SQL Server setup is always available

Database Mirroring solution helps build a high-availability database management system in SQL Server which is quite simple and suitable for medium-sized and lower-level databases.

Database Mirroring solution helps build a high-availability database management system in SQL Server which is quite simple and suitable for medium-sized and lower-level databases.

The requirement of a highly available database management system is becoming increasingly urgent, sometimes a vital factor for organizations and companies. However, to achieve high availability (almost always active) is not a simple thing, because there are always many factors that affect system performance: hardware problems, network infrastructure, errors Operating system, application software error, virus . The article introduces a solution to help achieve high availability (HA - High Availability) on the currently used database management system: **SQL Server** .

HA solutions on SQL Server

Failover cluster

This solution uses a shared hard drive - usually a **SAN** to hold the database. There are many ' **instances** ' of SQL Server installed, each instance is a node, but at one time only one node has control of the database. When this node has a problem, another node will replace it to manage the database.

Log shipping

Additional structure of a database as a mirror (copy). When there is a change or update from the main database, the log file that records these changes will be sent to the mirror server instance. In this way, an updated copy of the database is maintained. In case of an incident, the copy database will be converted into the main database in a short time.

Tính năng Database Mirroring	Phiên bản Enterprise	Phiên bản Developer	Phiên bản Standard	Phiên bản Workgroup	SQL Express
Đối tác	•	•	•		
Giám sát	•	•	•	•	•
An toàn = Có	•	•	•		
An toàn = Tắt	•	•			
Sẵn sàng trong suốt quá trình gỡ bỏ sau khi failover	•	•	•		
Khôi phục song song	•	•			
Sao lưu dữ liệu nhanh chóng	•	•			

Replication

If **Failover cluster** and **Log Shipping** are two solutions to ensure **high-availability** at database level, **Replication** only ensures high-availability at the level of objects in the database such as table, view . These objects will be copied to one 2nd instance of SQL Server for storage.

Data Mirroring in SQL Server

Database Mirroring (DM) is the new solution for building highly available databases in SQL Server. DM overcomes the disadvantages of previous solutions such as:

• *Compared to the Failover Cluster*, DM does not require special hardware like SANs, thus reducing the cost of configuration

• *Compared to Log Shipping*, DM can automatically switch to a mirror machine when an error occurs without the administrator having to act. Log shipping requires manual configuration with T-SQL. Therefore, DM is called 'hot standby', when the interruption time (downtime) can be calculated in seconds, and Log-shipping is called 'warm standby', because the interruption time can be in minutes or than.

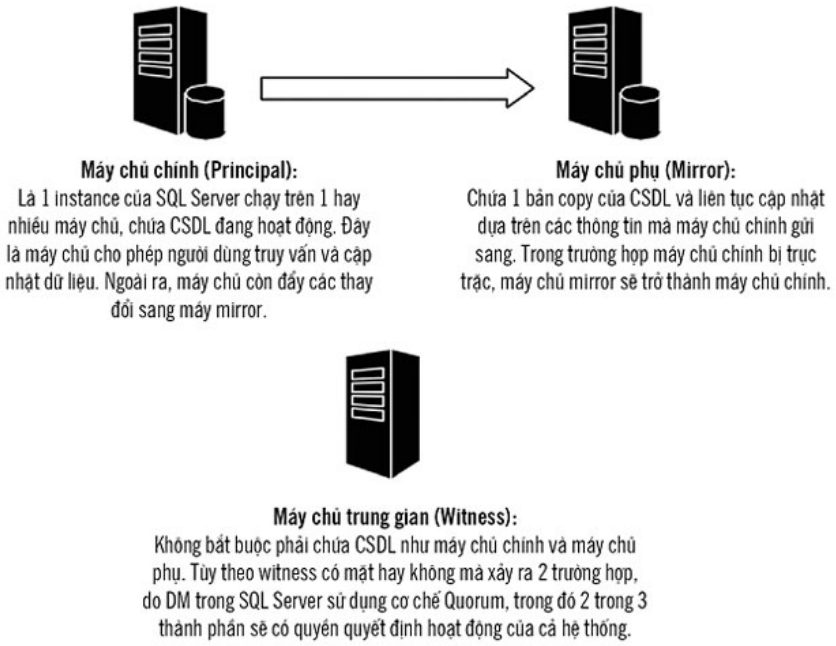
• *Compared to Replication* , DM is superior because it protects the entire database, and Replication only protects parts of the database, for example tables like master.

