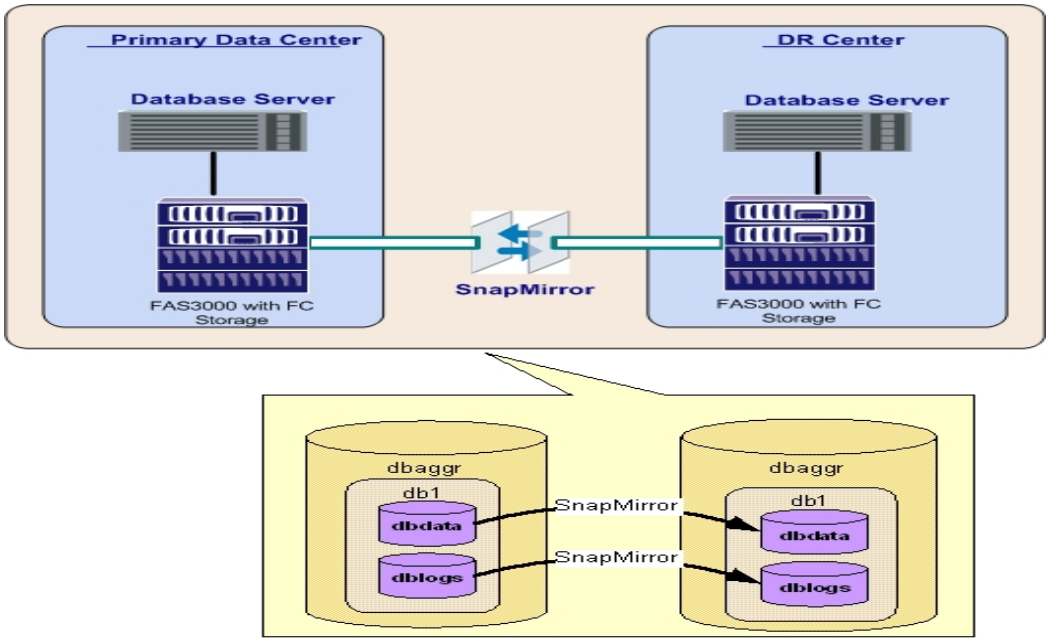

Figure 1) High-level DR scenario infrastructure.

In this architecture, the DB2 Home resides on the database server. The database directory, active logs, and archive logfiles for the database reside on the FlexVol volumes on a NetApp storage system. The volumes used for the database on the NetApp storage system at the primary data center are replicated to a secondary location using SnapMirror. The secondary location is referred

[4] *www.netapp.com/library/tr/3531.pdf.*

to as the disaster recovery site throughout this document.

8. CONFIGURATION DETAILS

In order to produce this document we used DB2 9 server. The database host was running RHEL 5 with kernel 2.6 patch 18. For database storage, we used a NetApp FAS3070 series (FAS3000). The environment details are:

- Database: DB2 9 server, DB2 instance (db2inst1)
- Database archive log: Enabled
- DB2 Home: Local on database Host (`/home/db2inst1`)
- NetApp storage system volumes used for the database data, transaction logs, and archive logs are `dbdata` , `dblogs`, and `dbarchlogs` respectively

Database layout:

Database directory:

`/mnt/dbsys`

Database data:

`/mnt/dbdata/`

Active logs:

`/mnt/dblogs/`

Archive logs:

`/mnt/dbarchlogs/`

9. DATABASE SETUP

As described in section 6, it is assumed that you already have DB2 9 server installed and a database instance is created, and that the database has its database directory, data, active logs, and transaction logs on NetApp storage system volumes. It is also assumed that the database has archive logging enabled.

10. CONFIGURE DATABASE REPLICATION USING SNAPMIRROR

As stated earlier, NetApp SnapMirror technology offers Sync, Semi-Sync, and Async mirroring options. For this paper we used Async mode of the SnapMirror technology. Before you start configuring SnapMirror, you need to complete the following three basic steps:

a). Identify the SnapMirror source and the destination NetApp storage systems. The source NetApp storage system should already have a DB2 9 database created on it. In our case, we used a NetApp storage system named `ntapsrc,` which has DB2 9 database create on it, as the SnapMirror source and second NetApp storage system named `ntapdst` that is on the disaster recovery site as the SnapMirror destination.

