Microsoft TechNet

Database Mirroring in SQL Server 2005

Published: April 1, 2005
By Ron Talmage, Solid Quality Learning

On This Page

Download

DatabaseMirroring.doc

517 KB
Microsoft Word file

Get Office File Viewers

Important:

Microsoft support policies only apply to the database mirroring feature as delivered with SQL Server 2005 Service Pack 1 (SP1) onwards. If you are not running SQL Server 2005 with SP1 or later, database mirroring should not be used in production environments. Microsoft support services will not support databases or applications that use database mirroring from the RTM release.

Top of page

Introduction

Database mirroring is a new SQL Server 2005 technology available for review for increasing database availability. Database mirroring transfers transaction log records directly from one server to another and can quickly fail over to the standby server. You can code client applications to automatically redirect their connection information, and in the event of a failover, automatically connect to the standby server and database. Fast failover with minimal data loss has traditionally involved higher hardware cost and greater software complexity. Database mirroring, however, can fail over quickly with no loss of committed data, does not require proprietary hardware, and is easy to set up and manage.

Top of page

Database Mirroring Overview

In database mirroring, an originating SQL Server 2005 instance continuously sends a database's transaction log records to a copy of the database on another standby SQL Server instance. The originating database and server have the role of *principal*, and the receiving database and server have the role of *mirror*. The principal and mirror servers must be separate instances of SQL Server 2005.

In all SQL Server databases, data changes are recorded in the transaction log before any changes to actual data pages are made. The transaction log records are placed first in a database's *log buffer* in memory, and then flushed to disk (or 'hardened') as quickly as possible. In database mirroring, as the principal server writes the principal database's log buffer to disk, it simultaneously sends that block of log records to the mirror instance.

When the mirror server receives a block of log records, it places the log records first into the mirror database's log buffer and then hardens them to disk as quickly as possible. Those transaction log records are later replayed on the mirror. Because the mirror database replays the principal's transaction log records, it duplicates the database changes on the principal database.

The principal and mirror servers are each considered a *partner* in the *database mirroring session*. A database mirroring session consists of a relationship between the partner servers when they mirror a database from one partner to another. A given partner server may have the principal role for one database and a mirror role for a different database.

In addition to the two partner servers (principal and mirror) a database mirroring session may have an optional third server, called the *witness*. The witness server's role is to enable automatic failover. When database mirroring is used for high availability, if a principal server suddenly fails, if the mirror server has confirmation from the witness, it can automatically take on the role of

principal and make its database available within a few seconds.

Some important items to note about database mirroring:

- The principal database must be in the FULL recovery model. Log records that result from bulk-logged operations cannot be sent to the mirror database.
- The mirror database must be initialized from a restore of the principal database with NORECOVERY, followed by restores in sequence of principal transaction log backups.
- The mirror database must have the same name as the principal database.
- Because the mirror database is in a recovering state, it cannot be accessed directly. You can create database snapshots on the mirror to indirectly read the mirror database at a point in time. (See 'Database Mirroring and Database Snapshots' later in this paper.)

Note: For more information about the terms related to database mirroring, see "Overview of Database Mirroring" in SQL Server 2005 Books Online.

Operating Modes

There are three possible *operating modes* for a database mirroring session. The exact mode is based on the setting of *transaction safety* and whether a witness server is part of the mirroring session.

Table 1: Database Mirroring Operating Modes

Operating Mode	Transaction safety	Transfer mechanism	Quorum required	Witness server	Failover Type
High Availability	FULL	Synchronous	Y	Y	Automatic or Manual
High Protection	FULL	Synchronous	Y	N	Manual only
High Performance	OFF	Asynchronous	N	N/A	Forced only

If safety is FULL and a witness is set, *synchronous* data transfer will occur, and a *quorum* is required for database service. A quorum vote requires at least two servers to decide which role, principal or mirror, each of the two partner servers should play.

In order to explore the three operating modes in more detail, let's first take a closer look at transaction safety and the role of a quorum.

Transaction Safety

If transaction safety (or just 'safety') is set to FULL, the principal and mirror servers operate in a synchronous transfer mode. As the principal server hardens its principal database log records to disk, it also sends them to the mirror. The principal then waits for a response from the mirror server. The mirror responds when it has hardened those same log records to the mirror's log disk. When safety is set OFF, the principal does not wait for acknowledgment from the mirror, and so the principal and mirror may not be fully synchronized (that is, the mirror may not quite keep up with the principal).

Synchronous transfer guarantees that all transactions in the mirror database's transaction log will be synchronized with the principal database's transaction log, and so the transactions are considered safely transferred. You set safety to FULL using

```
ALTER DATABASE [<dbname>] SET SAFETY FULL;
```

When safety is set to OFF, the communication between the principal and the mirror is asynchronous. The principal server will not wait for an acknowledgment from the mirror that the mirror has hardened a block of transaction records. The mirror will attempt to keep up with the principal, by recording transactions as quickly as possible, but some transactions may be lost if the principal suddenly fails and you force the mirror into service. (See the topic 'Forced Service' in SQL Server Books Online.)

The Quorum and the Witness Server

When the witness server is set, a database mirroring session requires a quorum to keep a database in service. A quorum is the minimal relationship among all the connected servers required by a synchronous database mirroring session. Because at least two servers are required for a quorum, when the witness is set the principal server must form a quorum with at least one other server to keep the database in service, regardless of the safety setting. Normally, if a witness is set, then the safety level is set to FULL as well.

The witness server assists the principal or mirror in forming a quorum. If a witness server is present, a loss of either the principal database or the mirror database leaves two servers to form a quorum. If the principal cannot see the mirror server, but it can form a quorum with the witness, it can keep its database in service. Similarly, if the mirror and witness servers cannot see the principal

server, and the mirror server can form a quorum with the witness, the mirror can take on the role of a new principal server.

The witness is not considered a single point of failure in a database mirroring session, because if the witness server fails, the principal and mirror continue to form a quorum. (For further information, see the topic "Quorum in Database Mirroring Sessions" in SQL Server Books Online.)

High Availability Operating Mode

The High Availability operating mode supports maximum database availability with automatic failover to the mirror database if the principal database fails. It requires that you set safety to FULL and define a witness server as part of the database mirroring session.

The High Availability mode is best used where you have fast and very reliable communication paths between the servers and you require automatic failover for a single database. When safety is FULL, the principal server must wait briefly for responses from the mirror server, and therefore the performance of the principal server may be affected by the capability of the mirror server. Because a single database failure will cause an automatic failover, if you have multi-database applications you want to consider other operating modes. (See "Multi-Database Issues" in the Implementation section later in this paper.)

In the High Availability mode, database mirroring is self-monitoring. If the principal database suddenly becomes unavailable, or the principal's server is down, then the witness and the mirror will form a quorum of two and the mirror SQL Server will perform an automatic failover. At that point, the mirror server instance will change its role to become the new principal and recover the database. The mirror server can become available quickly because the mirror has been replaying the principal's transaction logs and its transaction log has been synchronized with the principal's.

Also, SQL Server 2005 can make a database available to users earlier in the recovery process. SQL Server database recovery consists of three phases: the analysis phase, the redo phase, and finally the undo phase. In SQL Server 2005, a newly recovered database can become available for use as soon as the redo phase is finished. Therefore if a database mirroring failover occurs, the recovered new principal database can become available for use as soon as it finishes the redo phase. Because the mirror database has been replaying transaction log records all along, all the mirror serves has to do is finish the redo process, which normally can be accomplished in seconds.

High Protection Operating Mode

The High Protection operating mode also has transactional safety FULL, but has no witness server as part of the mirroring session. The principal database does not need to form a quorum to serve the database. In this mode only a manual failover is possible, because there is no witness to fill the tie-breaker role. An automatic failover is not possible, because if the principal server fails, the mirror server has no witness server with which to form a quorum.

Since there is no witness server defined, automatic failover cannot occur and a principal server which suddenly loses its quorum with the mirror does not take its database out of service.

High Performance Operating Mode

In the High Performance operating mode, transactional safety is OFF, and the transfer of log records is asynchronous. The principal server does not wait for an acknowledgement from the mirror that all transaction log records have been recorded on the mirror. The mirror does its best to keep up with the principal, but it is not guaranteed at any point in time that all the most recent transactions from the principal will have been hardened in the mirror's transaction log.

Since the safety is OFF, automatic failover is not possible, because of possible data loss; therefore, a witness server is not recommended to be configured for this scenario. If the witness is set, a quorum is required. If the witness is not set, then a quorum is not required. Manual failover is not enabled for the High Performance mode. The only type of failover allowed is *forced service* failover, which is also a manual operation:

```
ALTER DATABASE <dbname> SET PARTNER FORCE_SERVICE_ALLOW_DATA_LOSS
```

The forced service failover causes an immediate recovery of the mirror database. It may involve potential data loss on the mirror when it is recovered, if some of the transaction log blocks from the principal have not yet been received by the mirror. The High Performance mode is best used for transferring data over long distances (that is, for disaster recovery to a remote site), or for mirroring very active databases where some potential data loss is acceptable.

Database Snapshots and the Mirror Database

Because the mirror database is in a recovering state, it is not accessible and not readable. With SQL Server 2005 Enterprise Edition and Developer Edition, you can create database snapshots to read the mirror database at a point in time. Database snapshots provide a read-only view of a database, exposing data that is consistent at the point of the snapshot creation.

You access the database snapshot just as though it were another database. When you query a database snapshot, you read original versions of any database data that has been changed after the snapshot's creation from the database snapshot's file, and you read unchanged data from the original database. The end effect is that you see database data current at the point in time that you created the snapshot. (See the topic "Using Database Snapshots with Database Mirroring" in SQL Server Books Online for more

information.)

Because database snapshots do require some overhead on the mirror server, you need to be careful about how they might impact database mirroring performance. You can only mirror to one database, so if you need to scale out to many read-only reporting servers, transactional replication is a better choice. (For more information, see "Database Mirroring and Replication" in the Implementation section later.)

Client-side Redirect

In SQL Server 2005, if you connect to a database that is being mirrored with ADO.NET or the SQL Native Client, your application can take advantage of the drivers' ability to automatically redirect connections when a database mirroring failover occurs. You must specify the initial principal server and database in the connection string, and optionally the failover partner server.

There are many ways to write the connection string, but here is one example, specifying server A as the principal, server B as the mirror, and AdventureWorks as the database name:

```
"Data Source=A;Failover Partner=B;Initial Catalog=AdventureWorks;Integrated Security=True;"
```

The failover partner in the connection string is used as an alternate server name if the connection to the initial principal server fails. If the connection to the initial principal server succeeds, then the failover partner name will not be used, but the driver will store the failover partner name that it retrieves from the principal server on the client-side cache.

Assume a client is successfully connected to the principal, and a database mirroring failover (automatic, manual, or forced) occurs. The next time the application attempts to use the connection, the ADO.NET or SQL Native Client driver will detect that the connection to the old principal has failed, and will automatically retry connecting to the new principal as specified in the failover partner name. If successful, and there is a new mirror server specified for the database mirroring session by the new principal, the driver will retrieve the new partner failover server name and place it in its client cache. If the client cannot connect to the alternate server, the driver will try each server alternately until the login timeout period is reached.

The great advantage of using the database mirroring support built into ADO.NET and the SQL Native Client driver is that you do not need to recode the application, or place special code in the application, to handle a database mirroring failover.

If you do not use the ADO.NET or SQL Native Client automatic redirection, you can use other techniques that will enable your application to fail over. For example, you could use Network Load Balancing to manually redirect connections from one server to another, while the client just connects to a virtual server name. You might also write your own redirection code and retry logic.

However, all these techniques for coordinating client redirection with a database mirroring have an important limitation. Database mirroring occurs only at the database level, not the server level. Be careful if your application relies on querying several databases on a server, or uses multi-part object names to query across several databases. When several databases reside on one server, and they are mirrored to a standby server, it is possible that one of several databases might fail over to the standby but the others remain on the original server. In that case, you might need one connection per database that you are querying, so that you do not attempt cross-database queries on a standby server where only one database is a principal and the remaining are mirrors.

Database Mirroring and SQL Server 2005 Editions

The following table shows what database mirroring features are supported by the various editions of SQL Server 2005.

Table 2: Database Mirroring and SQL Server 2005 Editions

Database Mirroring Feature	Enterprise Edition	Developer Edition	Standard Edition	Workgroup Edition	SQL Express
Partner	√	√	√		
Witness	√	√	√	√	√
Safety = FULL	√	√	√		
Safety = OFF	√	√			
Available during UNDO after failover	√	√	√		
Parallel redo	√	√			
Database Snapshots	√	√			

A few database mirroring features require SQL Server 2005 Enterprise or Developer Editions:

- High Performance mode with safety OFF (asynchronous data transfer);
- database snapshots
- Use of multiple threads for replaying the transaction log on the mirror database (parallel REDO).

SQL Express and the Workgroup Edition can be used as a witness server, but they cannot be used as a partner server in database mirroring.

Top of page

Database Mirroring Dynamics

To gain a deeper understanding of SQL Server 2005 database mirroring, it is helpful to see how a database mirroring session can change over time. This section will cover the different database states in a database mirroring session, the mechanisms of synchronous and asynchronous log record transfer, and failover sequences.

Setup and Security

Once you have identified the principal, mirror, and, optionally, the witness servers, setting up Database Mirroring consists of essentially three steps.

1. You must make a backup of the database and restore it to the intended mirror server without recovery.

   Note: The principal database must be in the FULL recovery model before you make the database backup to restore to the mirror server. Database mirroring will not work if it must transfer Bulk-logged records in the transaction log. The mirror server must have sufficient free space for file growth as does the principal database. If you want to create database snapshots on the mirror, you must provision additional free space for them as well.

   If your backup, copy, and restore takes a relatively long time, you may need to take transaction log backups on the originating database in order to keep the log size under control. However, database mirroring will not be able to initialize if transaction log records have been truncated from the log by log backups. Therefore you must restore those transaction log backups in sequence to the intended mirror database with norecovery before database mirroring will be able to initialize.