However, DM is only available in **Enterprise / Developer** version of **SQL Server 2005 SP1 / 2008.**

1. Structure of DM in SQL Server

DM in SQL Server requires 3 main instance: 1 instance (principal role) to manage the database, 1 extra instance (mirror) to ensure database backup. An instance of a witness connects with two main and sub-instance instances to monitor and ensure the availability of the database.

When there is a witness face: The witness server connects to both the main server and the mirror server. Now the whole system becomes a quorum that 2 of the 3 components have the right to decide. In case the main server has a problem, the witness server will automatically switch the mirror server to the main server. If then, the main server is working again, the main server will assume the role of a mirror server (2 servers now change roles for each other) until the administrator's intervention (*diagram 1*).



When there is no witness server: The automatic conversion process will not work without the administrator's impact.

In SQL Server there is the concept of ' **endPoint** ' that can be interpreted as 'connection point', allowing SQL Server instances to communicate with each other via TCP (*diagram 2*).

Each endpoint is identified by a corresponding address and port. Theoretically, the address must be the full domain address, but in fact can be used in one of four ways:

- Determine via server name. Example: TCP: // PRINCIPAL: 7024.

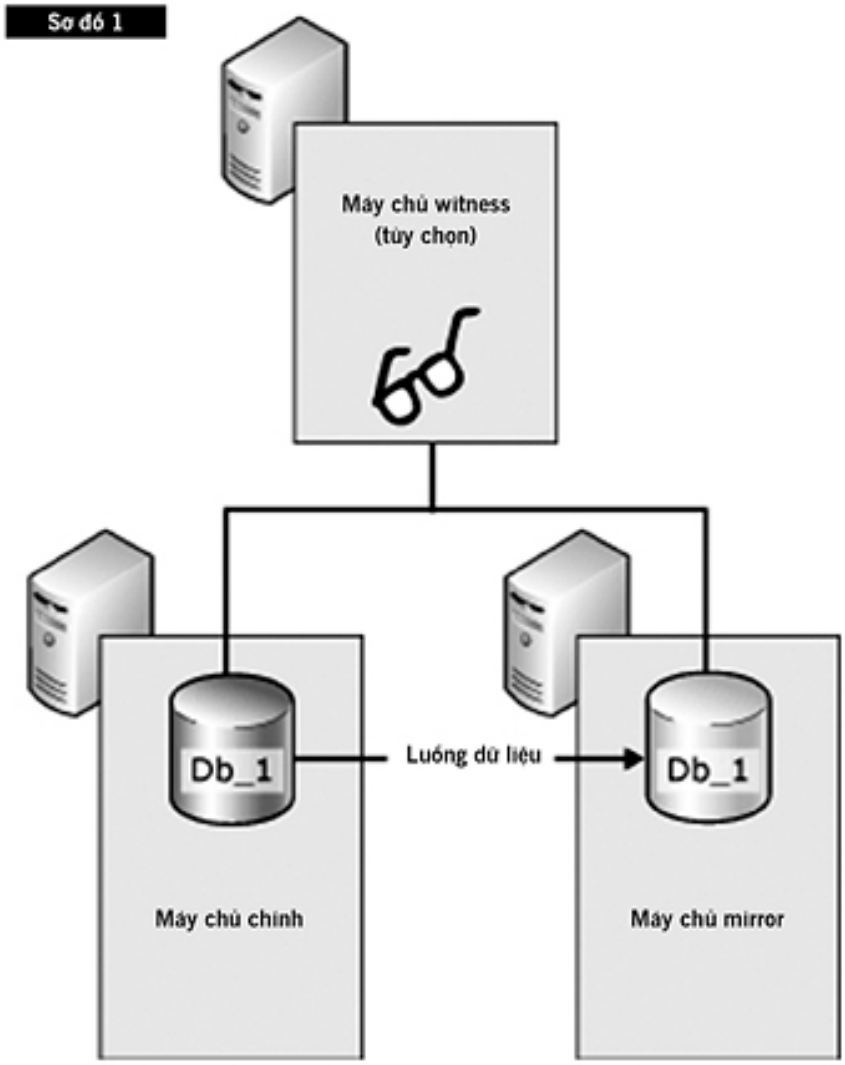
- Determine via domain name. Example: TCP: //PRINCIPAL.DELTAX.COM: 7024.

