

SQLFlex Mirroring Support

Using SQL Server Mirroring in DataFlex applications

Contents

- Overview
- How does it work?
- SQL Server Setup
- Using SQLFlex Mirroring Support

Overview

Database mirroring is an SQL Server 2005 technology 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.

How Does it Work?

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.

To implement mirroring, you will need at least two servers, or three with an optional setup in these roles: witness, principal and mirror.

An illustration on how mirroring works:

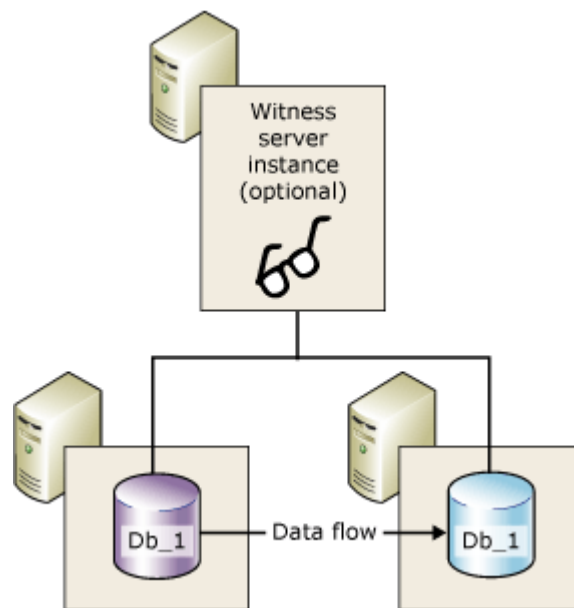


Figure 1

Server Roles

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.

Principal

The principal is the originating database and server, the one that send all the transaction log to the mirror the receiving database and server.

Mirror

Is receiving database and server of the *principal* server role, it acts as a backup of the *principal* data and logs. The mirror database is not available from a sql client and can only be used in a mirroring setup.

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.

Operational Modes

There are three operational modes in SQL Server mirroring:

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.

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.

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.

SQL Server Setup

We will setup a mirroring environment in two server instances a High Protection Operating Mode setup.

First take a look at these considerations.

- 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.

Requirements

You need SQL Server 2005 SP1 or above to use Mirroring. This is the supported databases table:

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	✓	✓			

Figure 2

i In this document we use two SQL server instances, MACHINE\MAIN and MACHINE\MIRROR both with SQL Server 2005 SP1(not express edition).

Setup Steps

This is the SQL setup we use here.

Database name: main(must be the same name in both roles)
Principal sever name: MACHINE\main
Mirror server name: MACHINE\mirror

Start Transaction Log in Both Servers

The principal and mirror server must have *transaction log* activated, you active it changing the startup command in the **SQL Server Configuration Manager**

1. Go to Start Menu\Programs\Microsoft SQL Server 2005\Configuration Tools\SQL Server Configuration Manager
2. Select SQL Server 2005 Services
3. Right click the service you want to configure, MACHINE\MAIN and then MACHINE\MIRROR
4. Select the Properties option
5. Select the Advanced tab
6. Add this to Startup Parameters at the end of the value 2005 does not have the SP1 or earlier applied yet (do this for both servers):

```
;-T 1400
```

Setup FULL Recovery in Principal

Check if principal has FULL recovery mode enabled:

```
select    recovery_model_desc, *
from sys.databases
where name = 'main'
```

If not, set FULL recovery in principal:

```
USE master;
GO
ALTER DATABASE main
SET RECOVERY FULL;
GO
```

Configuring Endpoints

Endpoints are port listeners where the mirror and principal are going to establish communication.

Principal Endpoint:

```
USE main;
GO
CREATE ENDPOINT MirroringEndPoint_Principal
    STATE=STARTED
AS TCP (LISTENER_PORT=10111)
FOR DATABASE_MIRRORING (ROLE=PARTNER) -- Enabled as
Partner only
GO
```

Mirror Endpoint:

```
USE main;
GO

CREATE ENDPOINT MirroringEndPoint_Mirror
    STATE=STARTED
AS TCP (LISTENER_PORT=10112)
FOR DATABASE_MIRRORING (ROLE=ALL) -- enabled as Witness or
Partner
GO
```

To inspect your created endpoints:

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

Backup Principal and Restore it in Mirror

i Important: In both, principal and mirror, the backup(log and data) should be located in the same exact location(e.g. C:\backup\data.bak C:\backup\log.bak).