2. The servers involved in the database mirroring session must trust each other. For local communication such as a domain, trusting means that each SQL Server instance login must have rights to connect to the other mirroring server, and do its endpoints. You establish this first by using the CREATE LOGIN command on each server, followed by the GRANT CONNECT ON ENDPOINT command. (See "Example of Setting Up Database Mirroring Using Windows Authentication" in SQL Server Books Online.)

   For communication across non-trusted domains, you must use certificates. If you use the CREATE CERTIFICATE statement to create a self-signed certificate, most of the database mirroring certificate requirements will be met. You must also make sure that the certificate is marked ACTIVE FOR BEGIN_DIALOG in the CREATE CERTIFICATE statement.

3. The next step is to establish database mirroring endpoints. *Establishing endpoints requires that you have system administrator rights to the SQL Server instance*. You must set up endpoints on each server that are specifically created as database mirroring endpoints. The easiest way to set up endpoints is to use the Configure Database Mirroring Security Wizard, which you can invoke by clicking the Configure Security button on the Mirroring page of the Database Properties dialog. The Configure Security dialogs will prompt you for computer names and port numbers, and optionally logins, before constructing and executing the CREATE ENDPOINT commands. You can also execute the CREATE ENDPOINT command using Transact-SQL. (See "How to: Create a Mirroring Endpoint (Transact-SQL)" in SQL Server Books Online.)

   If you are setting up database mirroring on a domain, and all SQL Server instances use the same service login and password, you do not need to create logins on each server. Similarly, on a workgroup, if all SQL Server instances use the same service login and password, you do not need to create logins on the servers. Just leave the logins blank on the Configure Database Mirroring Security Wizard when setting up endpoints.

   Each database endpoint must specify a unique port on the server. When working with SQL Server instances on separate machines, these port numbers can all be the same and the Configure Database Mirroring Security Wizard will automatically suggest port 5022 as the port. If any of the SQL Server instances are on the same machine, each instance must have a distinct port and the port numbers must be unique.

   Suppose you want to have three servers in a High Availability mirroring session. Server A will be the principal, server B the mirror, and server W the witness. For server A, the following command will create an endpoint on port 5022:

```
CREATE ENDPOINT [Mirroring]
AS TCP (LISTENER_PORT = 5022)
FOR DATA_MIRRORING (ROLE = PARTNER, ENCRYPTION = ENABLED);
```

Note that the role has been specified as PARTNER, so that this server may take on the role of principal or mirror for any given database mirroring database. The same command is issued on server B. Since server B is a SQL Server instance on a distinct physical machine, the port number is the same. Then for server W, you can issue

```
CREATE ENDPOINT [Mirroring]
AS TCP (LISTENER_PORT = 5022)
FOR DATA_MIRRORING (ROLE = WITNESS, ENCRYPTION = ENABLED);
```

Note that for server W, the role is specified as WITNESS.

By default, the endpoint is not started. You can next start each endpoint using the following query on each server:

```
ALTER ENDPOINT [Mirroring] STATE = STARTED;
```

Optionally, you can insert the STATE option in the CREATE ENDPOINT command. This process is repeated on each server.

When you create an endpoint using CREATE ENDPOINT, you can restrict access by IP address using the protocol specific arguments. You can restrict access to a particular set of IP addresses by combining the RESTRICT_IP with ALL option, and the EXCEPT_IP with the list of just those special IP addresses you want. (See "CREATE ENDPOINT" in SQL Server Books Online.)

You can inspect the database mirroring endpoints on a server by querying the sys.database_mirroring_endpoints catalog view:

```
SELECT *
FROM sys.database_mirroring_endpoints;
```

4. To start database mirroring, you next specify the partners and witness. *You need database owner permissions to start and administer a given database mirroring session.* On server A, the intended principal server, you tell SQL Server to give a particular database the principal role and what its partner (mirror) server is:

```
-- Specify the partner from the principal server
ALTER DATABASE [AdventureWorks] SET PARTNER =
N'TCP://B.corp.mycompany.com:5022';
```

The partner name must be the fully qualified computer name of the partner. Finding fully qualified names can be a challenge, but the Configure Database Mirroring Security Wizard will find them automatically when establishing endpoints.

The fully qualified computer name of each server can also be found running the following from the command prompt:

```
IPCONFIG /ALL
```

Concatenate the "Host Name" and "Primary DNS Suffix". If you see something like:

Host Name . . . . : A
Primary Dns Suffix . . . . : corp.mycompany.com

Then the computer name is just A.corp.mycompany.com. Prefix 'TCP://' and append ':<port number>' and you then have the partner name.

On the mirror server, you would just repeat the same command, but with the principal server named:

```
-- Specify the partner from the mirror server
ALTER DATABASE [AdventureWorks] SET PARTNER =
N'TCP://A.corp.mycompany.com:5022';
```

On the principal server, you next specify the witness server:

```
-- Specify the witness from the principal server
ALTER DATABASE [AdventureWorks] SET WITNESS =
N'TCP://W.corp.mycompany.com:5026';
```

You do not need to execute any additional commands on the witness server after the initial CREATE ENDPOINT.

Finally, you specify the safety level of the session, on the principal server:

```
-- Set the safety level from the principal server
ALTER DATABASE [AdventureWorks] SET SAFETY FULL;
```

At this point, mirroring will start automatically, and the principal and mirror servers will synchronize.You can adjust the timeout value for determining partner outage, using the TIMEOUT parameter to ALTER DATABASE. For example, to change the TIMEOUT value to 20 seconds (the default is 10), on the principal server issue:

```
-- Issue from the principal server
ALTER DATABASE [AdventureWorks] SET PARTNER TIMEOUT 20;
```

Finally, you can adjust the size of the redo queue on the mirror by issuing the ALTER DATABASE with the REDO_QUEUE option on the principal server. The following query will set the redo queue to 100 megabytes on the mirror:

```
-- Issue from the principal server
ALTER DATABASE [AdventureWorks] SET PARTNER REDO_QUEUE 100MB;
```

Once you have specified the partners, mirroring will start immediately.

Database Mirroring Catalog Views

A database mirroring session consists of the relationship formed between the partner serves and potentially the witness server. Each of the participating servers keeps some metadata about the session and the current state of the databases. You can inspect the session on the principal or mirror server by querying the *sys.database_mirroring* catalog view. The witness server metadata is returned using a different catalog view: *sys.database_mirroring_witnesses*. (For a fuller description of all the columns available in both catalog views, see "sys.database_mirroring" and "sys.database_mirrroing_witnesses" in SQL Server Books Online.)

A good way to get started in understanding how a database mirroring session works, and the states the databases can be in, is by inspecting the data from the catalog views. Let's start with a High Availability session (safety is FULL and there is a witness server). The following query returns the descriptions for basic database mirroring session information about either the principal or the mirror.

```
SELECT
      DB_NAME(database_id) AS 'DatabaseName'
    , mirroring_role_desc
    , mirroring_safety_level_desc
    , mirroring_state_desc
    , mirroring_safety_sequence
    , mirroring_role_sequence
    , mirroring_partner_instance
    , mirroring_witness_name
    , mirroring_witness_state_desc
    , mirroring_failover_lsn
FROM sys.database_mirroring
WHERE mirroring_guid IS NOT NULL;
```

The following is an analogous query returns relevant descriptive session information about the witness server that you run on the witness.

```
SELECT
      Database_name
    , safety_level_desc
    , safety_sequence_number
    , role_sequence_number
    , is_suspended
```

```
      , is_suspended_sequence_number
      , principal_server_name
      , mirror_server_name
FROM sys.database_mirroring_witnesses;
```

Now let's compare the output of both queries in a typical database mirroring session. Assume you've just set up mirroring from server A to server B with safety as FULL. (For sample code to set up the following configuration, see "Setup and Security" in Implementing Database Mirroring, later.) The witness server is server W and you're mirroring the AdventureWorks database. Table 3 shows the possible output of the above queries for the two partners.

Table 3: The output of sys.database_mirroring on a sample High Availability session for both partners.

Partnermetadatacolumn	Principalvalues:Server A	Mirrorvalues:Server B
db_name(database_id)	AdventureWorks	AdventureWorks
mirroring_role_desc	PRINCIPAL	MIRROR
mirroring_safety_level_desc	FULL	FULL
mirroring_state_desc	SYNCHRONIZED	SYNCHRONIZED
mirroring_safety_sequence	1	1
mirroring_role_sequence	1	1
mirroring_partner_instance	TCP://B.corp.mycompany.com:5022	TCP://A. .corp.mycompany.com:5022
mirroring_witness_name	TCP://W.corp.mycompany.com:5022	TCP://W.corp.mycompany.com:5022
mirroring_witness_state_desc	CONNECTED	CONNECTED
mirroring_failover_lsn	95000000007300001	95000000007300001

Note that each partner in the database mirroring session keeps all the same metadata, from that partner's perspective. Each partner keeps its own database name, the safety setting of the entire session, the mirroring state of the database, and two sequence counters.

- The mirroring_safety_sequence counter is kept on both partners and is incremented whenever the safety setting changes.
- The mirroring_role_sequence counter is kept on both partners and the witness and is incremented whenever a failover occurs.

Both partner database states and the witness state are kept by each partner server:

- The mirroring_state_desc shows the state that the partner database has in the session.
- The mirroring_witness_state_desc shows the state of the witness in the session.

Last, each partner contains the mirroring_failover_lsn. An LSN is a log sequence number, a number which uniquely identifies each transaction log record. The mirroring partners store the highest LSN +1 from the last set of log records that were hardened for the session. In the table above, because there was little activity in the principal database, the mirroring failover lsn was the same at both principal and witness. When a lot of data is being transferred, you may notice that the principal's value will be ahead of the mirror's value, because the mirror will be running somewhat behind.

Now let's compare the metadata found on the witness. The following table shows comparable information from the witness server metadata:

Table 4: The output of sys.database_mirroring_witnesses on the Witness server, correlated with partner metadata.

Witnessmetadatacolumn	Witnessvalues	Corresponding partnermetadatacolumn
database_name	AdventureWorks	db_name(database_id)
safety_level_desc	FULL	mirroring_safety_level_desc
safety_sequence_number	1	mirroring_safety_sequence
role_sequence_number	1	mirroring_role_sequence
is_suspended	0	
is_suspended_sequence_number	1	

principal_server_name	TCP://A. .corp.mycompany.com:5022	
mirror_server_name	TCP://B.corp.mycompany.com:5022	

Note that the witness server's metadata contains the safety description, the safety sequence number, and the role sequence number, though under slightly different names. The witness also keeps data about whether the session is suspended, and keeps the principal and mirror server names. Note that the witness sever catalog view does not report the mirroring failover lsn, and it does not keep the database states.

All the metadata required for database mirroring (in particular the mirroring failover lsn and partner server names) are kept by the mirroring partners. The witness only keeps data necessary for its role as a witness in a High Availability mode, in particular the role sequence number, which tracks the number of role changes in the session. This counter is used to help decide when a principal server can make a role change, as you'll learn in the scenarios in the next section.

Database Mirroring States and Transitions

Database states for each server are kept during the database mirroring session, recorded on each partner server, and reported by the sys.database_mirroring catalog view. The *mirroring_state* column returns a number for the state, and the *mirroring_state_desc* column returns the descriptive name for the state. (For a complete list of database state numbers and descriptive names, see "sys.database_mirroring" in SQL Server Books Online.) State information about the witness is also reported from the same catalog view.

In addition to the states reported for each database, there are three phrases that are useful in describing the servers and databases involved in database mirroring.

- The data on the principal is **exposed** when it is processing transactions but no log data is being sent to the mirror.
- **Cannot serve the database** - When a principal server does not allow any user connections to the database and any transactions to be processed.
- A server is **isolated** when it cannot contact any of the other servers in the database mirroring session, and they cannot contact it.

When a principal database is *exposed*, it is active with user connections and processing transactions. However, no log records are being sent to the mirror database, and if the principal should fail, the mirror will not have any of the transactions from the principal from the point the principal entered the exposed state. Also, the principal's transaction log cannot be truncated, so the log file will be growing indefinitely.

When a witness has been set, if the principal server cannot form a quorum with another server, it will stop *serving the database*. It will not allow user connections and transactions on the principal database, and will disconnect all current users. As soon as it can form a quorum again, it will return to serving the database.

A server may be operational but communication lines are down between it and both other servers in the database mirroring session. In that case, we'll call the server *isolated*. If a witness has been set, then, if the principal server becomes isolated., it will no longer be able to serve the database, because there is no server in the session with which it can form a quorum.

Now let's focus on the database mirroring states, starting with the principal server and database.

Principal Server Database States

When safety is FULL, the normal operating state for the principal database is the SYNCHRONIZED state.
When safety is OFF, the normal operating state for the principal database starts as the SYNCHRONZING state. Once the mirror has caught up, the state goes to SYNCHRONIZED and stays there regardless of how far behind it is.

- If safety is FULL, the principal database always starts off in the SYNCHRONIZING state, and transitions to the SYNCHRONIZED state when the principal and mirror transactions logs are synchronized.
- If safety is FULL and the principal server is disconnected from the witness server but can still process transactions, the database is exposed.

The following table shows the states that a principal database can be in, and some of the events that can cause a transition to another state.

Table 5: Principal Database states with a witness set and with safety FULL and safety OFF.