- Determine through IPv4. Example: TCP: //192.168.1.3: 7024.

- Determine through IPv6.

Note: In case the SQL Server Instance runs on the same machine, the TCP port must be different.

2. Exchange information between the main server and the secondary server



High-speed mode (High-Performance):

High-Performance mode corresponds to asynchronous copy creation. The main server sends updates to the mirror server and continues to make other changes without the need for a mirrored mirror server to be successfully updated.

By not waiting for the mirror server to update changes, the main server has faster access speed and avoids unnecessary loading.

This process can be illustrated by the following scheme:

For high-speed mode, the mirror server always updates more slowly than the main server, and data loss may occur in case the main server interrupts the operation without sending data to the server yet. mirror. However, this different database part is relatively small and acceptable. High performance mode - High performance mode is not required to have a Witness server.

High-Safety mode (High-Safety):

Unlike high speed mode, high safety mode uses Synchronous mechanism. When the application or user updates, it will be updated almost simultaneously on both the main server and the mirror server. This will ensure that when the main server has a problem, the mirror server will have a complete and complete copy of the database, thus ensuring high data security.

High security mode requires a witness server to ensure hot standby - hot standby.

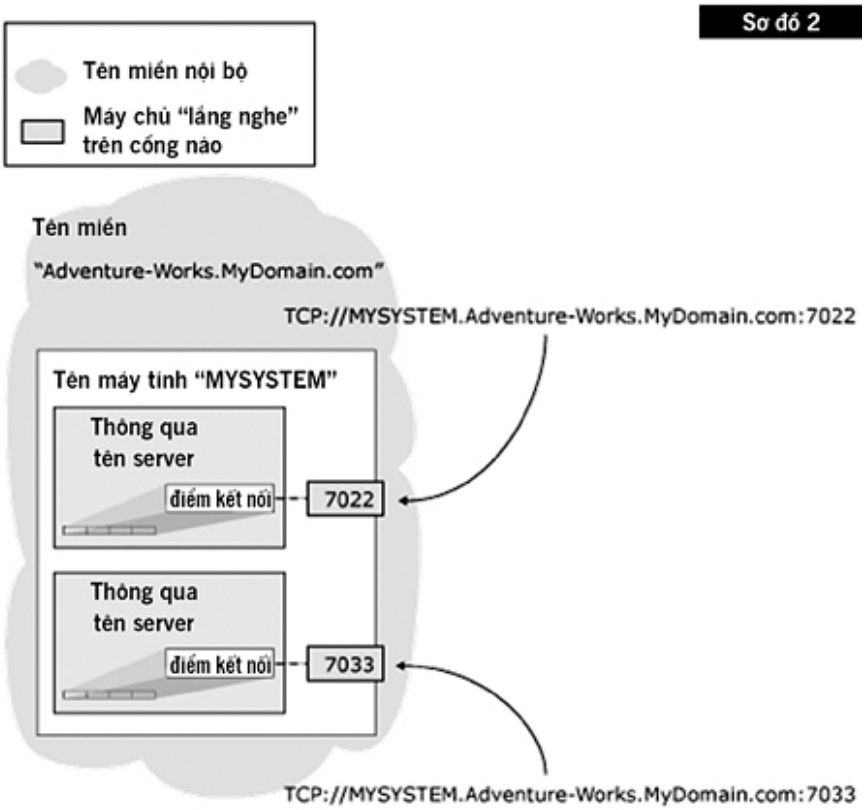
3. DM configuration:

Configuring DM on SQL Server consists of 3 basic steps:

- Backup (backup) the entire database on the main server and then restore (restore) on the mirror server.

- Create corresponding endpoints so that main servers, mirrors and witnesses work together.

- Create a session (Database Mirroring Session)



Configuring DM can be done using the interface of SQL Server Management Studio (SSMS), or can be configured with T-SQL.

Configuring with the interface of SQL Server Management Studio is quite simple, after having successfully restored data on the mirror server, just right-click on the database and select ' *Mirroring* ', then follow the steps .

The result will be the initialization of a DM session.

SSMS configuration allows to remove most operations when done with T-SQL, but if you want, you can still use T-SQL to achieve the highest flexibility.