b). You need to identify the names of the volumes on the source NetApp storage system that need to be replicated to destination NetApp storage systems using SnapMirror. The volume names on the SnapMirror destination storage system could be different then the volume names on the SnapMirror source system.

c). The SnapMirror feature needs to be enabled on both the source and on the destination NetApp storage system by executing the following command:

```
options snapmirror on
```

After completing the above described three steps, you need to plan about your Snapshot backup strategy. Your backup strategy may vary based on the recovery time objective (RTO) and recovery purpose objective (RPO) for each database. The RTO helps you to define the timing for full database Snapshot and RPO for incremental backup. In our example scenario, we have recovery time object of four hours and recovery purpose object of 15 minutes. Therefore, we plan to create a Snapshot backup of the database every four hours and incremental backup every 15 minutes.

When you are using SnapMirror to replicate your DB2 database, you can define your database replication in terms of the following three levels:

- Level 0 or baseline replication
- Level 1 replication (full database backup)
- Level 2 replication (incremental backup)

10.1. Level 0 or baseline Replication

Establish a SnapMirror relationship starts at level 0 replication between two volumes. This takes place only once and transfers all the data blocks from the SnapMirror source volume to the SnapMirror target volume. After level 0 replication, subsequent SnapMirror update operations replicate only changed data blocks from source to destination. The SnapMirror relationship can be initialized by executing the following command from the destination storage system:

snapmirror initialize –S <Source storage system>:<Source volume Name> <Destination storage system>: <Destination Volume Name>

For example, to initialize a SnapMirror relationship between a volume named dbdata on a source storage system named ntapsrc and a volume named dbdata on the destination storage system named ntapdst you need to execute the following command from the destination storage system:

```
snapmirror initialize –S ntapsrc:dbdata ntapdst:dbdata
```

We recommend you create a database-consistent Snapshot copy before you initialize a SnapMirror relationship. In order to create a consistent Snapshot copy for a DB2 database you need to complete the following four steps:

a). Connect to the database.

b). Suspend the database write temporarily by executing the following command from the database server:

```
db2 set write suspend for database
sync
```

c). Create a Snapshot copy for the volumes that are used for the database by executing the following command:

snap create <Volume Name> <Snapshot Name>

For example, to create a Snapshot copy named level0-snap-dbdata for a volume named dbdata you need to execute the following command on the storage system:

```
snap create dbdata level0-snap-dbdata
```

Alternatively, you can rsh to create a Snapshot copy from the database server as shown in the following example:

```
rsh ntapsrc snap create dbdata level0-snap-dbdata
```

Create Snapshot copies for all the volumes that are used for the database.

d). Resume the database writes by executing the following command.

```
db2 set write resume for  database
```

Now from the destination storage system initialize the SnapMirror relationship for all the volumes that are used for the database.

The time taken by the level 0 replication depends on the size of the database and network bandwidth and is done only once. You can check periodically on the status of the SnapMirror operation by executing the following:

```
snapmirror status
```

10.2. Level 1 replication

After you initialize SnapMirror, your production database at the primary data center is available for business use, and the SnapMirror process continues to replicate data in the background. After the baseline transfer is completed, you need to run periodic SnapMirror updates to keep your DR center up to date to meet your RPO and RTO. The subsequent SnapMirror updates can be scheduled using the /etc/snapmirror.conf file on the destination storage system or by executing the SnapMirror update command manually.

snapmirror update –S <Source storage system>:<Source volume Name> <Destination storage system>:<Destination Volume Name>

For example, to execute a manual SnapMirror update for a volume named dbdata, you need to execute the following command from the SnapMirror destination storage system:

```
snapmirror update –S ntapsrc:dbdata ntapdst:dbdata
```

You must create database-consistent Snapshot copies for the volumes that are used for the database before executing the SnapMirror update command. A database-consistent Snapshot copy can be created by completing the fours steps described in section 10.1.

Level 1 is considered a full database backup; therefore, you are not required to replicate the temp tablespace data and archive logs.

10.3. Level 2 replication

After the level 1, you need to periodically replicate you transaction log and archive logs to the DR site. The replication frequency is dictated by RPO. In our example scenario we have a 15-minute RPO; therefore our level 2 replication is every 15 minutes. In order to perform recovery as per our RPO you need both transaction log and archive logs. For level 2 you need to create database-consistent Snapshot copies for transaction log and archive log volumes. After creating consistent Snapshot copies, you need to execute the SnapMirror update command as shown in the following example:

```
snapmirror update –S ntapsrc:dblogs ntapdst:dblogs
snapmirror update –S ntapsrc:dbarchlogs ntapdst:dbarchlogs
```

For our example scenario, the replication can be summarized as shown in Table 1.