Backup *principal's* main database:

```
USE main
GO
BACKUP DATABASE main
TO DISK = 'C:\main_Data.bak'
WITH FORMAT
GO
BACKUP LOG main
```

```
TO DISK = 'C:\main_Log.bak'  
WITH FORMAT
```

Or you can use SQL manager to do this:

1. Right click the database, select **Tasks**
2. Backups
3. Add a location
4. Apply

Restore main database in the mirror:

```
RESTORE DATABASE main  
FROM DISK='C:/main_data.bak' WITH REPLACE,NORECOVERY  
Restore the log too:  
RESTORE LOG main  
FROM DISK='C:/main_Log.BAK' WITH NORECOVERY
```

You can *restore* this backup files in *SQL Manager* just do:

1. Right click the database, select **Restore**
2. Add the Backup: restore the data first and then restore the log
3. Add the location of the backup
4. Go to Options and select **NORECOVERY**

From now, that **database will be not available to open or make changes**, now it is only a mirror database and it is only usable to do mirroring. Even though we still need to run some commands to make the mirror environment work.

Setup Mirroring Partnerships

You can see partnership as a contract between the *principal* server and the *mirror* database where both agree to send their logs, so both databases gets synchronized and have the data.

Setting a partner(*principal*) to *mirror*:

```
ALTER DATABASE main  
SET PARTNER =  
'TCP://127.0.0.1:10111'  
GO
```

Setting a partner (*mirror*) to *principal*:

```
ALTER DATABASE main  
SET PARTNER = 'TCP://127.0.0.1:10112'  
GO
```

General Inspection

```
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;
```

Setting Failover in Principal

```
ALTER DATABASE main
SET PARTNER FAILOVER
GO
```

Screenshot of SQL Manager 2005:

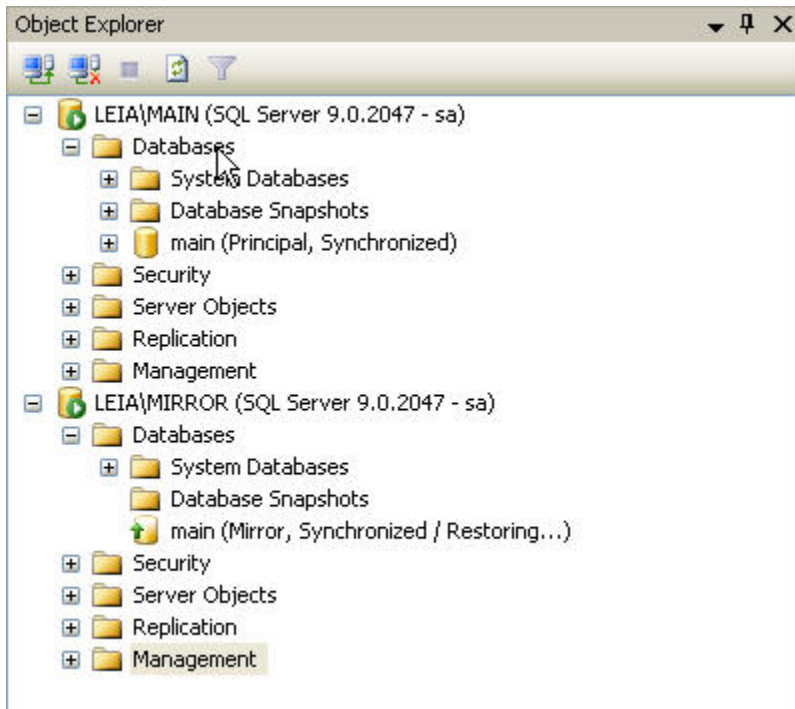


Figure 3

Using SQLflex Mirroring Support

To add support for SQL Server Mirroring we added a new command:

```
SET_MIRROR_SERVER to "MACHINE\MIRROR" "database"
```

Dataflex 3.2

To make Dataflex work with Mirroring and Mertech's driver, you can create a program that login set the mirror and execute your application:

loginchain.src:

```
use mertech.inc

string p1 p2

Load_Driver "SQL_DRV"

// Set the Mirror Server
SET_MIRROR_SERVER to "MACHINE\MIRROR" "main"
If (Err) showln "Error setting failover Server"

// Login into the main server
login "MACHINE\MAIN" "user" "password" "sql_drv"

cmdline p1
cmdline p2

chain (p1 + " " + p2)
```

Compile it and run it:

```
dfcomp loginchain.src
dfrun logchain your_application_executable
```

To test it you can shutdown the MAIN server and try to start your application, your application should start using the main server in a transparent way. And you can see it in the driver trace that your application login from the MIRROR and logout from the MAIN server.