Configuration on T-SQL can use 2 ways to authenticate each other endpoints: Verify by login or authentication by certificate. The T-SQL code is complete so the DM configuration is quite long, so we only introduce some typical steps.

Configuration with Login

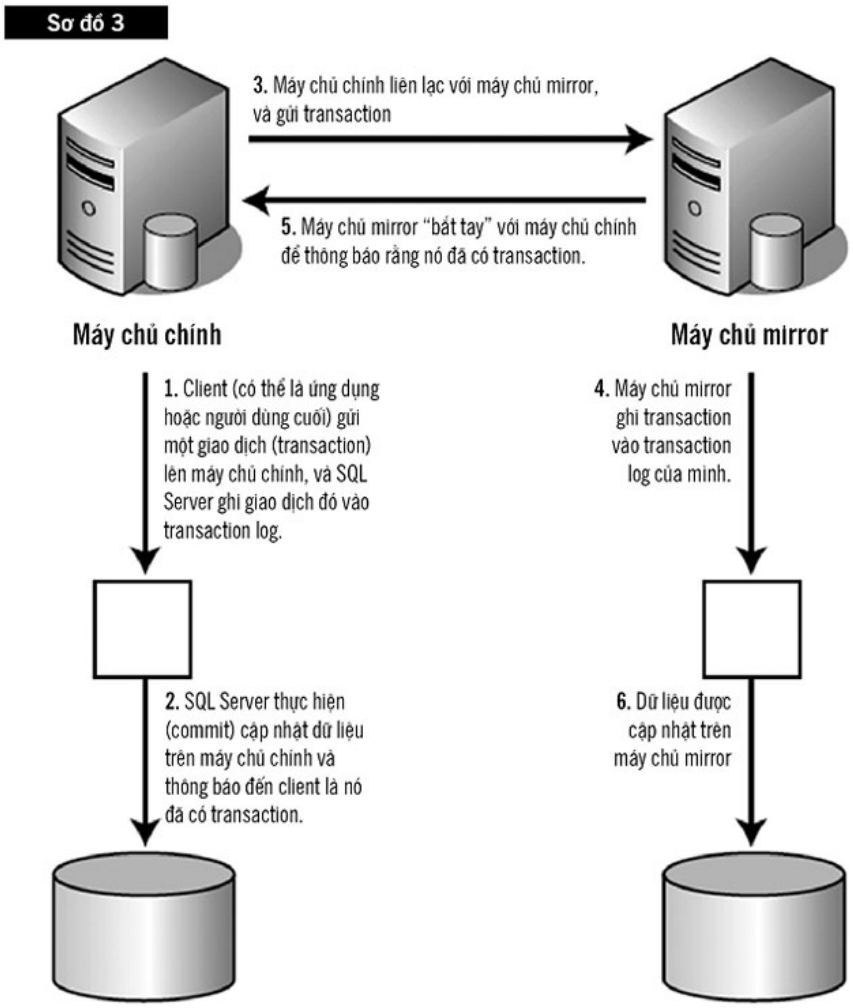
Suppose we use a Windows account to log in to SQL Server, in the case of using a SQL Server account the same way.

```
CREATE LOGIN [PRICIPAL-SRVAdministrator]
FROM WINDOWS
GO
```

Create endpoints:

```
CREATE ENDPOINT Partner
STATE = STARTED
AS TCP (LISTENER_PORT = 5022)
FOR DATABASE_MIRRORING (
AUTHENTICATION = WINDOWS NEGOTIATE,
ENCRYPTION = SUPPORTED,
ROLE = ALL)
GO
```

Note that creating endpoint with **ROLE = ALL** needs to perform on both principal and mirror servers, on the witness server, you replace with **ROLE = WITNESS** .