Safety	Principal Initial State	Event	Resulting State	Quorum	Exposed	Able to serve db
FULL	SYNCHRONIZING	Synchronization occurs	SYNCHRONIZED	Y	N	Y
FULL	SYNCHRONIZED	Session is paused	SUSPENDED	Y	Y	Y
FULL	SYNCHRONIZED	Redo errors on the mirror	SUSPENDED	Y	Y	Y

FULL	SYNCHRONIZED	Mirror unavailable	DISCONNECTED	Y, with witness N, no witness	Y -	Y N
OFF	SYNCHRONIZING	Session is paused	SUSPENDED		Y	Y
OFF	SYNCHRONIZING	Redo errors on the mirror	SUSPENDED	-	Y	Y
OFF	SYNCHRONIZING	Mirror unavailable	DISCONNECTED	Y, with witness N, no witness	Y -	Y N

Table 6 Principal Database states with no witness set and with safety FULL and safety OFF.

Safety	Principal Initial State	Event	Resulting State	Exposed	Able to serve db
FULL	SYNCHRONIZING	Synchronization occurs	SYNCHRONIZED	N	Y
FULL	SYNCHRONIZED	Session is paused	SUSPENDED	Y	Y
FULL	SYNCHRONIZED	Redo errors on the mirror	SUSPENDED	Y	Y
FULL	SYNCHRONIZED	Mirror unavailable	DISCONNECTED	Y	Y
OFF	SYNCHRONIZING	Session is paused	SUSPENDED	Y	Y
OFF	SYNCHRONIZING	Redo errors on the mirror	SUSPENDED	Y	Y
OFF	SYNCHRONIZING	Mirror unavailable	DISCONNECTED	Y	Y

When safety is FULL, the principal first enters the SYNCHRONIZING state and as soon as it synchronizes with the mirror, both partners enter the SYNCHRONIZED state. When safety is OFF, the partner databases start with the SYNCHRONIZING state. Once the mirror has caught up, the state goes to SYNCHRONIZED and stays there regardless of how far behind it is.

For both safety settings, if the session is paused or there are redo errors on the mirror, the principal enters the SUSPENDED state. If the mirror becomes unavailable, the principal will enter the DISCONNECTED state.

In the DISCONNECTED and SUSPENDED states:

- When a witness has been set, if the principal can form a quorum with the witness or mirror server, the principal database is considered *exposed*. That means the principal database is active with user connections and processing transactions. However, no log records are being sent to the mirror database, and if the principal should fail, the mirror will not have any of the transactions from the principal from the point the principal entered that state. Also, the principal's transaction log cannot be truncated, so the log file will be growing indefinitely.
- When a witness has been set, if the principal cannot form a quorum with another server, it cannot serve the database. All users will be disconnected and no new transactions will be processed.
- When safety is OFF, the principal database is considered exposed, because no transaction log records are being sent to the mirror.

Note: Management Studio's Object Explorer will report the principal database states next to the database name in the Server tree graph. It will report a principal's SYNCHRONIZED state as 'Principal, Synchronizing' and the DISCONNECTED state as 'Principal, Disconnected.'

Mirror Server Database States

The mirror database has the same database states as the principal database, but because the mirror is always in a nonrecovered state, it never serves the database while in the mirror role. The following table shows the most common events that can cause a database state change for the mirror server and database.

Table 7: Mirror database mirroring states with safety FULL and safety OFF.

Safety	Mirror Server State	Event	Resulting State
FULL	SYNCHRONIZING	Synchronization occurs	SYNCHRONIZED
FULL	SYNCHRONIZED	Session is paused	SUSPENDED
FULL	SYNCHRONIZED	Redo errors on the mirror	SUSPENDED

FULL	SYNCHRONIZED	Principal database unavailable	DISCONNECTED
OFF	SYNCHRONIZING	Session is paused	SUSPENDED
OFF	SYNCHRONIZING	Redo errors on the mirror	SUSPENDED

Just as with the principal, Management Studio's Object Explorer will report some of the mirror database states next to the database name in the Server tree. It will report a mirror's SYNCHRONIZED state as 'Mirror, Synchronizing' and the DISCONNECTED state as 'Mirror, Disconnected.'

Witness Server States
The witness server has three states in the sys.database_mirroring catalog view, CONNECTED, DISCONNECTED, and UNKNOWN.

Table 8: Witness server states (as recorded on the partner servers).

Witness Server State	Event	Resulting State
CONNECTED	Witness server unavailable	DISCONNECTED
	Principal unable to initialize the mirror	UNKNOWN

Since the state of the witness server is actually recorded in the partner servers, and not on the witness server, these states are set from the vantage point of the partners. So seeing a DISCONNECTED status for the witness server implies that the partners are disconnected from the witness. When database mirroring starts up, if the mirror cannot initialize with the principal, the witness will enter an UNKNOWN state.

Transferring Transaction Log Records
The sequence of operations that a principal and mirror SQL Server use in transferring messages and blocks of log records varies depending upon the safety setting. First we examine the synchronous transfer, then asynchronous.

When SQL Server records transaction events in the transaction log, the log records are briefly put into a log buffer before being written to disk. In database mirroring, each time a log buffer is written to disk (hardened), the principal sends the same block of log records to the mirror.

- When safety is FULL, as the principal SQL Server hardens its block of log records, it sends the same block to the mirror and treats both the local I/O to the log and the remote mirror server's I/O to its log as essentially the same. The transfer is called synchronous because the principal will wait for both its local I/O (hardening) and for the mirror's response that it has completed its I/O (hardening) before the transaction is considered completed.

  Each time the principal or mirror hardens its log buffer, it records the highest log sequence number (LSN) + 1 of the buffer in metadata as the mirroring_failover_lsn. The mirroring_failover_lsn is used to negotiate the latest guaranteed point in the transaction log, so that the two partner databases can synchronize initially and synchronize after a failover.

  The mirroring failover lsn will usually be ahead on the principal when the principal is sending log records to the mirror. The mirror, as it hardens log records, will record its mirroring_failover_lsn and respond back to the principal, but by the time the principal receives acknowledgement from the mirror, the principal may have hardened a new set of log records.

  Table 9 shows a sample sequence of events between the principal and the mirror with safety FULL.

  Table 9: A Safety FULL (Synchronous transfer) example sequence of events.

Server A	Server B
Principal, Synchronized	Mirror, Synchronized
A multi-statement transaction begins containing data modifications	
Transaction log records for the principal database are inserted into a transaction log buffer	
The transaction log buffer is written to disk (hardened), the block of log records is sent to the mirror and the principal records the log block's mirroring_failover_lsn and the principal waits for confirmation from the mirror	
	The mirror server receives the block of log records into a

	transaction log buffer
	The mirror writes the log buffer to disk, records the mirroring_failover_lsn, and notifies the principal that the log block has been hardened
The principal receives notification that the block of log records was written to disk by the mirror	Mirror continues to replay transaction log records in its REDO queue
A COMMIT for the transaction is entered into the transaction log buffer	
The transaction log buffer is written to disk (hardened), the block of log records containing the COMMIT is sent to the mirror, the principal records the block's mirroring_failover_lsn and the principal waits for confirmation from the mirror	
	The mirror server receives the block of log records into a transaction log buffer
	The mirror writes the log buffer to disk, records the mirroring_failover_lsn, and notifies the principal that the log block has been hardened
The principal receives notification that the block of log records was written to disk by the mirror and the transaction is considered committed	Mirror replays transaction log records including the COMMIT, from its REDO queue, changing data pages
New transactions are written to the principal's log buffer.	

The key point in the sequence above is that when safety is FULL, the principal server hardens its log buffer and sends a copy of the log records from the buffer to the mirror, both at the same time. It then waits for the completion of its own I/O and the I/O of the mirror before considering the transaction complete. When the principal receives its response from the mirror, the principal can then proceed to the next hardening.

Despite the close coordination between principal and mirror when safety is FULL, database mirroring is not a distributed transaction and does not use a two-phase commit.

- In database mirroring, you have two transactions being played out on two servers, not one transaction distributed across them.
- Database mirroring does not use the partner servers as resource managers in a distributed transaction
- Database mirroring transactions do not go through prepare and commit phases.
- Most importantly, unlike a distributed transaction, failures to commit on the mirror will not cause a transaction rollback on the principal.

- When safety is OFF, the principal does not wait for acknowledgment from the mirror server, so the number of committed transactions on the principal can get ahead of the mirror, as shown in Table 10.

Table 10: A Safety OFF (Asynchronous transfer) example sequence of events.

Server A	**Server B**
Principal, Synchronizing	Mirror, Synchronizing
A multi-statement transaction begins containing data modifications	
Transaction log records for data modifications are written to a transaction log buffer	
The transaction log buffer is flushed to disk (hardened), the block of log records is sent to the mirror and the principal records the block's mirroring_failover_lsn	

A COMMIT for the transaction is entered into a transaction log buffer, along with other transaction activity	The mirror server receives the block of log records into a transaction log buffer
The transaction log buffer is hardened to disk and the log block containing the COMMIT is sent to the mirror	The mirror writes the log buffer to disk, records the mirroring_failover_lsn, and notifies the principal that the log block has been hardened
The transaction is considered committed	Mirror continues to replay transaction log records in its REDO queue
	The mirror server receives the block of log records into a transaction log buffer
	The mirror hardens the log block by writing it do disk and records the mirroring_failover_lsn and notifies the principal that the log block has been hardened

Database Mirroring Role Change
Database mirroring failover is an issue that can be considered from the standpoint of the database mirroring servers or the application. From the standpoint of the database mirroring servers, failover is the conversion of the mirror server into a principal server and the use of the newly recovered database as the principal database in the session. The failover may be automatic, manual, or forced.

- Automatic - occurs only in the High Availability mode (safety is FULL and a witness is part of the session
- Manual - occurs in the High Availability and High Protection operating modes (safety is FULL) and the partner databases are both SYNCHRONIZED.
- Forced service (allow data loss) - used primarily in the High Performance (safety OFF) mode to immediately and manually recover the mirror database.

When safety is FULL, the best way to reverse the roles of the servers is to use manual failover, not forced service.

Automatic Failover
Automatic failover is a feature of database mirroring in the High Availability operating mode (safety FULL with a witness). In most cases, SQL Server can accomplish an automatic database mirroring failover in a few seconds. It can do this in part because the SQL Servers involved in a database mirroring session all test each other's presence. This process is called a 'ping', though it involves much more than an ordinary IP address ping. The mirror and witness servers contact the principal server for the presence of the physical server, for the presence of the SQL Server instance, and the availability of the principal database. Similarly, the principal and the witness each ping the mirror server for availability of the physical server, the SQL Server instance, and the recovering state of the mirror database.

Suppose the database mirroring session has been set up with safety FULL and with a witness server. The mirror server, Server B, finds through its ping that the principal Server A is unavailable. Server B communicates with the witness server and receives confirmation that the witness also cannot see Server A. Then Server B will form a quorum with the witness server and promote itself into the principal role. It will recover its database and notify the witness that it is now has the principal role (though the database will be in a disconnected state, and the new principal database's transaction log cannot be truncated)

Server B's new principal database continues to replay transaction log activity, but it has been in a redo state continuously and in most cases there will be little left to do. In database mirroring for all SQL Server editions, the new principal database becomes available as soon as it finishes its redo state. When the database enters the undo state, it becomes available for user connections. Finishing the redo state occurs normally in a few seconds, although the remaining undo phase could be much longer. In database mirroring, the new principal database becomes available for serving user connections as soon as the redo process is finished. The new principal server B's database is in a DISCONNECTED state and is exposed, but it can place its database into service just as soon as the redo phase is completed.

Usually it will take more time to redirect an entire application from the old principal to the new principal than a database mirroring automatic failover will take. The application must detect and retry connections, which may add some time to the process. In addition, if new logins using SQL Server authentication have been added to the servers, you may need to map those logins to the new principal using the system stored procedure *sp_change_users_login*. Complete application failover may also be delayed if any critical objects on the old principal, such as SQL Agent jobs, have not also been copied to the new principal server. (For more information, see "Preparing the Mirror server for Failover" in the Implementation section later.)

Now suppose the old principal server comes online. There is a negotiation process that must occur between the two servers if they are reversing roles, as in a manual failover, or an automatic failover where the old principal is repaired fairly quickly. Before

mirroring can restart, the two partner servers need to find out how they can synchronize with each other. The mirroring failover lsn plays a critical role in this process.

Server A (the new mirror) is behind, but it is not clear how much. Server A reports to server B (the new principal) the last mirroring failover lsn that it received from server B. Server B, on the other hand, has had committed work which has brought its mirroring failover lsn to a more recent state. Both servers then agree that server B has the correct failover lsn, and that server A must catch up to it. Server B sends a sufficient number of transaction records to server A that it can replay in order to become synchronized.

Manual Failover
A manual failover is a way of causing the two partner servers to reverse their roles in an orderly way and without any errors. It requires that safety be set to FULL, and that the principal and mirror databases are in the SYNCHRONIZED state.

You cause a manual failover by invoking the ALTER DATABASE command on the principal server:

```
ALTER DATABASE AdventureWorks SET PARTNER FAILOVER;
```

or by clicking the Failover button in the Database Properties/Mirroring dialog in Management Studio. A manual failover causes current users to be disconnected and rolls back any unfinished transactions from the old principal database. It will recover the mirror database by finishing all completed transactions in the redo queue, and rolling back (in the undo phase) unfinished transaction. The old mirror is assigned the principal role, and the old principal database takes on the new role of mirror. The two servers will negotiate a new starting point for mirroring based on their mirroring failover lsns, and proceed with their roles reversed.

You can use a manual failover as a way of accomplishing a 'rolling upgrade' to the server operating systems or SQL Server instances, provided you upgrade the mirror server first before initializing mirroring. For more information, see 'Manual Failover' in SQL Server Books Online.

Forced Service
You can cause a forced service on the mirror by invoking the ALTER DATABASE command:

```
ALTER DATABASE AdventureWorks SET PARTNER FORCE_SERVICE_ALLOW_DATA_LOSS;
```

Normally this is only useful if you have safety OFF, and if the principal is no longer operating. You can also use this command with safety FULL but if the recovered mirror server cannot form a quorum, it cannot serve the database. Therefore it's better to use this command only with safety OFF (the High Performance mode). Some data may be lost because the asynchronous data transfer may not have kept the mirror fully up to date with committed transactions from the principal.