TABLE 1				
Tablespace Name	**Level 0 (One time)**	**Level 1 (4 Hrs)**	**Level 2 (15 mins)**	**DB2 write suspend required**
Dbdata	Yes	Yes	No	Yes
Dblogs	Yes	Yes	Yes	Yes
Dbarchlogs	Yes	No	Yes	No
Dbtemp	Yes	No	No	Yes
Dbsys	Yes	Yes	No	No

11. DISASTER

Let us assume that the disaster strikes at 1:40 p.m. At this point, the system state on the primary site will look somewhat similar to Figure 2.

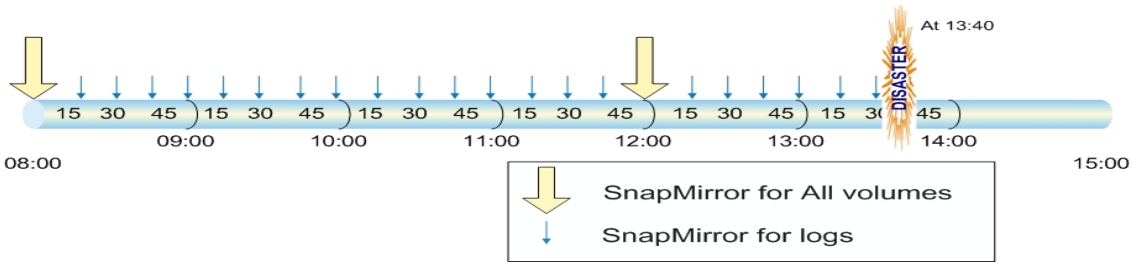

Figure 2) Database state at the primary site at the time disaster struck.

After a disaster the primary database becomes unavailable and the disaster recovery site will be in SnapMirror Snapshot consistent state up to the last successful SnapMirror point. That means the volumes at the disaster recovery site will have the same data as the corresponding source volume had at the time of successful SnapMirror update. For our example scenario, we have level 1 replication at 1200 hours and after that we have level 2 replication, which consists of transaction and archive logs every 15 minutes. In the given scenario, the database volumes at the DR site are in the following state at the time of disaster:

- Dbsys: consistent with primary database as of 1200 hours
- Dbdata: consistent with primary as of 1200 hours
- Dblogs: consistent with primary as of 1330 hours
- Dbarchlogs: consistent as of 1330 hours

In this case the DR site database can be recovered to the point the primary database was at 1330 hours.

The RPO is determined from the time that the SnapMirror copy of the transaction logs volume was created to the time the disaster started. The RTO will vary based on the transaction logs that need to be applied for the database recovery. For example, in our disaster recovery scenario, the disaster started at 1:40 p.m., and the database coordinated SnapMirror copy from which we want to restore the transaction logs volume was created at 1:30 p.m. Therefore, the RPO value will be 10 minutes. Figure 3) illustrates the RPO for example scenario.

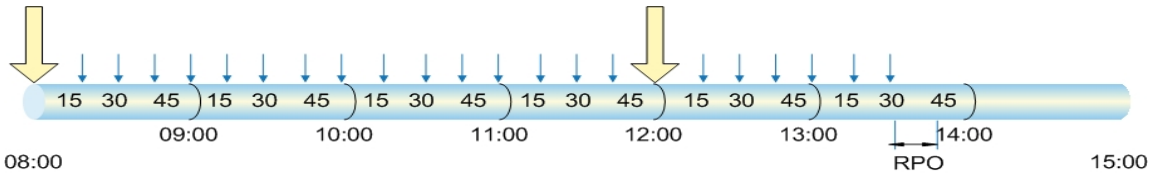

Figure 3) Recovery from a SnapMirror point.

12. RECOVERY OF THE DATABASE

In order to bring the database up at the DR site you need to perform the steps described in the following sub-sections.