Configuring by Certificate:

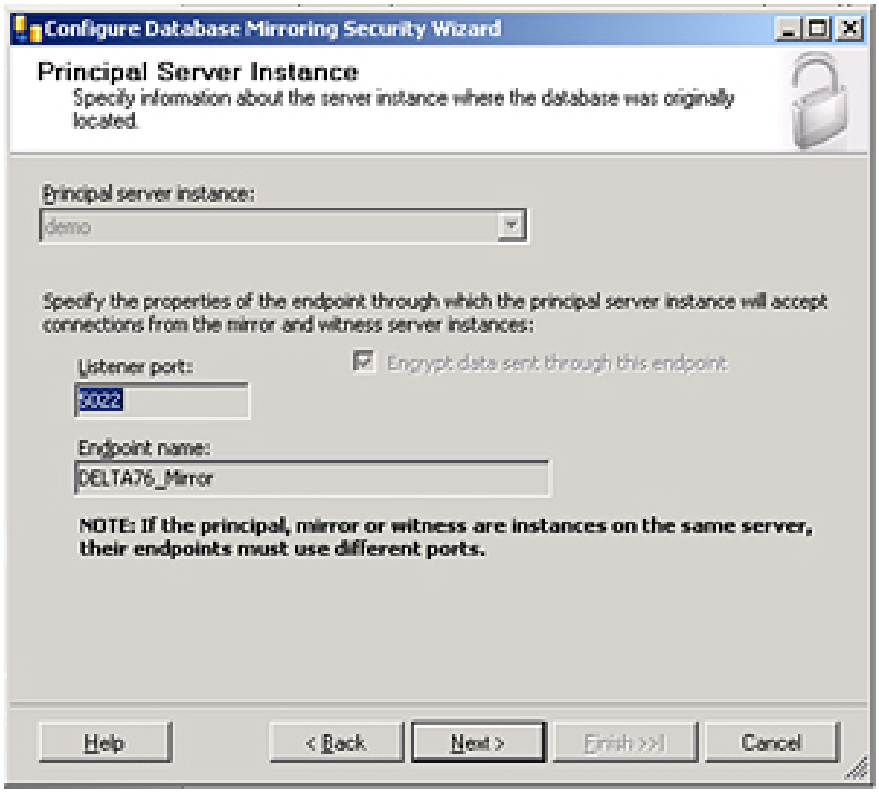
Instead of using a login account to let each endpoint identify each other, it is possible to use an alternative solution to create certificates - the certificate.

- Create master key encryption (required to export certificate):

t?o ng??i dùng key xác th?c t? m?t kh?u = 'abc123 !!';

- Create a cerfiticate:

```
create certificate PRINCIPAL_cert
with subject = 'PRINCIPAL certificate',
start_date = '2007/11/01',
expiry_date = '2020/11/01';
```



- Create the endpoint corresponding to the certificate:
Create endpoint endpoint_mirroring state = started
as tcp (listener_port = 7024, listener_ip = all)
for database_mirroring (authentication = certificate PRINCIPAL_cert, encryption = disabled, role = all);

- Export the certificate to a separate file:

Backup certificate PRINCIPAL_cert to file = 'c: PRINCIPAL_cert.cer';
Do the same on the mirror server and witness, pay attention to changing the role = witness when needed.After creating the Endpoints and exporting the certificates on all 3 instances, returning to principal server:

- Create a login for the mirror server:

create login MIRROR_login with PASSWORD = 'abc123 !!';
GO

- Create a user corresponding to that Login

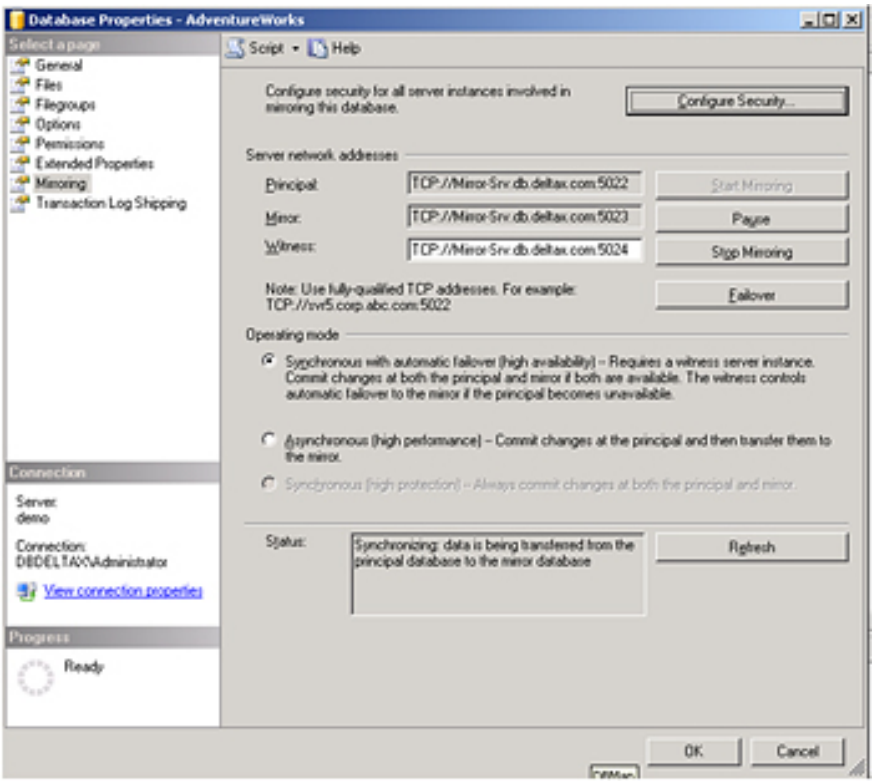
create user MIRROR_user from login MIRROR_login;
GO