Top of page

Database Mirroring Availability Scenarios

In this section you'll explore the expected availability outcomes of database mirroring scenarios based on the following two types of events:

- One or more servers or databases are down
- One or more communication lines between server are broken

A server loss could occur when either one of the partner databases, or one of the SQL Server instances, become unavailable. Alternately, the lines of communication between the database mirroring partner servers may be interrupted, even though the servers themselves continue to operate.

The following scenarios consider *simultaneous failures of two components as a sequential failure of one component followed by another*. For example, if Servers A and B appear to fail simultaneously, the mirroring system will treat that event as a sequence: Server A followed by Server B, or vice versa.

You can use the following rules to determine the expected results of an unavailability event:

1. **When a witness has been set, the principal server requires a quorum with at least one other server to keep its database in service.**

   If a principal cannot form a quorum, it can no longer serve the database.

2. **When safety is FULL, if neither the mirror nor the witness can see the principal, the mirror server can form a quorum with the witness and change its role to become a new principal server, assuming a the mirroring session was SYNCHRONIZED when the principal went away.**

   This is an automatic failover.

   When safety is FULL, if the principal has done work while in quorum with the witness but disconnected from the

3. **mirror, a principal failure will not allow the mirror to form a quorum with the witness and take on the role of a new principal server.**

   This prevents work being lost by the session.

4. **When safety is FULL, if a failed principal server rejoins a session after being down or isolated, and the old mirror has previously taken on the role of new principal (in quorum with the witness), the old principal will take on the role of new mirror in the session.**

   During the failover event, the mirror and the witness incremented the mirroring role sequence counter. Because the old principal's mirroring role sequence counter is less than that on the other partner server and the witness server, those two servers have formed a quorum before the old principal rejoined the session, and new work may have occurred on the new principal. The old principal takes on the mirror role in the session.

5. **When a witness has been set, the principal requires a quorum to serve the database. If the witness has not been set (the status of the sys.database_mirroring column "mirroring_witness_name" is NULL), then a quorum is not required by the principal to serve the database.**

   No witness means that no automatic failover can occur. If there is no automatic failover, there is never a problem with "split brain," and the principal does not need to contact another server to form a quorum.

High Availability Scenarios with Server Loss

The purpose of database mirroring in High Availability operating mode is to keep the database as available as possible. In this mode, database mirroring will bring the mirror database into service quickly if the principal database becomes unavailable. In the following set of scenarios, we begin with a High Availability configuration with three independent servers.

In the following High Availability scenarios, Server A starts out as the principal, Server B is the mirror, and Server W is the witness, as illustrated in Figure 1.

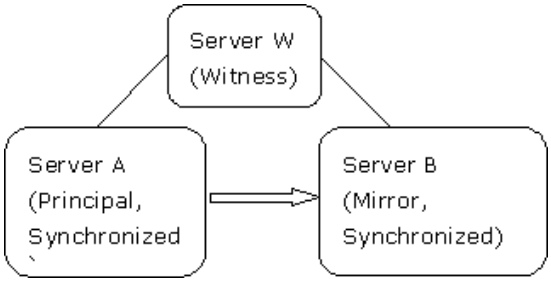

Figure 1: The sample database mirroring session starts in the High Availability operating mode.

Physically, all three servers may be on the same site and connected through a local network, or they may be at independent sites connected through a WAN. Server A and Server B may change roles, but Server W always remains the witness server.

Now consider what would happen if one of the servers goes down.

Scenario HASL1.1: Principal Server Loss

The following scenario considers what happens when the principal server is lost in a High Availability scenario. Figure 2 shows the various roles and how they can change among the partners.

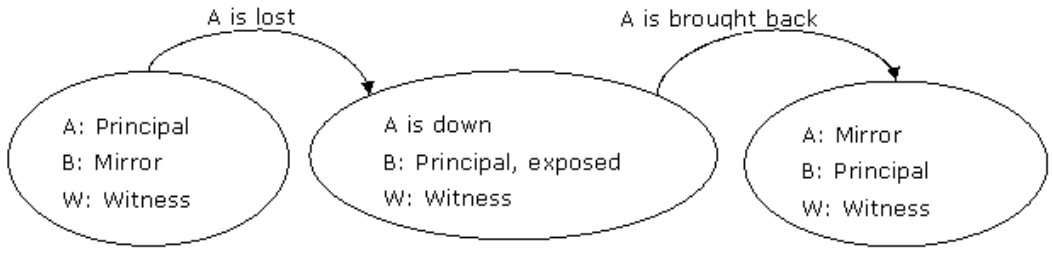

Figure 2: In High Availability mode, when the principal Server A fails, a failover occurs.

After the principal server A is lost, the mirror and the witness servers form a quorum and an automatic failover occurs. If you are able to bring back the original principal server, then it will take on the mirror role.

Note: To cause a failover in High Availability mode, a failure could occur at many levels: the physical server could be down, the SQL Server instance on the principal could be stopped or failed, or the principal database on the server could be unavailable or suspect. When a principal server is lost in the following scenarios, it could be cause by any of these events.

Because Servers B and W can form a quorum, and neither can see server A, Server B can promote itself as the new principal. However, without a mirror server, the mirroring session is considered exposed.

When you bring back Server A, it becomes the new principal and the mirroring session is no longer exposed

A single server failure may be a rare event. An even rarer event would be to have two servers fail. Even though rare, it is useful to inspect the results.

Two servers might fail simultaneously or nearly at the same time, but from the database mirroring standpoint, the result will be as if one server failed followed by another. Therefore these scenarios only consider what happens when servers fail in sequence.

The next two scenarios consider what occurs when the principal Server A failure is followed by two alternate server failures:

- a failure of the new principal Server B;
- a failure of the witness Server W.

Scenario HASL1.2: Principal Server Loss followed by New Principal Loss
As you saw in the previous scenario, in the High Availability mode, if the principal server is lost first, a failover occurs. Figure 3 shows that if the new principal server fails next, no matter how you restore the servers, the new principal (Server B) keeps its principal role.

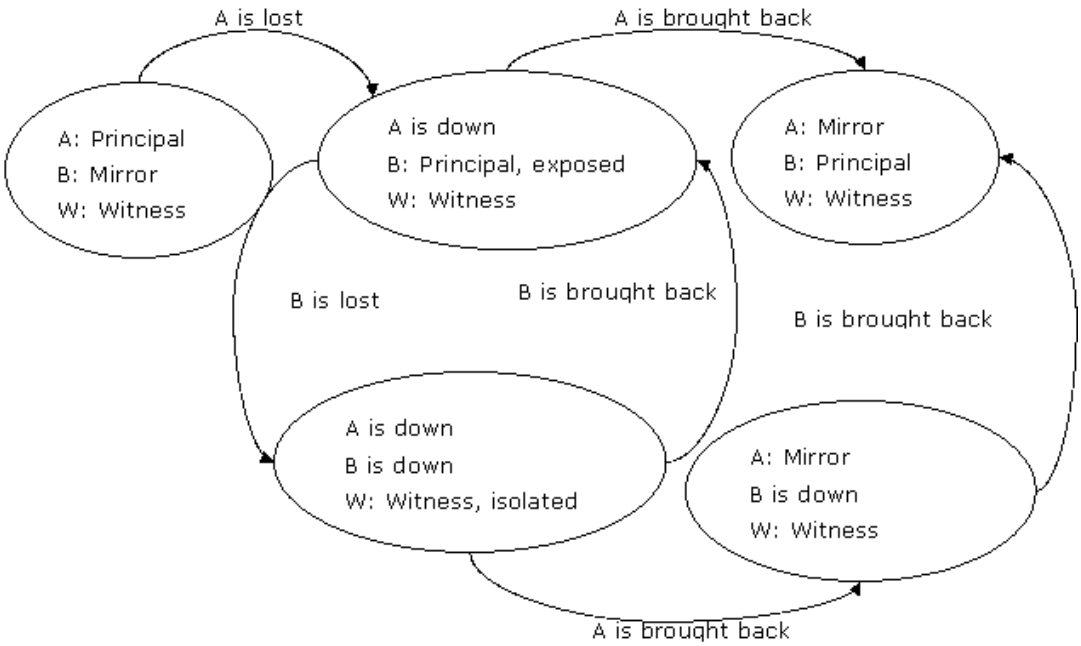

Figure 3: Role changes due to principal server loss followed by new principal loss.

After Server A fails, Server B becomes the new principal, but it cannot send data to a mirror, so the principal is exposed, even though it can still serve the database. When a Server A failure is followed by Server B failing, there is no mirroring because Server B is down.

If you bring back Server A first, it detects from the mirroring_role_sequence number at the Witness W that the witness has since formed a new quorum. Server A adopts the role of mirror and waits for Server B to come back. As soon as Server B is brought back, it will begin the process of mirroring to Server A. If you bring back Server B first, then you are back to the original scenario shown in HASL1.1.

Note: if Server W is lost after Server A and then Server B is lost, bringing all three servers down, the switched roles for Server A and Server B will persist independently of the server restoration sequence.

Scenario HASL1.3. Principal Server Loss followed by Witness Loss
When the principal server is lost, failover occurs. The witness server may fail next, as shown in Figure 4.

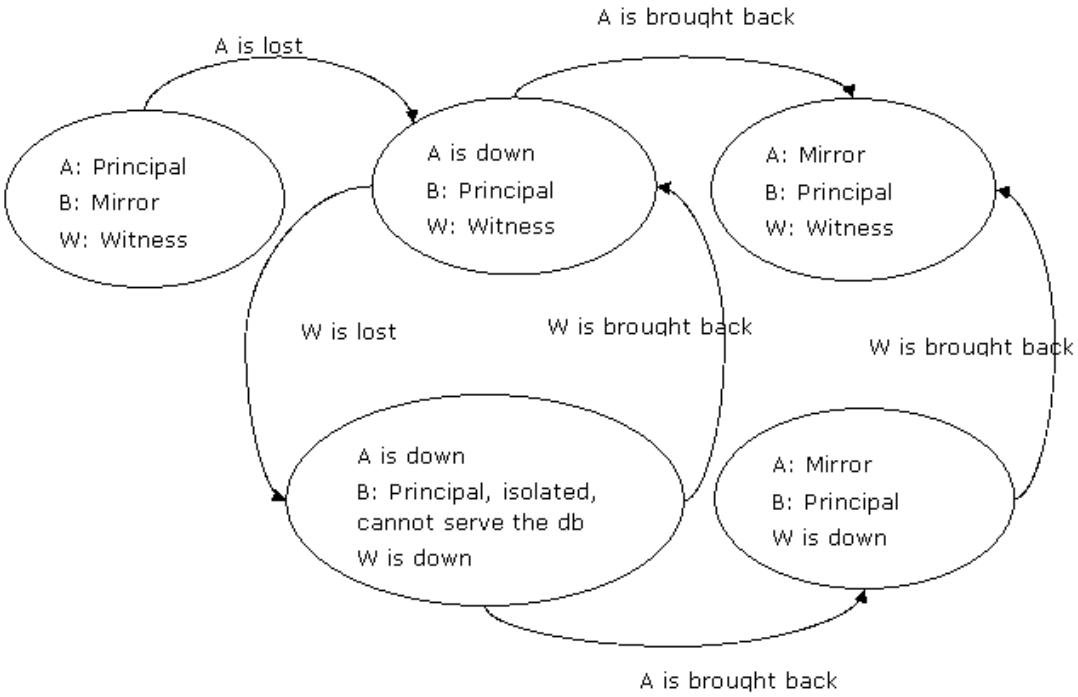

Figure 4: When a witness server loss follows an initial principal server loss, the new principal cannot serve the database.

When the witness server W is lost after the principal Server A is lost, the new principal Server B is still the principal but it is isolated, cannot form a quorum, and cannot serve the database.

If you bring back Server A first, Server B's mirroring_role_sequence number will be one greater than Server A's, because a failover has occurred. This indicates to Server A that Server B now has the principal role after Server A did. Server A forms a quorum with server B and becomes the mirror, and then both servers synchronize. Until Server W is brought back, no automatic failover can occur.

Note: if Server B is lost after Server A and then Server W is lost, the new roles for Server A and Server B will persist independently of the server restoration sequence.

Scenario HASL2.1 Mirror Server Loss
If the mirror server is lost first, the principal server is considered exposed because it cannot send data to the mirror. Figure 5 shows how the roles behave when Server B, the mirror, is lost.

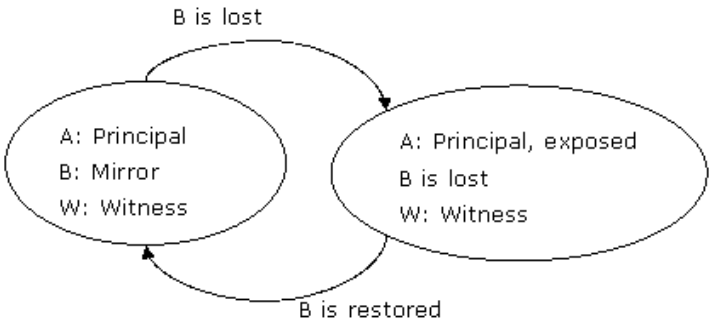

Figure 5: In High Availability mode, when the mirror Server B fails, no failover occurs.

No automatic failover will occur and the partners will not exchange roles. When Server B is restored, all three servers return to their original roles and states.

The following tables show the database states and quorum during the loss and recovery of the mirror Server B.

Note that the session is exposed without a mirror, because the data is not being placed in redundant databases.

As soon as you can restore Server B, it will resume its mirror role and when the two partners are synchronized, the mirroring session is no longer considered exposed.

The next two scenarios consider what occurs when the mirror server B failure is followed by a failure of the principal Server A or witness Server W.

Scenario HASL2.2: Mirror Server Loss followed by Principal Loss
When the mirror server is followed by a loss of the principal server, the roles of the partner servers remain unchanged. Figure 6 shows how the roles change as you take various paths to bring back the servers.