Visual Dataflex

SQLFlex automatic login dialog also has an option to set your mirror server and database:

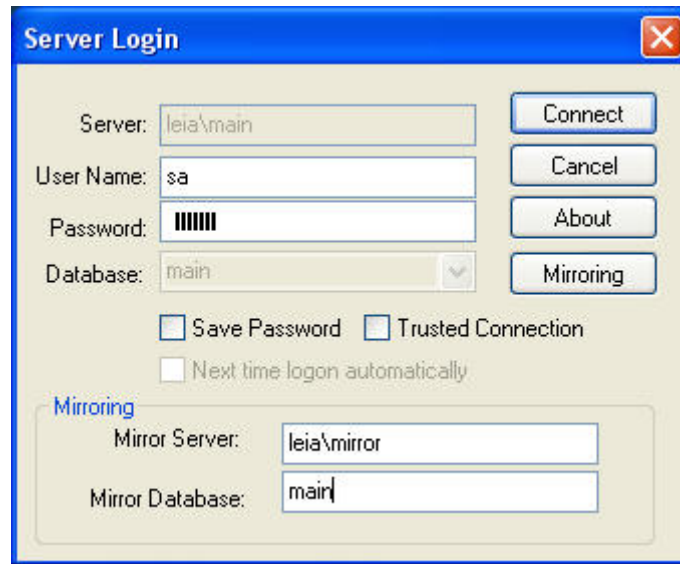


Figure 4

With VDF besides of automatic login dialog, you can still use the same command as in Dataflex 3.2 modifying the source code of your application. If you have a personalized login dialog, using the command `SET_MIRROR_SERVER` is the best option.

After that, your Visual Dataflex application will be running in a mirroring environment. To test this, you can shutdown the main server, while continuing running your application:

You can make some changes, remember, the *principal* server is down:

Customer Number: 1
Name: Mertech Data

Address Balances Comments

Street Address: 1400 NW 190 Ave
City/State/Zip: Miami FL 33176-6017
Phone Number: 305-555-0012
Fax Number: 305-555-0017
E-Mail Address: customerservice@amiles.com

Figure 7

Put the server up again, and look in SQL manager the changes you just did:

Microsoft SQL Server Management Studio Express

Object Explorer

Table - dbo.CUSTOMER

RECNUM	CUSTOMER_NU...	NAME	ADDRESS	CITY	STA1
1	1	Mertech Data	1400 NW 190 Ave	Miami	FL
2	2	American Produc...	12314 Portland ...	Redmond	WA
3	3	Ortega Emergen...	56789 SW 199th...	Dallas	TX
4	4	Ace Manufactur...	123 NorthEast I...	Peoria	IL
5	5	True Value Mach...	7654 City Cente...	Warwick	RI
6	6	Big River Public ...	P.O. Box 331	Salem	OR
7	7	General Food Su...	4444 Park Ave.	Tucson	AZ
8	8	Wilson Bolt & Nu...	P.O. BOX 456	Atlanta	GA
9	9	Rectangle Servic...	Route 12, Box 4...	Columbia	MD
10	10	Youth for Brass ...	P.O. Box 789	Denver	CO
11	11	America West H...	P.O. Drawer 3333	San Diego	CA
12	12	Stunning Softwa...	9876 N. Dakota St.	Fresno	CA
13	13	3A Software	1600 Technolog...	La Mesa	CA
14	14	Zorro Cutlery	12121 SE 12th S...	San Diego	CA
15	15	Automotive Part...	7890 NW 64th	Miami	FL

Figure 8

Conclusion

Having a Mirroring setup is very big advantage over not having it, you can ensure that your application will be up 24/7 and the process is transparent so the user cannot notice that the database change. Your users data is very important and if your application uptime is 100% of the day it will make sure that you are not losing any data at any point in time.