- Create a certificate from the server's .cer file:

Create certificate MIRROR_cert
Authorization MIRROR_user
From file = 'c: MIRROR_cert.cer';
GO

- Granting permission to connect to the endpoint for the server's login mirror:

Grant CONNECT ON Endpoint :: endpoint_mirroring to [MIRROR_login];
GO



Do the same thing for the witness server certificate, as well as on mirror and witness servers so that 3 computers can identify and authenticate each other.
After creating the endpoints, you can check them with the query:

SELECT name, state_desc, role_desc
FROM sys.database_mirroring_endpoint

The final task is to initiate a session for the DM:

** On principal server:*

ALTER DATABASE AdventureWorks
SET PARTNER = 'TCP: //mirror-srv.deltax.com: 5022'
GO

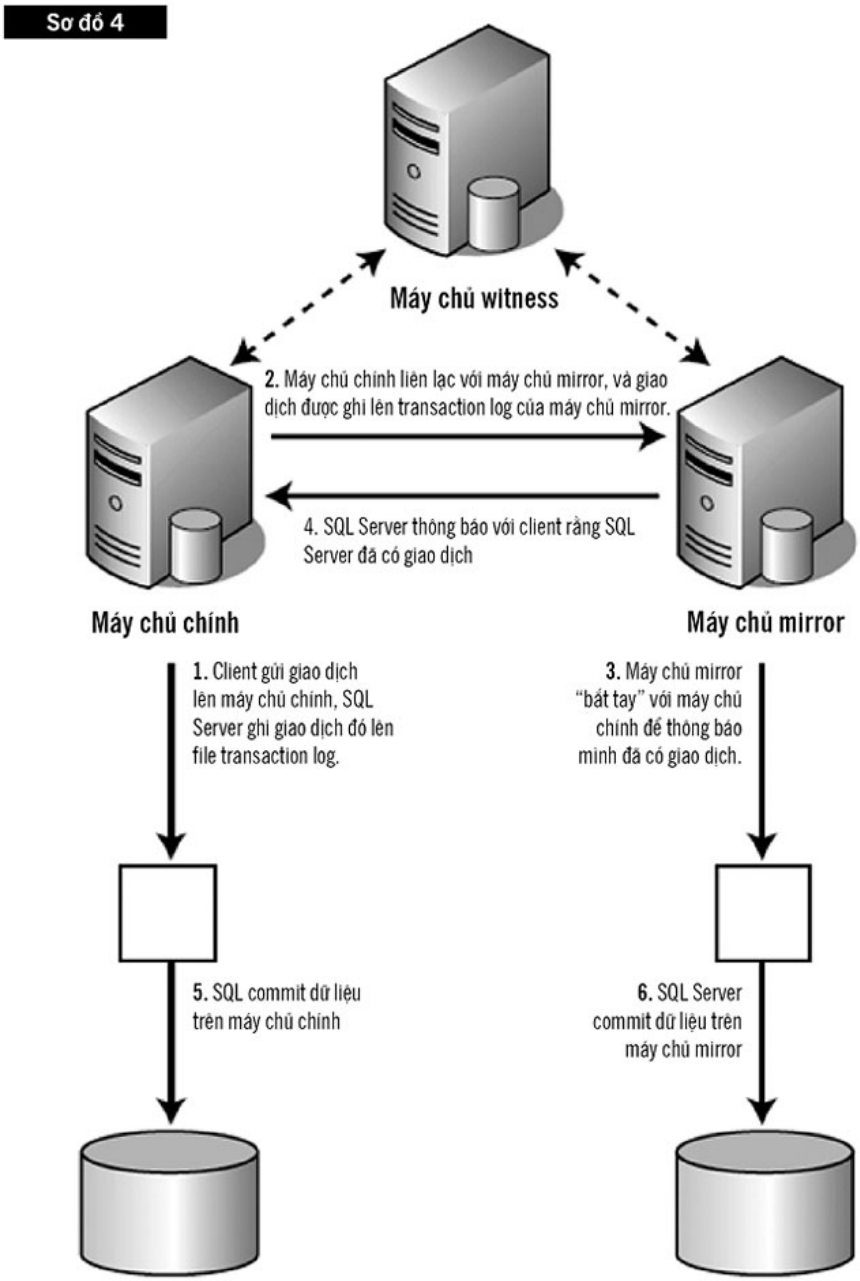
** On the mirror server:*

ALTER DATABASE AdventureWorks
SET PARTNER = 'TCP: //pricipal-srv.deltax.com: 5022'
GO

** On principal server, set up a witness server:*

```
ALTER DATABASE AdventureWorks
SET WITNESS = 'TCP: //witness-srv.deltax.com: 5022'
GO
```

After the system has been in operation, it can be monitored with **Database Mirroring Monitor** tool **:**



4. Programming middleware:

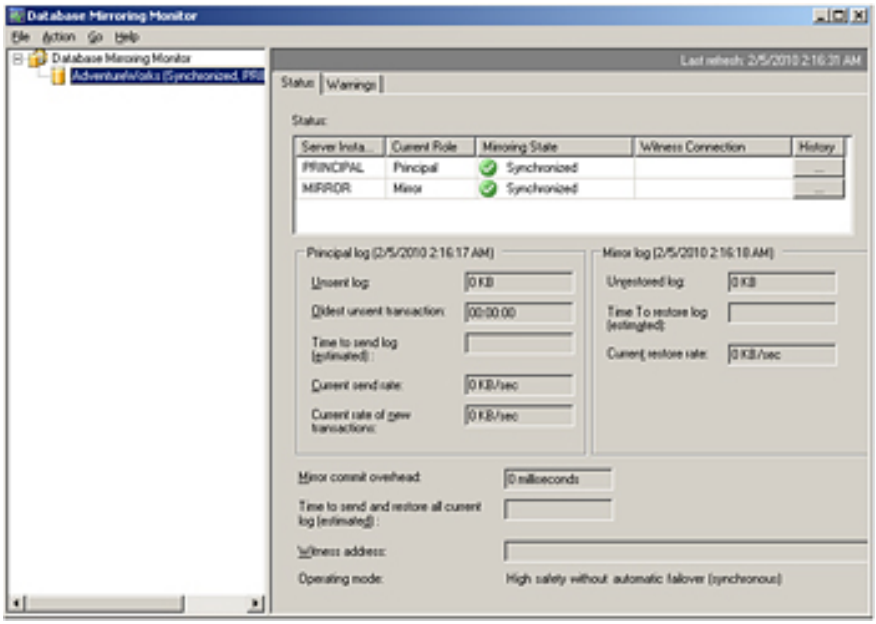
The use of DM is said to be almost transparent for database connection from the middleware side. If you use the ADO.NET library, simply modify the **ConnectionString** to add the ' **failover partner** ' field to the mirror server, for example:

Data Source = pricipal.database.com; Failover Partner = mirror.database.com; Initial Catalog = AdventureWorks;
Integrated Security = True;

In addition, ADO.NET creates a ' **connection pool** ' that allows caching of connections that have been initiated, so in the event of a problem that leads to a server switch, you need to actively implement additional operations. delete this cache.

SqlConnection.ClearPool (conn);

Epilogue



DM in SQL Server is quite simple, easy to configure, use and monitor, but its capabilities are relatively limited. It is only suitable for medium-sized and lower-level databases, and for large databases that have strict requirements for continuity, the proposed approach has not been met, but the overall operating system solutions are needed. hardware system, network.
The article hopes to help you get an overview of how to build a highly available database management system in SQL Server.

Reference: http://technet.microsoft.com/en-us/library/cc917680.aspx

Good luck.