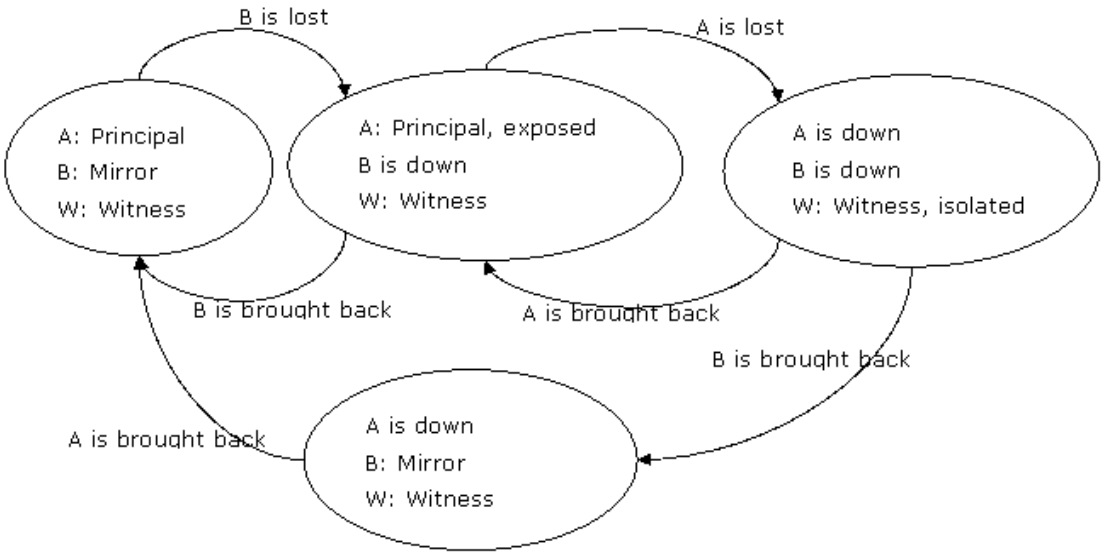

Figure 6: If a mirror server loss is followed by principal loss, the witness is isolated.

After Server B and then Server A are lost, the state is that shown in the upper-right corner of the diagram.

If you bring back Server B first, it detects from the witness Server W that Server A is still the principal and that a failover has not occurred - the mirroring_failover_lsn has not increased. As a result, Server B remains the mirror. Then bringing back Server W restores the session to its original state.

Note: if Server W is lost after Server B and then Server A are lost, bringing back the servers in any order has the same result.

Scenario HASL2.3: Mirror Server Loss followed by Witness Loss
When a mirror server loss is followed by a witness server loss, the principal server is isolated and cannot form a quorum with either other server. Therefore it must take its database out of service, as shown in Figure 7 in the upper-right corner.

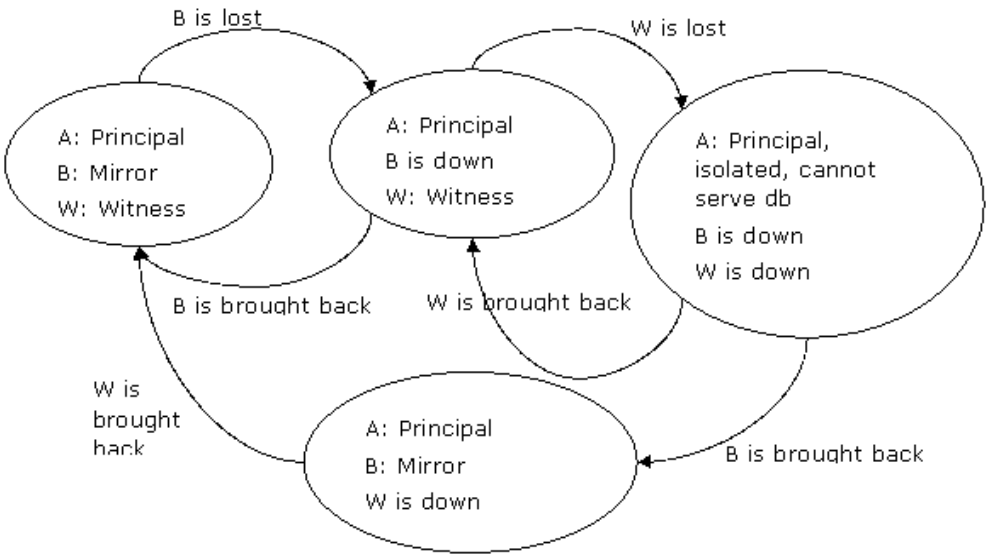

Figure 7: A mirror server loss followed by a witness loss leaves the principal server unable to serve the database.

During the server outages of mirror followed by the witness, the principal Server A retains its principal role, but because it cannot form a quorum with any other server, and safety is FULL, it cannot serve the database, and disconnects all users.

If Server B is restored first, mirroring resumes, although no automatic failover is possible without a witness.

If Server W is restored first, the scenario is the same as shown in Figure 5.

Note: if Server A is lost after Server B and then Server W were lost, bringing back the servers in any order preserves the same final results.

Scenario HASL3.1: Witness Server Loss
When the witness server fails, mirroring continues but no automatic failover is possible. Loss of just one more server will mean there is no quorum, and the principal database will no longer be able to serve the database.

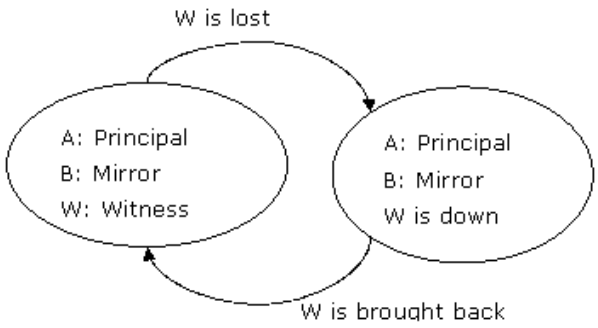

Figure 8: In High Availability mode, when the witness Server W fails first, mirroring continues.

When you restore Server W, the partner servers Server A and Server B keep their original roles.

The following tables show the changes of database state and quorum during the failure and restoration of the witness server.

The next two scenarios consider what occurs when the witness server W failure is followed by a failure of the principal Server A or mirror Server B.

Scenario HASL3.2: Witness Server Loss followed by Principal Loss
When the witness server fails first, mirroring continues, but no automatic failover is possible. If just one of the remaining two servers fails, no quorum is possible and the remaining servers will be isolated.

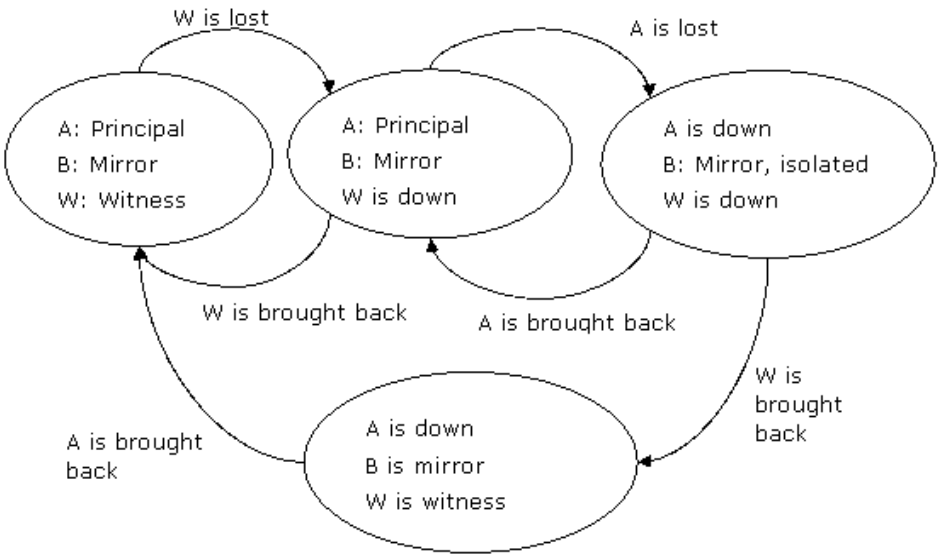

Figure 9: The partner roles remain the same after an initial witness server loss followed by a principal server loss.

If you bring back server W first, Server B detects from the witness that the last good principal was server A and so Server B remains the mirror. When you finally bring back Server A it keeps the principal role.

Note: if Server B is lost after Server W and then Server A were lost, bringing back the servers in any order will not affect the end results

Scenario HASL3.3: Witness Server Loss followed by Mirror Loss
If the Witness server fails, and then the Mirror server fails, the principal server becomes isolated. Since safety is FULL and the principal is unable to form a quorum, it can no longer serve the database, as shown in Figure 10.

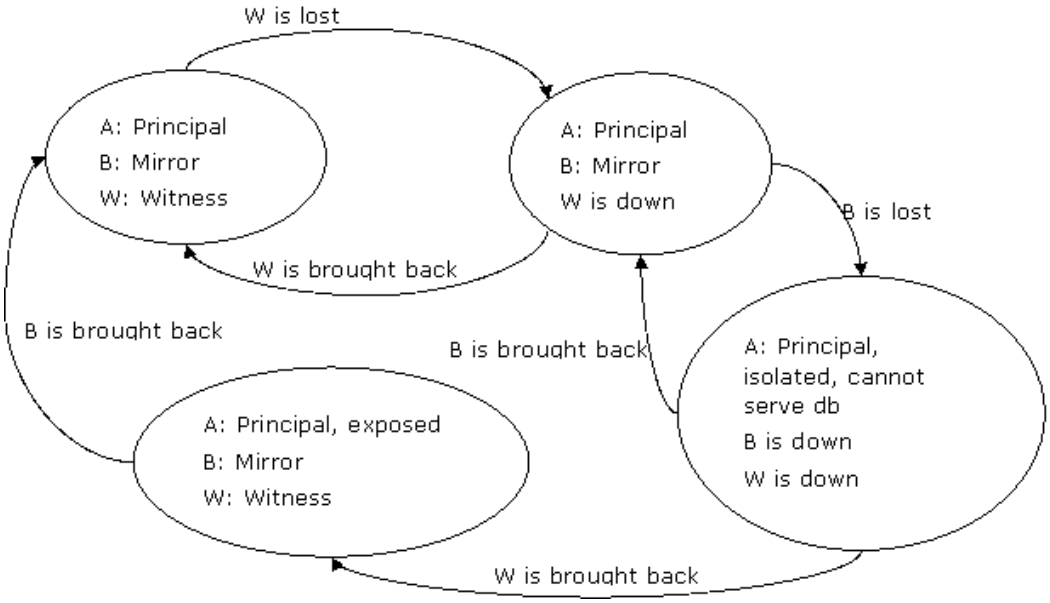

Figure 10: When the Witness server fails, followed by the Mirror server, the principal must take its database out of service.

Note: if Server A is lost after first Server W and then Server B are lost, bringing back the servers in any order will not affect the final results.

Summary: High Availability Scenarios with Server Loss
There are a number of conclusions to draw from these scenarios. In the High Availability operating mode:

1. If the principal server becomes unavailable first, an automatic failover will occur and the former mirror server will take on the principal role and make its database available for user activity. Subsequent server failures and restores will not affect the overall configuration of the mirroring with the new principal. Mirroring will continue in the reverse direction.
2. If the mirror server becomes unavailable first, an automatic failover will not occur. Subsequent server unavailability, and the restoration sequence, will not affect the mirroring partner roles.
3. If the witness becomes unavailable first, no automatic failover is possible, and the partner servers will keep their original roles. Subsequent server unavailability and restoration will not affect the partner roles.

The following table summarizes the results for High Availability scenarios where one or two servers are lost. The conditions assumed in the table are that safety is FULL and that the mirroring session servers have the following conditions:

A: Principal, SYNCHRONIZED

B: Mirror, SYNCHRONIZED

W: Witness, CONNECTED

The table shows the failover scenario highlighted in grey.

Table 11: A summary of single and dual server failures, showing partner server roles and database states.

First event	Quorum	Result	Intermediate state	Second event	Quorum	Result	Ending state	First Server Back	Result	Second Server Back	Final State
A is lost (HASL1.1)	B and W	*Failover:* Database on B is in service, session is exposed	A is down B: Principal, DISCONNECTED W: Witness, CONNECTED	B is down (HASL1.2)	None	No database in service	A is down B is down W: Witness	Bring back A	A: Mirror, DISCONNECTED B is down W: Witness	Bring back B	A: Mirror, SYNCHRONIZED B: Principal, SYNCHRONIZED W: Witness, CONNECTED
				W is down (HASL1.3)	None	No database in service	A is down B: Principal, DISCONNECTED, not in service W is down	Bring back A	A: Mirror B: Principal W is down	Bring back W	
B is lost (HASL2.1)	A and W	Database on A is in service, session is exposed	A: Principal, DISCONNECTED B is down W: Witness, CONNECTED	A is down (HASL2.2)	None	No database in service	A is down B is down W: Witness	Bring back B	A is down B: Mirror, DISCONNECTED W: Witness	Bring back A	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, CONNECTED
				W is down (HASL2.3)	None	No database in service	A: Principal, DISCONNECTED, not in service B is down W is down, DISCONNECTED	Bring back B	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W is down	Bring back W	
W is lost (HASL3.1)	A and B	Database on A is in service, session is exposed	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W is down	A is down (HASL3.2)	None	No database in service	A is down B: Mirror, DISCONNECTED W is down, DISCONNECTED	Bring back W	A is down B: Mirror, DISCONNECTED W: Witness	Bring back A	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, CONNECTED
				B is down (HASL3.3)	None	No database in service	A: Principal, DISCONNECTED, not in service B is down W is down, DISCONNECTED	Bring back W	A: Principal, DISCONNECTED, not in service B is down W: Witness	Bring back B	

34

High Availability Scenarios with Communication Loss
The High Availability operating mode requires three SQL Sever instances. If the servers are at two or three independent sites with

significant distance between them, it is quite possible that there may be communication problems between the sites. In other words, the servers may stay available but the communication lines may be interrupted. These scenarios are a little more complex than the previous set but the principles to apply are the same.