12.1. Breaking the SnapMirror Relationship

Just after disaster the primary site becomes unavailable; so does the SnapMirror source volume. The destination volumes are in SnapMirror status not available for writes. In order to make destination volumes available for database read/writes, you need to break the SnapMirror relationship of all the volumes that are used by the database. You can break the SnapMirror relationship by executing the following command from the destination NetApp storage system:

`snapmirror break <volume name>`

For example, to break a SnapMirror relationship and make a volume named `dbdata` read/writable, you would execute the following command on the NetApp storage system:

```
snapmirror break dbdata
```

This command changes the destination volume's status from '**snapmirrored'** to '**broken-off'**, thus making it ready for the database read/writes.

12.2. Restore the database system and data volumes

The volumes at the standby database are available once the SnapMirror relationship is broken off, but the database is not consistent, and it is very important that during recovery we should have a consistent database after a disaster. A volume can be brought into a database-consistent state by restoring it from the database-coordinated Snapshot copies. The following commands can be used to restore the volume on a standby storage system:

`snap restore –f –s <snapshot name> <volume name>`

For example, to restore a volume named `dbdata` with the level 1 Snapshot copy named level1-snap-dbdata, you would execute the following command on the NetApp storage system:

```
snap restore -f -s level1-snap-dbdata dbdata
```

The level1-snap-dbdata is the last database-consistent Snapshot copy for data before the disaster.

12.3. Restoring the database logs (online and archive).

After restoring the database system and data volumes from the level 1 Snapshot copy, you need to restore the transaction logs and archive logs volumes from the last database-coordinated Snapshot copy. The following commands can be used to restore the volumes on the standby storage system:

snap restore –f –s <snap name> <volume name>

For example, to restore a database log volume named `dblogs` with the level 2 Snapshot copy named level2-snap-dblogs, you would execute the following command on the NetApp storage system:

```
snap restore -f -s level2-snap-dblogs dblogs
```

The level2-snap-dblogs is the database-consistent Snapshot copy for the log volume before the disaster.

12.4. Mount the volumes on standby

At the disaster recovery site, you need to mount the volumes that are used for the database to a database server by adding following line to the /etc/fstab file and mount it:

<filer name>:/vol/<volname> <mount point> rw,nointr,rsize=32768, wsize=32768,bg,vers=<nfs version>,sec=sys,tcp 0 0

mount <mount point>

For example to create a mountpoint for the volume dbdata on destination storage with mountpoint as /db2inst1/dbdata on the database server, you would add the following to the /etc/fstab and mount it:

```
ntapdst:/vol/dbdata /db2inst1/dbdata  rw,nointr,rsize=32768,
wsize=32768,bg,vers=nfs3,sec=sys,tcp 0 0
mount /db2inst1/dbdata
```

12.5. Initialize the database as mirror

Make sure the DB2 instance owner at the standby server has sufficient access to write to the volumes. Start the database instance and relocate the database by executing the following commands on the database server as instance owner:

db2start

db2inidb <dbname> as mirror relocate using <configuration file>

For example, to initialize the primary database prod as a mirror with configuration file db2inidb.conf you would execute:

```
db2start
db2inidb prod as mirror relocate using db2inidb.conf
```

The format of db2inidb.conf can be found in the appendix.

12.6. Roll forward the database

The db2inidb will initialize the mirror database and put it in roll-forward pending state. In order to apply the logs restored in step 12.3 to the database, you need to roll forward the database using the online and archive logs if applicable. The roll-forward command is as follows:

db2 rollforward db <db name> to end of logs and complete

For example,

```
db2 rollforward db prod to end of logs and complete
```

After completing the preceding steps, the database recovery from the SnapMirror point of the database volumes will be complete.

13. CONCLUSION

NetApp SnapMirror is a proven data replication technology that offers great ROI. It provides flexible and robust methods to keep recovery objectives. The SnapMirror technology can be easily integrated with other NetApp technologies such as cluster, MetroCluster, and SyncMirror® to further improve recovery objectives and ensure high availability.

APPENDIX: SCRIPTS

db2inidb.conf

```
DB_NAME=prod,prod
DB_PATH=/db2inst1/dbdata, /db2inst1/dbdata
INSTANCE=db2inst1,db2inst1
NODENUM=0
LOG_DIR=/db2inst1/dbogs/NODE0000,/db2inst1/dblogs/NODE0000
CONT_PATH=/db2inst1/dbdata/*,/db2inst1/dbdata/*
```