The following High Available operating mode scenarios with communication loss are covered in two sets. The first set is based on three SQL Server instances on independent sites, and therefore with three independent communications lines. The second set is based the servers being on two independent sites, with one communication line between a pair of servers and a third.

Starting with the first set, assume you have three independent communication lines between all the servers in a database mirroring session. For example, the principal, mirror, and witness could be located at three independent co-location sites. (It is also possible that the three servers could be on one site but be connected by private networks.)

The initial conditions are that Server A houses the principal database and has synchronized with Server B as its partner, which has the mirrored database. Safety is set to FULL, and a witness server (Server W) is part of the database mirroring session. Figure 11 shows this initial configuration.

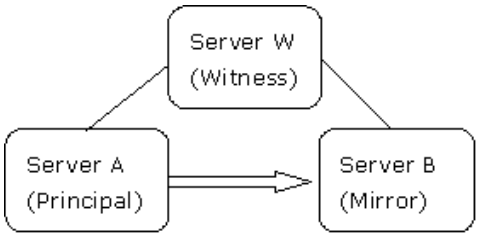

Figure 11: The initial High Availability configuration for three independent servers has three independent communication lines

Note: for an explanation of the diagrams on the following pages, see "High Availability Mode with Server Loss" above.

Based on Figure 11, there are three different lines that could break first: A/B, A/W, and B/W. Note that when a single communication line is down all three servers still operate. Only the line between the principal and the mirror has any effect, as shown in Table 12.

Table 12: Summary of single-line communication line breaks.

Initial condition	Event	Quorum	Result	Condition
A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, CONNECTED	A/B link broken	*A and W*	Database on A in service, exposed	A: Principal, DISCONNECTED B: Mirror, DISCONNECTED W: Witness, CONNECTED
	A/W	A and B	Database on A in service	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, CONNECTED
	B/W	A and B	Database on A in service	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, CONNECTED

Only the break in the principal/mirror connection has an effect. The other breaks of principal/witness or mirror/witness do not change

the database mirroring session's behavior.

In summary, what Table HACL1 shows is:

- Only a principal/mirror break, among single-line communication breaks, affects database mirroring. The principal runs exposed, because no log records are sent to the mirror.

Now consider what happens if a second line breaks. The two lines could break simultaneously, or in sequence.

If two line breaks occur simultaneously, the end result will the same as one line breaking followed by the other. However, the exact sequence is unpredictable beforehand; only the subsequent behavior will indicate the sequence to which the simultaneous break will be equivalent.

For our purposes, therefore, we will only consider sequential line breaks. Table 13 shows the basic scenarios, as named in this section, for communication line breaks in the High Availability Mode.

Table 13: Most of the communication line breaks at the two-line level become equivalent to the server-down scenario of one server down.

Scenario	First line break	Scenario	Second line break	Result	Equivalent scenario for remaining servers	See scenario
HACL1.1	*A and W*	HACL1.2	A/W	Server A isolated	Server A down	(none)
		HACL1.3	B/W	Server B isolated	Server B down	HASL2.1
HASL2.1	A/W	HASL2.1	A/B	Server A isolated	Server A down	HASL1.1
		HASL2.2	B/W	Server W isolated	Server W down	HASL3.1
HACL3.1	B/W	HACL3.1	A/W	Server W isolated	Server W down	HASL3.1
		HACL3.2	A/B	Server B isolated	Server B down	HASL2.1

What Table HACL2 shows is that all sequential two-line communication breaks are equivalent to the single-server-down scenarios of the previous section, so we won't repeat any analysis of them here.

What is important to note is that:

- Only one scenario with two communication breaks leads to a failover: the principal/witness line break followed by the principal/mirror line break.

When the principal/mirror line breaks, followed by the principal/witness, no failover results, even though the principal server is isolated, and the mirror and witness cannot see it.

Let's examine scenario HACL1.2 more carefully

Scenario HACL1.2: Principal/mirror link broken followed by principal/witness link broken
If the principal/mirror link is broken, followed by a break in the link between the principal and the witness, then the principal becomes isolated. It cannot see the other servers and loses its quorum. In the meantime, the mirror and witness do not know whether the principal server is still up, so server B takes on the role of principal and an automatic failover occurs. Figure 12 illustrates these events.

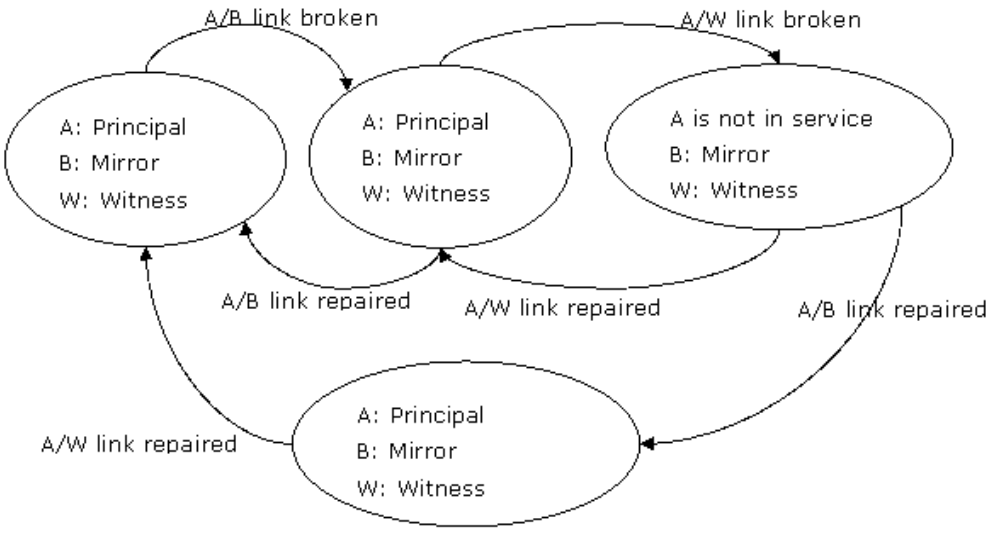

Figure 12: In High Availability mode, when the principal/mirror link is broken and followed by a break in the principal/witness line, no failover occurs

After the principal/mirror link is broken, followed by the principal/witness break, Server A is isolated and takes its database out of service. Server B and W do not form a quorum because Server A may have done work that is not on Server B.

If the principal/witness (A/W) line break is repaired first, Server A will resume its principal role, in a DISCONNECTED state. However, no mirroring will be occurring because there the line between the principal and the mirror is still not repaired.

If the principal/mirror (A/B) line break is repaired first, Server A will resume its mirroring to Server B although without a witness, so the session is exposed. However, no automatic failover is possible until the principal/witness line is finally repaired.

Summary: High Availability Scenarios with Communication Loss: Three Sites
The following table summarizes the behavior of one and two-line breaks with three independent servers. The initial conditions for the table are that the Safety level is FULL and the servers are:

A: Principal, SYNCHRONIZED

B: Mirror, SYNCHRONIZED

W: Witness, CONNECTED

The failover scenario path is highlighted in grey.

Table 14: Summary of one-line and two-line breaks for Safety FULL and three independent servers in the High Availability mode.

First Event	Quorum	Result	Intermediate state	Second Event	Quorum	Result	Ending state	Line Repaired	First repair state
A/B link broken (HACL1.1)	A and W	Database on A in service, exposed	A: Principal, DISCONNECTED B: Mirror, DISCONNECTED W: Witness	A/W link broken (HACL1.2)	B and W	No database in service, A isolated	A: Principal, DISCONNECTED, cannot serve db B: Mirror, DISCONNECTED W: Witness	A/B	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness
				B/W link broken (HACL1.3)	A and W	Database on A in service, exposed B isolated	A: Principal, DISCONNECTED B: Mirror, DISCONNECTED W: Witness	A/B	
A/W link broken (HACL2.1)	A and B	Database on A in service	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness	A/B link broken (HACL2.2)	B and W	Database on B is in service, exposed	A is not in service B: Principal, DISCONNECTED W: Witness	A/W	A: Mirror, DISCONNECTED B: Principal, DISCONNECTED W: Witness
				B/W link broken (HACL2.3)	A and B	Database on A is in service	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, isolated, DISCONNECTED	A/W	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness
B/W link broken (HACL3.1)	A and B	Database on A in service, exposed	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness	A/W link broken	A and B	Database on A is in service	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness, isolated, DISCONNECTED	B/W	A: Principal, SYNCHRONIZED B: Mirror, SYNCHRONIZED W: Witness
				A/B link broken	A and W	Database on A is in service, exposed	A: Principal, DISCONNECTED B: Mirror, DISCONNECTED, isolated W: Witness	B/W	A: Principal, DISCONNECTED B: Mirror, DISCONNECTED W: Witness

Scenario HACL4: Two sites with the witness on the mirror site

When there is only one line of communication between the sets of servers, you must choose where to put the witness server. To start with, assume you put the witness with the mirror database server. There will be just one line of communication between the two sets of servers, and it line may be interrupted, as shown in Figure 13.

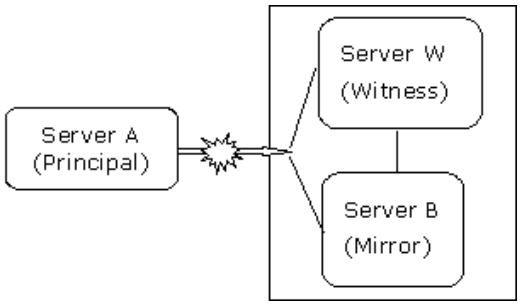

Figure 13: The communication line between the principal and the mirror/witness site is interrupted.

Server A cannot see the witness Server W or the mirror database's Server B, and therefore cannot form a quorum. Server B and Server W can form a quorum, but neither can see the principal on Server A. The result of the line break is illustrated in Figure 14.

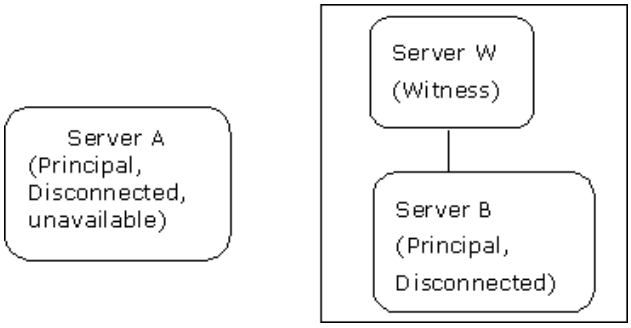

Figure 14: Failover occurs when communication is interrupted and the witness is on the mirror site.

Because Server A cannot see the witness Server W or its former mirror partner Server B, it must enter a disconnected state and make its database unavailable.

Server B and Server W can form a quorum. Server B cannot see Server A, and Server W Server B attempts to become the principal and bring its database online. Because Server W cannot see Server A, it agrees with Server B. Server B now has a quorum, takes on the principal role in this session, and recovers its database.

If you restore the communication line, Server A will see that Server B is a principal, and it will also detect that the witness Server W views Server B as the principal. Server A will change its role to that of mirror, and attempt to synchronize with new principal. When done, the resulting configuration is illustrated in Figure 15.

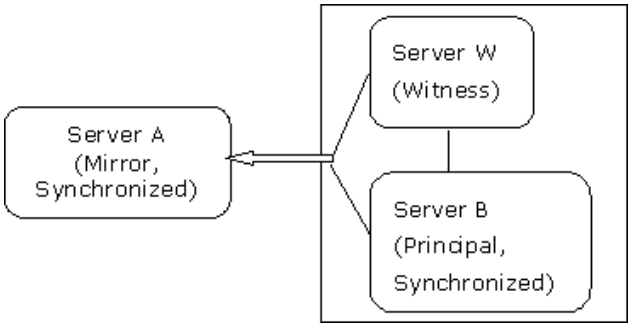

Figure 15: The restored version of this scenario has mirroring in the reverse direction.

To summarize: When the witness resides on a remote site with the mirror, automatic failover will occur if the communication line between sites is interrupted.

Scenario HACL5: Two sites with the witness on the principal site
In this High Availability scenario, assume you put the witness server on the same site as the principal database's server, as shown in Figure 16, and the communication between the two sites is interrupted.

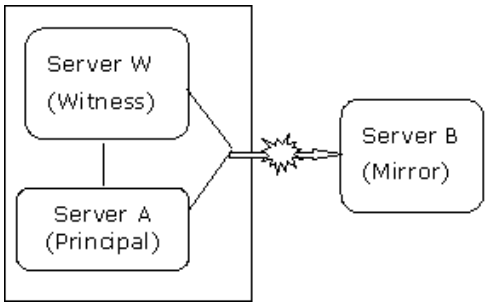

Figure 16: Communication between the principal/witness and mirror is interrupted.

In this case, Server B that houses the mirror database is isolated from the principal and the witness. Server A and Server W continue to form a quorum, so Server A keeps its database as the principal. However, Server A also puts the database into a disconnected state, because it cannot see Server B and no data transfer is possible. Server B also cannot see Server A, so it goes into a

disconnected state as well. The resulting configuration is shown in Figure 17.

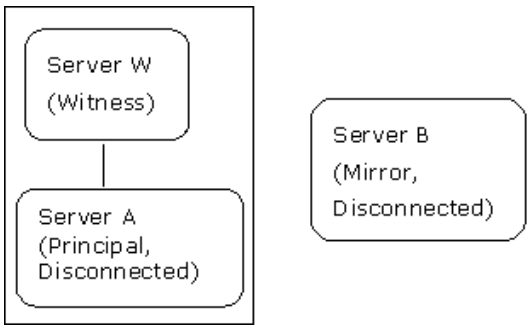

Figure 17: Loss of communication in this scenario results in both partners disconnected.

Server A continues to accept transactions but its transaction log cannot be truncated. If you restore the line quickly, the mirroring session will be able to resynchronize and return to the original operating state.

To summarize: with the witness server on the same site as the principal, and the mirror on a remote site, communication interruption between sites will not cause an automatic failover.

Summary: High Availability Scenarios with Communication Links Broken
In a High Availability configuration with three independent servers, there are three independent lines of communication.

- When a principal/mirror communication loss occurs, no automatic failover will occur.
- When a principal/witness communication loss occurs first, and is then followed by a principal/mirror communication break, automatic failover will occur. Restoring the lines will preserve the reverse direction of the mirroring.
- When a mirror/witness communication loss occurs, no automatic failover will occur.

In a High Availability configuration with only one line of communication, the witness server resides either with the principal or the mirror.

- When the witness resides on a remote site with the mirror, automatic failover will occur if the communication line between sites is interrupted.
- When the witness server is on the same site as the principal, and the mirror on a remote site, communication interruption between sites will not cause an automatic failover.

High Protection Scenarios

The High Protection operating mode works with safety FULL, but with no witness. Because only a server with a principal database and a server with a mirror database are involved, there is only one line of communication between them. This dramatically reduces the number of scenarios.

Case 1. In the High Protection operating mode, only two SQL Server instances are involved, the principal and the mirror. Because there is no witness, automatic failover is not possible. There is only one communication line between the servers, and it can be interrupted, resulting in the configuration illustrated in Figure 18.

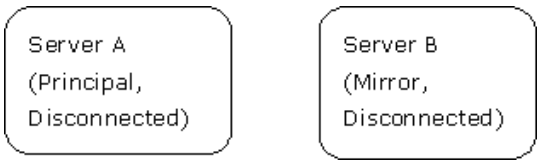

Figure 18. When communication is interrupted in a High Protection scenario, both partners become disconnected but the principal still serves the database.

Because a witness has not been set, the principal does not need a quorum with the mirror, the principal serves the database as normal.

Case 2. If the mirror database becomes unavailable in a High Protection scenario, the principal is the same as in the above scenario, as shown in Figure 19.

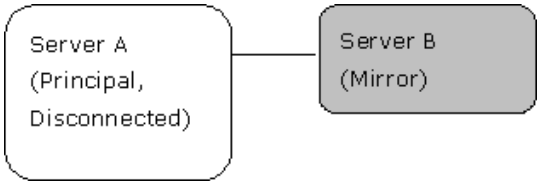

Figure 19: If the mirror server is unavailable in a High Protection scenario, the principal database is unaffected.

Case 3. If the principal database becomes unavailable in a High Protection scenario, the mirror database must remain a mirror, but it will be in a disconnected state, as shown in Figure 20.

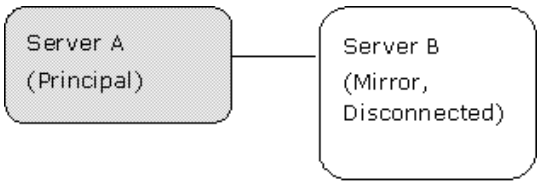

Figure 20: If the principal becomes unavailable in High Protection, the mirror database will enter a disconnected state.

Because the High Protection operating mode does not have a witness set, disruptions do not cause the principal database to become unavailable, and the mirror database remains in a recovering state.

High Performance Scenarios

The High Performance operating mode works with safety OFF. No role is played by a witness server. Only a server with a principal database and a server with a mirror database are involved, and there is only one line of communication between them.

Case 1. In the High Performance operating mode, two SQL Server instances are involved. One houses the principal database and the other contains the mirror database. Therefore there is only one communication line between the servers, and it can be interrupted, resulting in the configuration illustrated in Figure 21.

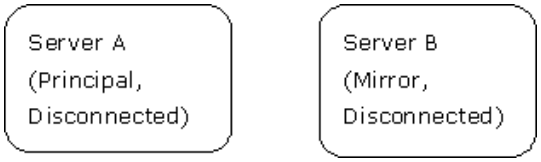

Figure 21: When communication is interrupted in a High Performance scenario, both partners become disconnected but the principal database remains available.

Because no witness has been set, Server A does not require a quorum to keep its database available. Therefore the principal, though disconnected, continues to accept user activity. If you restore communication, the mirror database will attempt to catch up but may not be able to, or may be subject to redo errors if it cannot retrieve all the missing transactions.

Case 2. If the mirror database becomes unavailable in a High Performance scenario, the result for the principal is shown in Figure 22.

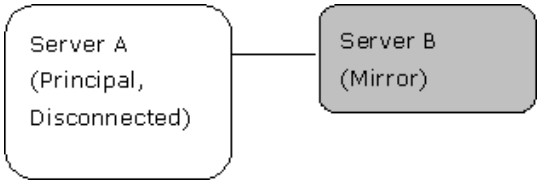

Figure 22: If the mirror server is unavailable in a High Performance scenario, the principal database is unaffected.

The principal database remains available because safety is OFF.

Case 3. If the principal database becomes unavailable in a High Protection scenario, the mirror database remains a mirror, but it will be disconnected, as shown above in Figure 23.

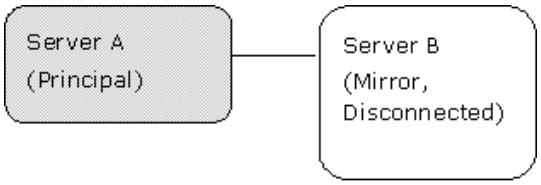

Figure 23: If the principal becomes unavailable, Server B is unaffected but enters a disconnected state.

In the High Performance operating mode, just as in High Protection, no automatic failover is possible. Because no witness has been set, the principal server keeps its database available when the mirror becomes unavailable. Also because safety is OFF, transactions are not guaranteed to reach the mirror. Some transactions may be lost if you force a failover to the mirror.

It is not recommended to run with the safety OFF and a witness set. The witness cannot provide automatic failover, because the safety is not FULL. Furthermore, if the witness and the mirror are unavailable, then the principal has lost quorum and brings its copy of the database offline.

Top of page

Implementing Database Mirroring

Basic information about implementing database mirroring can be found in the SQL Server 2005 Books Online, under "How To" topics for database mirroring. In this section of this white paper, we'll look at a particular example of a database mirroring implementation, along with some best practices.

Monitoring Database Mirroring

You can detect the state of each database mirroring partner by examining the principal or mirror databases in SQL Server 2005 Management Studio's Object Explorer. If the principal is synchronized with the mirror, Object Explorer will append a (Principal, Synchronized) message next to the principal database name, and (Mirror, Synchronized) next to the mirror server's name. You can also inspect the status of a database mirroring session by observing the Status box of the Mirroring page of the Database Properties dialog when invoked from the principal server. (The Database Properties dialog will not open on the mirror database.)

You can also query the database mirroring catalog views, sys.database_mirroring and sys.database_mirroring_witnesses. (For more information about using the catalog views to inspect the database states in a mirroring session, see "Database Mirroring Catalog View Metadata" in the Database Dynamics section previously in this paper. The catalog views are also fully documented in SQL Server Books Online.)

Database Mirroring Perfmon Counters

You can use Perfmon counters to monitor traffic between the partners in a database mirroring session. The SQL Server Database Mirroring object contains a number of useful Perfmon counters for the principal and witness servers. (See "Monitoring Database Mirroring" in SQL Server Books Online.)

Each counter in the Database Mirroring object can be set per database. If you are mirroring more than one database on a server, you can measure the activity of each database individually as well as the total mirroring activity across all the databases.

For the principal server, the *Log Bytes Sent/sec* counter will tell you the rate at which the principal is sending transaction log data to the mirror, while the *Log Send Queue* shows you how many bytes in the transaction log buffer are yet to be sent to the mirror at any given point in time. As transaction log data is sent from the principal to the mirror, the principal's Send Queue will be depleted, but it will also grow as new log records are entered into the log buffer. The *Transaction Delay* counter on the principal will show the delay the principal is experiencing due to waits for acknowledgment of log records received by the mirror database. The *Pages Sent/sec* relates to the principal sending data pages to the mirror to assist in synchronization.

On the mirror server, the *Log Bytes Received/sec* shows how well the mirror is keeping up with the principal (see the Log Bytes Sent/sec counter above). The *Redo Queue* counter shows the size of the redo queue that the mirror uses to replay transactions from the principal during its ongoing redo phase. The *Redo Bytes/sec* shows you the rate at which the mirror is able to replay those transactions from its redo queue.

For each partner, the *Sends/sec* and *Receives/sec* counters show the number of discrete send and receive actions, giving gives you a sense of the rate at which the servers are communicating. The *Bytes Sent/sec* and *Bytes Received/sec* counters show the total number of bytes sent and received on each partner server for those sends and receives.

Estimating Redo and Catch-up Time

You can use values from the Redo Queue and Redo Bytes/sec to estimate the time it should take the mirror database to finish a redo and become available, should a failover occur. The estimate consists of the following simple formula:

Estimated time to redo (in seconds) =
(Redo Queue)/(Redo Bytes/sec)

Similarly, you can use the Log Send Queue and Log Bytes Received/sec counters to estimate the time it should take the mirror to catch up with the principal, if activity on the principal has gotten ahead and has since quieted down. The estimate is given by the following formula:

Estimated time to catch up (in seconds) =
(Log Send Queue)/(Log Bytes Received /sec)

Profiler Events
SQL Server 2005 Profiler contains one event class for database mirroring. The Database:Database Mirroring State Change event will record whether the server being monitored undergoes a state change. (See the topic "Database Mirroring State Change Event Class" in SQL Server Books Online.) It is helpful to include the Database Name and the State columns when using this event class. You can use this event to alert you to any state change in the database mirroring session.

Troubleshooting Database Mirroring

Two areas where errors are most likely to occur are during setup and during run-time.

Troubleshooting Setup
If you've installed database mirroring but it won't start, retrace the steps that you took during setup.

1. Make sure that the mirror server has caught up close enough in time to the principal. If you see the following message when you attempt to start mirroring,

   The remote copy of database "AdventureWorks" has not been rolled forward to a point in time that is encompassed in the local copy of the database log. (Microsoft SQL Server, Error: 1412)

   you know that the mirror is not caught up. You need to apply transaction log backups from the principal to the mirror (with NORECOVERY) in order to catch the mirror up to a point where it can start receiving log records from the principal.

2. Make sure that the SQL Server Windows service accounts on each server are trusted on each other server. If the servers are on non-trusted domains, make sure the certificates are correct.

3. Make sure that the endpoints are not just defined, but also started, by querying the sys.database_mirroring_endpoints catalog view:

   ```
   SELECT * FROM sys.database_mirroring_endpoints;
   ```

   Also make sure that the fully qualified computer names are correct, and the port numbers are correct. If you are mirroring among instances that are on a single physical server, the port numbers must be unique. Each SQL Server service login should have CONNECT permission to the endpoint.

   Finally, make sure that each endpoint role has been defined properly for the server as you intended their roles.

4. Make sure you have identified the correct partner names in the ALTER DATABASE commands. You can inspect the partner names in the sys.database_mirroring catalog view on the principal and mirror (and sys.database_mirroring_witnesses witness if in High Availability mode).

Troubleshooting Runtime Errors
If database mirroring has been set up correctly, and running, and then an error occurs, check the current status of the session. If was put in a SUSPENDED state due to the error, redo errors may have occurred on the mirror. Check to make sure there is sufficient disk space on the mirror for both redo (free space on the data drives) and for log hardening (free space on the log drive). When you are ready to restart the session, use ALTER DATABASE to RESUME the session.

If you cannot connect to the principal database, it is most likely because safety is FULL and the principal server cannot form a quorum. This can happen, for example, if your system is in High Protection mode (safety is FULL, but there is no witness), and the mirror has become disconnected from the old principal. You force the mirror server to recover, using the following command on the mirror:

```
ALTER DATABASE [AdventureWorks] SET PARTNER FORCE_SERVICE_ALLOW_DATA_LOSS
```

The problem is that once you've recovered the mirror database, it becomes a principal server but cannot form a quorum, and so it cannot serve the database. In that case, just set safety OFF to allow it to serve the database.

Safety vs. Performance

Database mirroring performance is a function of the type of activity and the transactional safety setting.

The performance of the principal server is affected by the transfer of log records to the mirror. The overhead that database mirroring places on a principal server is a function of the type of activity. Database mirroring will perform best with many long transactions by many users, because the normal transactional activity of the database server will mask the overhead of transferring log records to the mirror. When a single user is issuing a large number of small transactions in sequence, the overhead of database mirroring for each transaction will become more prominent.

Performance of the principal server will also be affected by the Safety setting. When safety is FULL, the principal must wait for an acknowledgment by the mirror that it has received log records before it can issue a transaction commit message back to the client. With many users and long transactions, that overhead will be inconsequential. Systems with single threads and many small transactions may operate better with safety OFF.

Because the mirror server is continuously replaying data modification transactions that it receives from the principal, *the data cache on the mirror server will be 'hot'*. In other words, the data cache will be populated with data and index pages based on the same kinds of changes made on the principal. To make the mirror cache even more like the principal's cache, *database mirroring also passes SELECT hints to the mirror* so that the cache used for querying data also is reproduced on the mirror server. This will aid in making the mirror more like the principal and will reduce the remaining redo time in the case of a failover. Obviously, any additional activity on the mirror server, including queries against database snapshots, will affect the state of the cache and could increase the duration of time to finish the redo phase in the event of a failover.

Testing Database Mirroring

When setting up your own systems to test database mirroring, you have a number of options available. All mirroring requires is that the servers in the database mirroring session must be distinct SQL Server instances. Therefore you can learn about and test database mirroring on a single physical server, if you install multiple instances of SQL Server 2005's relational engine. You can also test multiple instances in a single virtual server, but your tests will be more reliable if done on a physical server.

When testing database mirroring for load or stress purposes, you will need distinct physical servers. Two or three instances on a single server may consume an unrealistic share of the physical server's resources. Equally important is good connectivity between servers. The better the network connection between the principal and the mirror, the better the transfer rates for log records and messages will be.

The most realistic testing will be on the actual target servers or on a test bed that has the same physical properties as your final system. When you test with multiple instances on a single server, you can only simulate the effects of server loss on database mirroring, by stopping one of the instances or shutting down a machine. With multiple physical servers, you can also test the loss of connectivity by disconnecting a network cable.

The following practices may assist you in creating test conditions:

- To test a server failure, just shut down the SQL Server instance, either through SQL Configuration Manager or using SHUTDOWN WITH NOWAIT.
- To test a communication failure, remove a network cable from the server.
- To test a database failure, stop the SQL Server service and rename the underlying .mdf file, and then restart the SQL Server.
- To cause a redo error on the mirror, add a file to the principal database on a driver volume that does not exist on the mirror server.
- Another way to cause a redo error on the mirror is to force the mirror server data file to run out of disk space.
- To force a database shutdown on the principal, force the principal's data file to run out disk space.
- To cause a log buffer hardening to fail on the principal or mirror, force the log file to run out of disk space.

Preparing the Mirror server for Failover

Database mirroring is strictly a database-to-database relationship. Only database data is sent using the log records from the principal to the mirror. Just as with log shipping and replication, you must prepare the standby server with the mirror database to fully take over in the event of a failover. There are several levels to consider when preparing the mirror server.

At the physical server level, you should make sure that the standby server has the same physical CPU and memory configuration, or as close as possible, or else the standby will under perform after a failover. You may also have supporting applications, monitors, or other executables that support the database applications and must be configured on the mirror server

At the SQL Server level, you should also ensure that the standby has the same SQL Server configurations (for example, AWE, max degree of parallelism). But most fundamental will be the logins and their permissions. All active SQL Server logins on the principal server must also be present on the mirror server or the application will not be able to use it as the new principal server in the event of a failover. SQL Server Integration Services has a Transfer Logins task that you can use to copy logins and passwords from one server to another, but you may still need to set database permissions for those logins. If you transfer logins to a server in a different domain, the SIDs may not match and you will need to match them.

There are numerous support objects that may exist on the principal SQL Server that you will need to migrate to the standby server: SQL Agent jobs and alerts, SQL Server Integration Services packages, support databases, linked server definitions, backup devices, maintenance plans, SQL Mail or Database Mail settings, and possibly Distributed Transaction Coordinator settings, to name a few.

When SQL Agent jobs are transferred to the standby server, most will have to remain disabled. In the event of a failover, you will need to enable those jobs.

After a failover, if your application is using SQL Server authentication, you will need to resolve the logins on the new principal SQL Server with the users on the new principal database. The best tool for this is the stored procedure sp_change_users_login.

Multi-Database Issues

Many applications make use of multiple databases on a single server. One application may reference multiple databases, or perhaps many applications all make use of several databases. However, database mirroring works with a single database at a time. You need to take this into account when designing mirroring into your database architecture.

If you desire the High Availability mode, your best fit will be when one application matches up with one database. Then if an automatic failover occurs, the application no longer requires any databases on the principal server. Consider what might happen if you have multiple databases on a single server and operate in the High Availability mode. If there is a physical server outage, a SQL Server instance failure, or a communication failure, all the databases would automatically fail over to the standby server, and their mirrors would then become new principal databases. If the witness is visible, the application could connect to the new principal databases. But what would happen if one of the databases incurred a torn page from a disk fault, so that only the one database failed over? In that case, it might be impossible to get the application to connect to all the right databases.

Therefore applications that rely on multiple databases will not be good candidates for the High Availability mode of database mirroring. You may be able to use safety OFF, with the realization that you will not have automatic failover, but you will have a high-performance method of keeping another database server in sync.

Top of page

Database Mirroring and High Availability Technologies

SQL Server 2005 now has at least four high availability technologies, and while there is some overlap, each technology has its own relative advantages and disadvantages. Those technologies are

- Database Mirroring - For this discussion, we will be considering the High Availability operating mode with safety FULL and a witness.
- Failover Clustering - The most typical configuration is a two-node Windows failover cluster with one SQL Server instance.
- Log Shipping - Assume SQL Server built-in log shipping with a separate monitoring.
- Transactional Replication - Consider a configuration with a separate distribution server and one subscriber which serves as the standby server if the publisher server fails.

In this section we'll compare the basic features of those four technologies, and drill down into areas where database mirroring may complement or prove a better solution.

The following table shows a number of availability features for all four technologies.

Table 15: Comparing SQL Server 2005 High Availability technologies.

Category	Availability Feature	Database Mirroring (HA Mode)	Failover Clustering	Log Shipping	Transactional Replication
Failover characteristics	Standby Type	Hot	Hot	Warm	Hot
	Automatic role change	Yes	Yes	Custom coding required	Custom coding required
	Failover preserves committed work	Yes	Yes	No	No
	Failover type	Automatic and Manual	Automatic and manual		
	Database downtime during failover	Less than 10 seconds	30 seconds + database recovery	Variable	Variable
Physical configuration	Redundant storage locations	Yes	No (shared disk)	Yes	Yes
	Hardware requirements	Standard servers	Cluster Certified Servers and Storage	Standard servers	Standard servers
	Physical distance limit	None	100 Miles	None	None
	Additional server role	Witness	None	Monitor	Distributor
Management	Complexity level	Low	High	Low	Medium
	Standby Accessible	Via database snapshots; possible performance impact	No	R/O but incompatible with restores	Yes for read only work
	Multiple Secondaries	No	No	Yes	Yes
	Load delay on secondary	No	No	Yes	No
	Scope of availability	Database	Server instance	Database	Database
Client Access	Client redirect	Support in ADO.NET and SQL Native Client	None required, Virtual IP	Custom coding required	Custom coding required

The above table summarizes many of the characteristics of all four high availability technologies. The next sections make some more detailed comparisons.

Database Mirroring and Clustering

The most important contrast between database mirroring and failover clustering is the level at which each provides its redundancy. Database mirroring provides protection at the database level, whereas clustering provides protection at the server instance level. Another important difference is that in database mirroring, the principal and mirror servers are separate SQL Server instances with distinct names, whereas a SQL Server instance on a cluster gets one virtual server name and IP address that remains the same no matter what node of the cluster is hosting the instance.

If you need database protection at the server level (for example, your application requires access to many databases on the same database server simultaneously), failover clustering may be a more appropriate choice. However, if you are concerned to provide availability for one database at a time, database mirroring has a number of advantages.

Unlike clustering, database mirroring does not require proprietary hardware and does not have a potential failure point with shared storage. Database mirroring brings the standby database into service much faster than any other high availability technology, and works well with new capabilities in ADO.NET and SQL Native Access Client for client-side failover.

You cannot use database mirroring within a cluster, but you may consider using database mirroring as a method for creating a hot standby for a cluster instance database. If you do, be forewarned that because a cluster failover is longer than the timeout value on database mirroring, a High Availability mode mirroring session will react to a cluster failover as a failure of the principal server. It would then put the cluster node into a mirroring state.

Database Mirroring and Transactional Replication

Database mirroring and transactional replication are both based upon reading the transaction log of an originating server, but after that their technologies diverge considerably. (For more details about transactional replication, see the associated topics in SQL Server Books Online.) Transactional replication is often used for high availability because it can deliver user transactions from a publisher database to a subscriber in matters of seconds. Database mirroring has the advantage that it is as fast as or faster than replication, but delivers all of a database's transactions, not just those related to user tables.

Transactional replication is an appropriate technology for scaling out data to multiple subscribers for reporting. Transactional replication subscriber databases are normally considered read-only anyway, so they are ideal candidates when access to near-real-time data is required.

Database mirroring is compatible with transactional replication, and is most useful as method of keeping a hot standby of a publisher database. Other methods of protecting a replication publisher, such as log shipping, cannot keep a standby server for the publisher ahead of the publisher's own subscribers. In other words, transactional replication can deliver transactions to its subscribers much faster than a transaction log backup scheme. Because database mirroring is so fast, it is much more suitable for keeping a hot standby of a publisher database.

If the publisher should fail, however, you will have to manually re-establish the recovered standby database as the publisher, and reconnect it to the distribution server, just as you must currently do if using log shipping to maintain a publisher server standby.

Database Mirroring and Log Shipping

Database mirroring and log shipping both rely on the restore and recovery capabilities of SQL Server databases. A database mirroring mirror database is in a constantly recovering state, more or less continuously replaying transactions from the principal. A log shipping secondary replays transactions applied to it periodically from transaction log backups. Because bulk-logged data is appended to a transaction log backup, log shipping can work in the bulk-logged recovery model. Database mirroring, on the other hand, transfers log records directly from the principal to the mirror and cannot deliver bulk-logged data.

In many cases database mirroring can provide the same kind of data redundancy as log shipping with higher availability and automatic failover. However, if your application relies on multiple databases on one server, log shipping may be an equally valid approach (see "Multi-Database Considerations" in the previous section).

Additionally, there are database mirroring scenarios where log shipping may supplement availability. For example, you could have a High Availability database mirroring configuration in-house, and log ship the principal server to a remote site for disaster recovery purposes. Figure 24 illustrates how such a configuration might take place.

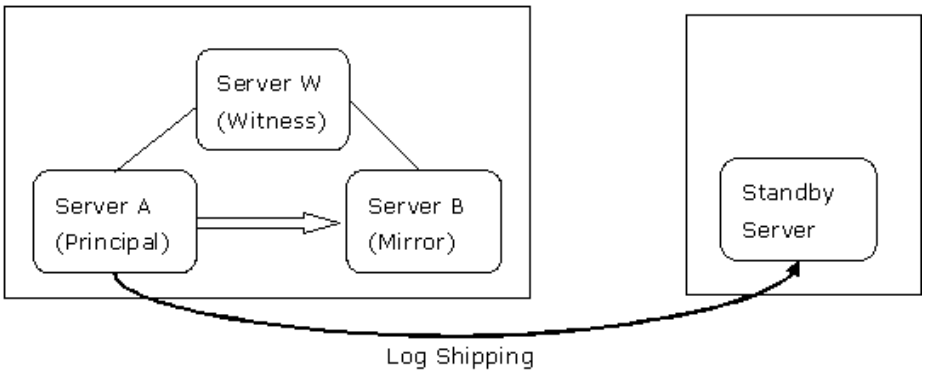

Figure 24: You may log ship a principal database to a remote location.

The advantage here is that in the event of a loss of the entire site, the data is available on the secondary site. However, in the event of a database mirroring failover, log shipping from Server B to the remote standby will normally have to be reinitialized.

Another scenario for using log shipping to complement database mirroring would be as a local standby for the principal server where the database mirroring session is being used for disaster recovery. In this case, the mirroring session is in the High Performance mode, with the mirror on a remote site as the remote standby.

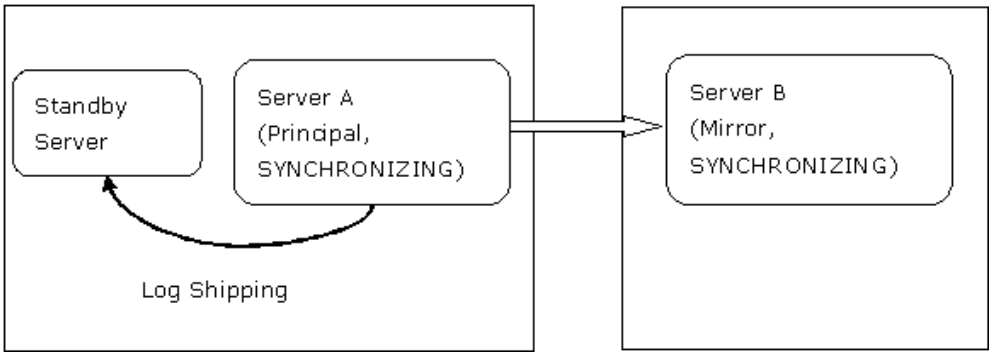

Figure 25: You can log ship a principal database as a method of preserving all transactions.

In the High Performance mode, there is a potential for data loss if the principal fails and the mirror is recovered using a forced service recovery. If you are log shipping the old principal, and if the transaction log file of the old principal is undamaged, you can make a 'tail of the log' backup of the principal to get the last set of log records from the transaction log. If the standby log shipping database has had every other transaction log backup applied to it, you can then apply the tail of the log backup to the standby server and not lose any of the old principal's data. You can then compare the data in the log shipping standby server with the remote database and potentially copy missing data to the remote server.

In any case, comparing log shipping to database mirroring should make it clear that *it is important to keep database and transaction log backups of the principal database.* Applying these log backups to a log shipping server can supplement your database mirroring configuration.

Top of page

Conclusion

Database mirroring is a new SQL Server 2005 technology that can deliver high availability and high performance solutions for database redundancy. In database mirroring, transaction log records are sent directly from a principal to a mirror database whenever the principal's transaction log buffer is written to disk (hardened). This technique can keep the mirror database nearly up to date with the principal, and with no loss of committed data. In the High Availability operating mode, if the principal fails, the mirror server will automatically become a new principal and recover its database. Using the new ADO.NET or SQL Native Access Client drivers, applications can also perform an automatic failover from the client servers as well. Database mirroring becomes an important new option in the array of high availability technologies supported by SQL Server 2005.

For More Information

Database Mirroring FAQ http://www.microsoft.com/technet/prodtechnol/sql/2005/dbmirfaq.mspx

SQL Server TechNet site: http://www.microsoft.com/technet/prodtechnol/sql

SQL Server Developer Center: http://msdn.microsoft.com/sql

Microsoft SQL Server site: http://www.microsoft.com/sql/

Top of page

Manage Your Profile